UNIVERSITA' DEGLI STUDI DI
NAPOLI FEDERICO II

Scuola Politecnica e delle Scienze di Base

Corso di Laurea Magistrale in Ingegneria Informatica

Tesi di Laurea Magistrale in Progettazione e Sviluppo di Sistemi Software

Analisi di Issue, Pull Request e Test in progetti open-source di Realtà Aumentata

Anno Accademico 2019/2020

Relatore

Ch.mo Prof.ssa Anna Rita Fasolino

Correlatore

Ch.mo Prof. Porfirio Tramontana

Candidato

Enzo Troisi

matr. M63/000895

Indice

Introduzione

Negli ultimi anni il mondo è fortemente influenzato della tecnologia, e sta vivendo dei grossi cambiamenti che stanno caratterizzando l'universo delle tecnologie imponendo forti riflessioni sulle modalità di interazione che quotidianamente vengono instaurate con il loro utenti.

Fino a non molti anni fa il digitale veniva considerato un prodotto solo per pochi, quando invece oggi rappresenta un aspetto quasi fondamentale delle nostre vite, non solo il digitale è diventato ormai indispensabile per ogni più piccolo aspetto della nostra quotidianità e influenza il nostro modo di esprimersi con il mondo e di vederlo.

La realtà aumentata fa parte di queste tecnologie che su tutte sta prendendo sempre più piede e forma, trovando ampio spazio di interesse, curiosità, e soprattutto di studio in ambito ingegneristico.

Bisogna tenere però in considerazione che, sebbene i primi riferimenti di studi risalgano alla fine degli anni '60, solamente nei primi anni 2000 si arriva ad una comprensione più matura dei possibili sviluppi di questa nuova tecnologia.

Sul mercato i primi prodotti strettamente correlati alla realtà aumentata, spuntano nel 2009, dove essa arriva definitivamente al grande pubblico, con l'apparizione delle prime applicazioni per smartphone e altri dispositivi mobili.

Come in tutte le tecnologie anche questa richiede grande attenzione ed è frutto di molto studio soprattutto nell'ambito del testing ovvero di testare queste applicazioni dove l'obiettivo principale di queste ricerche è appunto quello di trovare un modo semplice, ma soprattutto efficace per testare e monitorare i malfunzionamenti e gli eventuali bug a tempo di compilazione e progettazione, tutto nel modo quanto più automatizzato possibile.

Capitolo 1: Realtà Aumentata

1.1 Cos'è?

Procediamo inizialmente con una definizione per capire bene di cosa si tratta in modo da approfondire in seguito varie sfumature su come viene utilizzata e le sue potenzialità.

Per realtà aumentata, quindi intendiamo riferirci ad un arricchimento della percezione sensoriale umana tramite delle informazioni, che in generale vengono manipolate e indirizzate elettronicamente, con un dispositivo smartphone et similia per esempio, che non sarebbero, in alcun altro modo, percepibili mediante i sensi dell'uomo.

"Tecnica di realtà virtuale, in inglese augmented reality (AR*), attraverso cui si aggiungono informazioni alla scena reale. Questa tecnica è realizzabile attraverso piccoli visori sostenuti, come i caschi immersivi, da supporti montati sulla testa che permettono di vedere la scena reale attraverso lo schermo semitrasparente del visore (*see-through*), utilizzato anche per mostrare grafica e testi generati dal computer." [1].*

Che cos'è quindi la realtà aumentata?

La risposta più semplice che può essere data è quella di spiegare come funziona, ovvero ad esempio di immaginare di padroneggiare un dispositivo mobile, uno smartphone o un tablet, e di inquadrare un oggetto qualsiasi, potendo poi vedere attraverso il display qualsiasi tipo di informazione "aggiuntiva", un testo, delle immagini, dei filmati o delle animazioni.

Pertanto, gli elementi che definiscono "aumentata" la realtà vengono inseriti mediante un dispositivo mobile, come ad esempio nel più classico dei casi uno smartphone, oppure mediante l'utilizzo della webcam di un computer.

Vi sono anche altri sensori, tipo i dispositivi che sfruttano le percezioni sensoria come quelle di visione, occhiali a proiezione sulla retina, oppure di ascolto come gli auricolari, o ancora come quelli di manipolazione tattile per esempio i guanti.

Tutti questi strumenti, quindi, aggiungono delle informazioni di carattere multimediale alla realtà che già normalmente viene percepita, attraverso i sensi.

Si parla di AR anche quando le informazioni che solitamente vengono considerate aumentate, sono da valutare come una riduzione della quantità di informazioni che molto spesso sono considerate percepibili dalla via sensoriale, sempre con l'obiettivo di rappresentare un contesto più chiara o più utile.

1.2 Tipologie

Fondamentalmente la realtà aumentata si caratterizza in due principali tipi, ovvero quella utilizzata su dispositivo mobile e quella su computer.

Nel dettaglio possiamo considerare la prima, come quella per cui necessariamente il mezzo, ossia lo smartphone, deve essere munito di strumenti interni

quali ad esempio un sistema GPS, per garantire la geolocalizzazione del dispositivo e/o di un magnetometro.

Deve inoltre essere in grado di permettere la visualizzazione in Real Time di flussi video, per esempio attraverso la fotocamera (sia interna che esterna), oltre che spesso di dover consentire un accesso ad Internet in modo da poter trasmettere e ricevere dati online.

Il compito di questo dispositivo sarà quindi quello di inquadrare una visione del mondo reale e dell'ambiente circostante in modo da poter gestire ed applicare le funzionalità di una App di AR.

Una App di AR sarà in grado di sovrapporre, tramite l'astrazione del mondo reale, appresa dai vari punti di interesse che sono stati "geolocalizzati", per esempio un contenuto digitale e/o elementi tridimensionali.

L'altra tipologia di AR è quella trattata su un computer, la quale fa uso di alcuni marcatori, ossia dei disegni stilizzati, che vengono mostrati al mezzo di interfacciamento, in questo caso la webcam.
Questi punti vengono riconosciuti dal computer, e grazie ad essi verranno sovrapposti in Real Time i vari contenuti multimediali, per esempio video, audio, elementi 3D, etc. proposti dalle funzionalità dell'applicazione AR che si sta utilizzando.

1.3 Come funziona?

La realtà aumentata può essere implementata in svariati modi e può essere oltretutto utilizzata mediante dispositivi di diverso tipo.

L'utilizzo più diffuso e conosciuto è quello tramite Smartphone e tablet, dove le app di realtà aumentata funzionano tramite gli strumenti ed ai sensori dei dispositivi in questione, come il giroscopio[1], il GPS, l'accelerometro[2] e la fotocamera.

Grazie all'app di AR, opportunamente installata sul dispositivo che si sta utilizzando, sarà possibile vedere, attraverso il display dello stesso, l'ambiente in cui si trova l'utente arricchito di tutte le informazioni aggiuntive che non sono presenti nel mondo reale.

1.4 Realtà Virtuale

La fotocamera gioca un ruolo quindi fondamentale poiché catturerà le immagini del mondo esterno, le quali attraverso l'app verranno in un certo senso modificare.

Un semplice esempio può essere quello che avviene nel settore del business dove si potrà in quadrare un macchinario e vedere intorno ad essa il funzionamento, il video di manutenzione, o gli ultimi problemi che sono stati riscontrati etc.

Un'altra forma molto conosciuta è quella della Realtà Aumentata che fa uso di occhiali, la quale è stata resa molto famosa grazie ai Google Glass progettati dal Team X-Lab.

In questo caso si tratta di occhiali che vanno indossati, i quali sono dotati di una particolare fotocamera e di un display posizionato sopra all'occhio destro, in cui le informazioni aggiuntive si sovrappongono alla realtà.

[1] Il giroscopio è un dispositivo fisico rotante che, per effetto della legge di conservazione del momento angolare, tende a mantenere il suo asse di rotazione orientato in una direzione fissa.
[2] Un accelerometro è uno strumento di misura in grado di rilevare e/o misurare l'accelerazione, effettuando il calcolo della forza rilevata rispetto alla massa dell'oggetto (forza per unità di massa).

Sono uno dei progetti obiettivamente più famosi di realità aumentata, ma allo stesso tempo non il più riuscito, poiché ci sono altre aziende che hanno proposto e continuato a sviluppare e progettare visori AR come ad esempio Microsoft, Epson e Sony.
Su tutti la Microsoft forse è quella che rappresenta attraverso gli HoloLens la proposta più accattivante del mercato.

"Per ottimizzare la potenza della realtà aumentata, le organizzazioni devono inserire dei professionisti in qualunque fase del processo di implementazione"[2].
La realtà aumentata non deve essere però confusa con la realtà virtuale.

*"La simulazione del mondo fisico (*simulazione del reale*) è un ambito tecnico-scientifico in cui, avvalendosi dell'informatica, dell'interfaccia uomo-macchina (HMI,* Human machine interface*) e di altre scienze applicate, un essere umano interagisce in prima persona, in modo naturale, con un ambiente tridimensionale di sintesi completamente creato dal computer." [3]*

La realtà virtuale, infatti, crea un ambiente totalmente artificiale, costruito al computer, il che rende tutto verosimile, poiché si avvale di tecnologie che riescono a dare la sensazione all'utilizzatore di trovarsi realmente immerso in quel determinato scenario che viene mostrato e rappresentato.

Le informazioni che vengono aggiunte o sottratte dal punto di vista elettronico sono dominanti, tanto che le persone si immedesimano perfettamente in una situazione, che riesce a dare delle percezioni dei cinque sensi non sembrano essere più presenti le quali vengono sostituite da altre.

1.5 Similitudini e differenze tra Realtà Aumentata e Virtuale

Innanzitutto, possiamo considerare il fatto che nella realtà aumentata, la persona continua a vivere il mondo reale, ovvero quella che viene definita realtà fisica, ma sfrutta le informazioni che vengono aggiunte e/o manipolate della realtà stessa.

Una differenza tra AR e VR, considerata per certi versi artificiosa, dove la realtà viene mediata, nel senso che può essere considerata come un continuo, dove queste due realtà si posizionano in modo adiacente e non sono semplicemente visti come due concetti opposti.

La mediazione avviene solitamente in real time, e le informazioni del mondo reale che circonda l'utente possono diventare interattive e manipolabili digitalmente.

La realtà aumentata, inoltre, implica un'esperienza di immersione completa che include il mondo reale, che viene però modificato con l'aggiunta di animazioni e contenuti digitali che consentono di avere una conoscenza più approfondita dell'ambiente che ci circonda.

1.6 Origine

Una primissima apparizione di questa, ancora primordiale, tecnologia può essere fatta risalire a degli occhiali di realtà aumentata in un lavoro di Ivan Sutherland[3].

"The ultimate display would, of course, be a room within which the computer can control the existence of matter. A chair displayed in such a room would be

[3] Ivan Edward Sutherland è un informatico e ricercatore statunitense, pioniere di internet, vincitore del Premio Turing nel 1988 per l'invenzione del software e Sketchpad, predecessore delle interfacce maggiormente utilizzate nella computer grafica. A lui si deve anche l'ideazione degli occhiali per la realtà virtuale.

good enough to sit in. Handcuffs displayed in such a room would be confining, and a bullet displayed in such a room would be fatal. With appropriate programming such a display could literally be the Wonderland into which Alice walked." [4]

Traduzione:*" Il display finale sarebbe, ovviamente, una stanza in cui il computer può controllare l'esistenza della materia. Una sedia esposta in una stanza del genere sarebbe abbastanza buona per sedersi. Le manette esposte in una stanza del genere sarebbero limitanti e un proiettile esposto in una stanza del genere sarebbe fatale. Con una programmazione appropriata, un tale spettacolo potrebbe letteralmente essere il Paese delle Meraviglie in cui Alice è entrata."*

Sutherland nel suo saggio incluse più di una semplice descrizione di display immersivi, in cui descriveva con stupore le sue scoperte riguardanti la possibilità di riuscire a vedere attraverso questi dispositivi "attraverso gli oggetti solidi" come se questi fossero trasparenti.[5]
Intorno alla fine degli anni '60, Sutherland riuscì a costruire una sorta di primo sistema VR, infatti nel 1968 terminò il primo display montato sulla testa, ma a causa del suo peso, doveva essere appeso al soffitto e per questo fu appropriatamente soprannominato "Spada di Damocle", come mostrato nella figura sottostante.

Figura 1 – I primi occhiali come sistema VR

Questo display includeva già il rilevamento della testa e utilizzava ottiche trasparenti.

L'anno 1992 ha segnato la nascita del termine "realtà aumentata".

Questo termine è infatti apparso per la prima volta nel lavoro di Caudell[4] e Mizell[5] alla Boeing[6], che ha cercato di assistere i lavoratori in una fabbrica di aeroplani visualizzando gli schemi di assemblaggio di fasci di cavi in un HMD trasparente.

“The enabling technology for this access interface is a heads-up (see-thru) display head set (we call it the “HUDset”), combined with head position sensing and workplace registration systems. This technology is used to “augment” the visual

[4] Thomas Caudell, Professore e Ricercatore del Department of Electrical and Computer Engineering
[5] David Mizell, Computer Scientist
[6] Boeing è un'industria aeronautica statunitense produttrice di velivoli per uso sia civile che militare.

field of the user with information necessary in the performance of the current task, and therefore we refer to the technology as “augmented reality” (AR).”[6]

Traduzione: “La tecnologia abilitante per questa interfaccia di accesso è una cuffia con display heads-up (see-thru) (chiamiamo è l '“HUDset”), combinato con la posizione della testa sistemi di rilevamento e registrazione sul posto di lavoro.
Questa tecnologia viene utilizzata per "aumentare" la visuale campo dell'utente con le informazioni necessarie in lo svolgimento dell'attività corrente e quindi ci riferiamo alla tecnologia come "Realtà aumentata" (AR).”

Negli anni ’90 sono stati effettuati numerosi investimenti economici da parte soprattutto di alcune Università e di gruppi di ricerca in Usa, Germania e Giappone, che hanno fondato numerosi programmi industriali, con l’obiettivo di studiare queste tecnologie, fino ad arrivare ai primi anni 2000, quando i telefoni cellulari e il mobile computing hanno iniziato a evolversi rapidamente.

Tra il 2003 ed il 2005, Wagner[7] e Schmalstieg[8], hanno presentato il primo sistema AR portatile che funziona in modo autonomo su un "assistente digitale personale", un precursore degli smartphone di oggi.

“To demonstrate the power of our solution we have implemented an application that guides a user through an unfamiliar building to their destination. The PDA is used

[7] Daniel Wagner,Vienna University of Technology, Graz University of Technology, Qualcomm
[8] Dieter Schmalstieg, Professore di Computer Graphics and Virtual Reality, Graz University of Technology

as a see-through video device as well as an input device. The application tracks fiducials attached to the building's walls using an optical tracking toolkit."[7]

Traduzione: "Per dimostrare la potenza della nostra soluzione, abbiamo implementato un'applicazione che guida un utente attraverso un edificio sconosciuto fino alla sua destinazione. Il PDA viene utilizzato sia come dispositivo video trasparente che come dispositivo di input. L'applicazione tiene traccia dei fiducials attaccati alle pareti dell'edificio utilizzando un toolkit di tracciamento ottico."

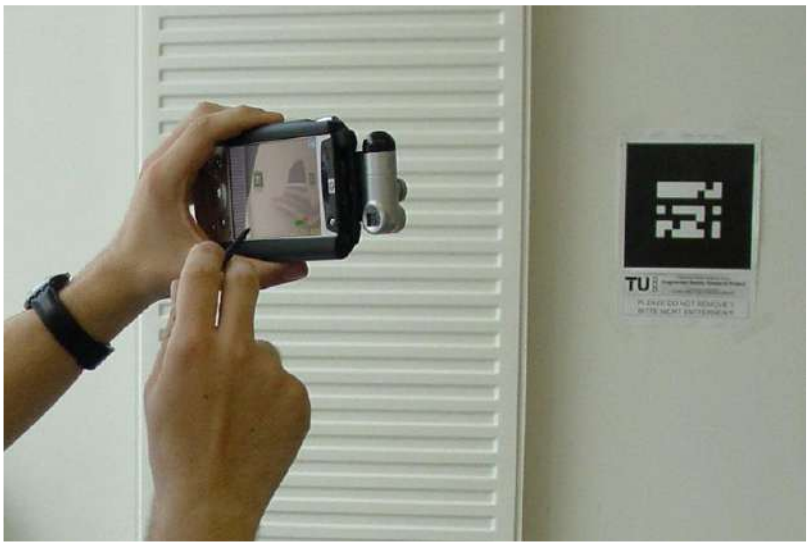

Figura 2 – PDA, il primo sistema AR portatile

Ci sono voluti diversi anni, fino al 2008, per l'introduzione del primo sistema di tracciamento delle caratteristiche naturali veramente utilizzabile per gli smartphone. Questo lavoro è diventato l'antenato del popolare toolkit Vuforia[9] per gli sviluppatori AR.

[9] Vuforia Engine è la piattaforma più utilizzata per lo sviluppo AR, con supporto per i principali telefoni, tablet e occhiali. Gli sviluppatori possono aggiungere facilmente funzionalità avanzate di visione artificiale alle app Android, iOS e UWP, per creare esperienze AR che interagiscono realisticamente con gli oggetti e l'ambiente.

Oggi, gli sviluppatori AR possono scegliere tra molte piattaforme software, ma questi sistemi modello continuano a rappresentare direzioni importanti per i ricercatori.

1.7 Uno sguardo al futuro

La realtà aumentata è destinata già nei prossimi anni a cambiare quasi radicalmente le nostre abitudini, così come il modo di vedere il mondo.

È sicuramente una rivoluzione che può essere definita silenziosa, ma che a sua volta sta portando a risultati molto interessanti visibili già in molti settori, da quello medico a quello sportivo, dal marketing all'intrattenimento, dall'educazione al divertimento.

L'obiettivo della realtà aumentata sarà anche e soprattutto quello di dare un aiuto all'uomo nella vita quotidiana, nello svolgimento di normali attività, come ad esempio guidare un'auto, esistono infatti dei navigatori satellitari che riescono a sfruttare la fotocamera di un dispositivo mobile, la quale puntata sulla strada fornisce informazioni in tempo reale sulla intensità del traffico, oppure nella ricerca di luoghi di interesse.

Molte innovazioni nasceranno per dare grandi opportunità anche per le aziende, basti pensare all'utilizzo dei visori che saranno in grado di monitorare processi e flussi di produzione, permettere di controllare lo stato di un cantiere a distanza attraverso delle visuali 3D.

Diversi studi, come quelli de Ventana Research, hanno rilevato che: "Quasi la metà (46%) delle organizzazioni vedono la tecnologia mobile come un gioco un ruolo fondamentale nel migliorare le informazioni sui prodotti gestione.

L'AR presenta una vasta gamma di ricerche di settore rileva che fino a uno un terzo dei clienti ne utilizza una qualche forma oggi. L'adozione continuerà ad

aumentare – AR ha un valore dimostrabile nel marketing e nella vendita perché genera entrate e migliora l'esperienza d'acquisto. Le aziende trarranno vantaggio dalle innovazioni in AR come esso diventa chiaro che la tecnologia consentirà loro di realizzare il pieno potenziale di investimenti in processi e prodotti esistenti." [8]

Capitolo 2: Diffusione di massa

La realtà aumentata sta ridefinendo le modalità in cui tutti coloro che operano in prima linea apprendono nuove informazioni ed interagiscono in modo del tutto digitale con il mondo fisico che li circonda.
Grande ed ampia diffusione stanno quindi avendo la diffusione di informazioni ed un trasferimento più rapido delle conoscenze, attraverso metodologie di formazione che hanno un accesso più immediato alle competenze in remoto con esperienze della clientela molto avanzate.
Tutto ciò si traduce in una esecuzione ed evoluzione sempre più rapida, ed allo stesso tempo un numero di processi manuali minore con la possibilità di prendere delle decisioni di gran lunga di migliore qualità.

Secondo la Ventana Research: “Per iniziare, le organizzazioni dovrebbero valutare il proprio processi di marketing e vendita e definizione delle priorità come applicare l'innovazione digitale all'acquirente esperienza nell'acquisto di prodotti. L'intento dovrebbe essere quello di sviluppare un avvincente nuova esperienza di acquisto virtuale, che utilizza AR per fornire un modo più completo per gli acquirenti di comprendere e interagire con i prodotti.” [8]

Questo ci fa capire come è diventato importante per le aziende cercare di stare al passo con i tempi dovendo sfruttare tutto quello che è loro a disposizio-

ne, ne sono un esempio alcuni marchi che tra i primi hanno iniziato da utilizzare la realtà aumentate.

Su tutti vi sono Lego, Ikea e Tesco, che addirittura prima della diffusione dei dispositivi mobili, smartphone e tablet, utilizzavano dei tipi di installazione del tutto interattive, o addirittura dei computer domestici dotati di fotocamera.

Un esempio era quello di evitare che alcuni clienti aprissero le confezioni, per far capire il contenuto di esse la Lego usava la realtà aumentata come una sorta di "Smart Packaging", ovvero in un reparto dedicato, inquadrando la confezione dal monitor era possibile visualizzare una sorta di tutorial/trailer con dei personaggi animati coinvolti in una gag.

Un'altra azienda, la Tesco, iniziò ad usare la realtà aumentata su dei volantini per le promozioni, dove da casa il cliente poteva inquadrare col proprio pc il marker[10] che corrispondeva all'oggetto, il quale gli compariva tra le mani.

Permettere la visualizzazione di un televisore in 3D come se fosse davvero presente nel negozio davanti all'oggetto reale è stato senz'altro un modo per migliorare la Digital Experience dandole una serie di informazioni aggiuntive di grande importanza, come ad esempio la dimensione del dispositivo, dando in questo modo al cliente un modo facile ed interattivo per poter valutare il suo ingombro, o ancora le tipologie di ingressi per il collegamento dei vari accessori etc.

La realtà aumentata negli ultimi anni ha iniziato a lavorare per risolvere delle sfide riguardanti la forza lavoro in modo da consentire di acquisire e docu-

[10] Marker: "Augmented reality markers or short AR-markers are visual cues which trigger the display of the virtual information. Markers are normal images or small objects which are trained beforehand so that they can be recognized later in the camera stream." [11]
I marker di realtà aumentata o brevi indicatori AR sono segnali visivi che attivano la visualizzazione delle informazioni virtuali. I marker sono immagini normali o piccoli oggetti che vengono addestrati in anticipo in modo da poter essere riconosciuti successivamente nel flusso della telecamera.

mentare con molta facilità quelle che sono le best practice[11] di esperti, e di presentare contenuti digitali rapportati in un contesto del mondo reale.

La realtà aumentata ha quindi trovato molti ambiti dove poter creare degli applicativi soprattutto in quest'ultimo periodo nell'emergenza sanitaria causata dal coronavirus, dove l'agire e operare a distanza, in modo da rispettare appunto il distanziamento fisico, diventa di fondamentale importanza.

2.1 Settore Industriale

Nel settore industriale la realtà aumentata riesce ad offrire un modo migliore per creare e distribuire delle istruzioni di lavoro devono risultare quanto più fruibili possibile, sia dal punto di vista della gestione che dell'utilizzazione, sovrapponendo contenuti digitali agli ambienti di lavoro reali.
Lo scenario tecnologico della realtà aumentata abilita ampi e numerosi orizzonti applicativi, come l'inquadramento, per esempio, di una stampante il quale sarà possibile spiegare come sostituire la cartuccia attraverso una simulazione animata, o ancora è possibile presentare un macchinario industriale dettagliando le spiegazioni ed i manuali di istruzioni grazie soprattutto al supporto di contenuti che in tempo reale, compaiono sullo schermo mostrando sia l'esterno di ogni componente, video che spiegano il funzionamento di ingranaggi e sistemi, la dinamica dei processi e delle specificità.

2.2 Industria 4.0

La diffusione quindi della realtà aumentata in questo settore è strettamente collegata alla nascita della definizione dei Industria 4.0, la quale prevede una

[11] Esperienza che ha permesso di ottenere risultati eccellenti in un determinato ambito.

digitalizzazione e totale connessione di tutti i macchinari che sono presenti negli stabilimenti adibiti alla produzione.

Figura 3 - Industria 4.0

L'obiettivo è quello di ottimizzare i costi e abbreviare il cosiddetto "time to market[12]", per consentire alle aziende industriali di poter affrontare al meglio delle loro possibilità una migliore competizione globale.

Pertanto, la realtà aumentata gioca un ruolo fondamentale nell'innovazione di queste "smart factory[13]", dove per poter garantire una continuità tra tempi di attività, la qualità dei prodotti e i procedimenti di produzione, bisogna fornire la giusta attenzione sul settore secondario, il quale necessita di una migliore formazione del personale, un maggiore controllo della qualità ed una modalità di progettazione quasi radicalmente innovata.

[12] Il time to market (o TTM) è un'espressione anglofona che indica il tempo che intercorre dall'ideazione di un prodotto alla sua effettiva commercializzazione.

[13] Il concetto di smart factory si compone di tre parti: Smart production, ovvero tutte le nuove tecnologie produttive che creano collaborazione tra tutti gli elementi presenti nella produzione come la collaborazione tra operatore, macchine e strumenti; Smart service, tutte le "infrastrutture informatiche" e tecniche che permettono di integrare con i sistemi, ed anche tutte le strutture che permettono, in modo collaborativo, di integrare le aziende (fornitore – cliente) tra loro e con le strutture esterne (strade, hub, gestione dei rifiuti, ecc.); il terzo componente è Smart energy, in cui si pone un occhio attento ai consumi energetici, creando sistemi più performanti e riducendo gli sprechi di energia secondo i paradigmi tipici dell'energia sostenibile.

Le diverse soluzioni di realtà aumentata possono e devono svolgere un grosso lavoro nella fase di preproduzione, permettendo così di ridurre il tempo per lo sviluppo di nuovi prodotti e i costi relativi ad essi.

Consentire la lavorazione tramite queste applicazioni, può migliorare la comunicazione e la comunicazione delle informazioni all'interno dello stesso gruppo di lavoro, evitando produzioni anticipate di veri e propri prototipi.

Nel settore industriale l'ambito che ha più note queste attività è quello dell'assemblaggio ed alla produzione, i quali tramite l'uso di smart glasses, consentono agli addetti al lavoro di poter lavorare con tutta libertà con le mani, così da garantire un buon supporto nel quotidiano svolgimento delle attività, soprattutto in quelle più complesse, suggerendo loro anche le azioni giuste da compiere passo dopo passo.

Figura 4 - Smart Glasses

2.3 Teleassistenza nell'automobilismo

L'industria ha iniziato ad usare la realtà aumentata, come detto fin ora, in svariati settori industriali e per diversi motivi, tra questi vanno menzionati anche quello dell'assistenza e della formazione.

Nell'ambito dell'automobilismo alcuni brand usano questo tipo di tecnologia per fare teleassistenza, ma anche per le presentazioni dei nuovi modelli, negli show room, ed in varie mostre.

Figura 5 - Realtà aumentata usata dalla Jeep [9]

In figura 5 è possibile vedere come alcune compagnie, in questo caso la Jeep, abbia usato questa tecnologia per mostrare ai clienti, in attesa del lancio commerciale del modello "Jeep Compass", varie caratteristiche e visualizzare gli interni, come se l'auto fosse fisicamente presente in concessionaria.
Mediante il proprio dispositivo si può fare un giro intorno alla propria auto, configurata a seconda delle proprie esigenze, scegliendo colorazioni, optional, e rivestimenti sia interni che esterni, cambiamenti che avvengono in tempo reale, con un semplice "click".

Sempre in ambito automobilistico, anche nelle officine meccaniche il controllo di alcune parti del motore o dell'impianto elettrico, per i vari monitoraggi e manutenzione possono essere risolti con la realtà aumentata, la quale offre delle info molto dettagliate per ogni singola parte su cui vanno effettuati i dovuti controlli o interventi.

Le informazioni possono essere mostrate come dei cartelli "pop-up" vicino ai vari componenti, oppure come dei "video tutorial" di un tecnico che esegue in modo corretto le procedure per intervenire sulla macchina.

2.4 Medicina

Importante impiego di questa innovazione digitale sta assumendo anche il settore medico, con lo sviluppo di nuovi prodotti e servizi, consentendo così un futuro sempre più hi-tech, anche se ad oggi queste tecniche sono ancora gran parte di fase di sperimentazione, ma ovviamente ci si può aspettare che nei prossimi anni possa entrare a far parte della prassi medica e che diventerà una pratica corrente grazie soprattutto ai benefici dell'accessibilità che possono essere apportate.

In linea generale i campi di maggiore applicazione in medicina sono:

- La riabilitazione cognitiva e motoria;
- La terapia per i disturbi psichiatrici;
- L'apprendimento in un contesto del tutto simulativo.

Nella riabilitazione il trattamento della realtà aumentata può essere quindi impiegato sia dal punto di vista fisico-organico che cognitivo-psicologico, ossia rispettivamente, per esempio, per riattivare la mobilità delle articolazioni e per la cura dei disturbi mentali.

Vengono applicati dei veri e propri trattamenti, gestiti tramite dei programmi e sessioni volte ad ottenere un recupero fisico, o a migliorare i deficit nelle aree di funzionalità compromesse.

2.4.1 Tecniche per la radiologia

Il modello di realtà aumentata è stato adottato anche nella elaborazione di immagini TAC e RMN, dove un metodo che viene sempre più utilizzato ed introdotto nelle sale operatorie e quello che consente ai medici, specialisti nel campo operatorio relativo, di sovrapporre immagini computerizzate, sia a fini di "apprendimento", ma anche in modo "preventivo" per addestrarsi sul simulatore, in modo da ottimizzare al massimo i risultati.

Grazie alla combinazione di immagini bidimensionali che vengono mostrate tramite la risonanza magnetica e all'utilizzo di visori, il medico sarà in grado di visualizzare in maniera più semplice e dettagliata l'analisi delle strutture anatomiche, e dell'individuazione di eventuali anomalie negli organi interni.

Tra i principali campi di applicazione della medicina vi sono l'urologia, la chirurgia generale, la cardiochirurgia e la ginecologia.

2.5 Educazione

Su tutti, l'equipe di "Google for Education" cerca di sostenere l'insegnamento e l'apprendimento dovunque, in qualsiasi momento e su qualsiasi dispositivo. Infatti, negli ultimi cinque anni sono state rese possibili delle esperienze di apprendimento di realtà aumentata molto coinvolgenti per milioni di studenti in tutto il mondo tramite Google Esplorazioni e Tour Creator.

Mentre le scuole in tutto il mondo cercano di re-immaginare l'istruzione da zero per un approccio ibrido, sono state pensate con attenzione e cura nei

dettagli il modo di come poter ottimizzare determinati strumenti per adeguarli e renderli accessibili nel presente e, al tempo stesso, adatti per il futuro.

Le esperienze immersive tramite visori hanno consentito, seppur non sono sempre accessibili a tutti gli studenti, rendendo la transizione a una didattica mista più difficile.

2.6 Coronavirus

Per via dell'emergenza sanitaria causata dal Coronavirus (Covid-19), la realtà aumentata ha trovato anche qui nuovi ambiti per sviluppare applicativi che ben si adattano ad affrontare questa esigenza dovuta dal distanziamento fisico.

Un esempio è Würth, una azienda leader mondiale nella distribuzione di prodotti e sistemi per il fissaggio e l'assemblaggio, che grazie alle collaborazioni con Microsoft Italia ha sviluppato "HoloMaintenance Link".

Questa è una piattaforma che permette agli artigiani di poter gestire in modo totalmente interattivo la risoluzione da remoto delle richieste di consulenza, di assistenza e di manutenzione.

Il cliente, che decide di sfruttare questa opportunità, potrà ricevere rapidamente assistenza dal proprio artigiano di fiducia, dove cliccando su un link ricevuto via SMS o mail, avviando una videochiamata con computer o con dispositivo mobile.

Da qui entra in gioco la realtà aumentata, in cui il cliente inquadrando con la fotocamera del proprio dispositivo il macchinario che necessita di assistenza cosicché il tecnico potrà individuare la causa del guasto o del malfunziona-

mento in modo del tutto remoto guidare il suo cliente nell'esecuzione delle operazioni necessarie.

Una volta terminata l'assistenza il sistema genererà in modo automatico un ticket con lo storico della chiamata, comprese tutte le operazioni svolte in realtà aumentata.

La piattaforma può essere utilizzata anche indossando il visore "HoloLens 2", permettendo di svolgere l'assistenza remota in casi più complessi, dove indossando il visore, il tecnico potrà visualizzare nella realtà fisica, da ogni punto di vista, ed una versione digitale tridimensionale dei componenti del macchinario interagiranno con essi.

Capitolo 3: Strumenti

Da smartphone e tablet ai visori Google Glass e HoloLens, le istruzioni AR possono essere facilmente pubblicate e visualizzate su una vasta gamma di dispositivi in tutta l'azienda.

Figura 6 - Visore RealWear

3.1 Google glass

Google Glass è una marca di smart glasses, un HMD[14] che ha la forma di un paio di occhiali, sviluppati da X[15], i quali sono dotati di realtà aumentata, at-

[14] Un head-mounted display (in italiano schermo montato sulla testa), o HMD, è uno schermo montato sulla testa dello spettatore attraverso un casco ad hoc e può essere monoculare o binoculare. Ad oggi viene usato in campo aeronautico o per allenamenti e simulazioni, ma sono in sviluppo progetti per un uso ludico, didattico educativo e in campo medico.

traverso di essi è possibile visualizzare delle informazioni come avviene sugli smartphone senza l'uso delle mani (*hands-free*).

I Glass funzionano tramite l'uso di comandi vocali, e rientrano nel concetto di *ubiquitous computing*[16].

Figura 7 - Google Glass

Il prodotto, Google Glass Explorer Edition, è stato reso disponibile per gli sviluppatori di Google I/O, negli Stati Uniti, verso i primi mesi del 2013, ma per ampliare il programma di Explorer, Google decise, circa un anno dopo, di renderli disponibili a tutti coloro fossero residenti negli Stati Uniti, e che li volessero acquistare.

Questa possibilità fu però limitata per alcuni giorni fino ad esaurimento delle scorte, per poi iniziare la grande distribuzione del dispositivo solo successivamente, a partire dalla Gran Bretagna.

Nell'aprile del 2013 Google pubblicò le specifiche tecniche di questi dispositivi, che prevedevano [10]:

- Aste regolabili e telaio resistente, adatti a qualsiasi viso

[15] X è una struttura semi-segreta gestita da Alphabet, che si occupa di sviluppare importanti innovazioni tecnologiche. Fondata nel 2010 da Google Inc., era nota coi nomi Google X Lab e Google X.

[16] Lo ubiquitous computing (*computazione ubiqua*) è un modello post-desktop di interazione uomo-macchina, in cui l'elaborazione delle informazioni è stata interamente integrata all'interno di oggetti e attività di tutti i giorni.

- Display ad alta risoluzione, equivalente ad uno schermo HD da 25 pollici da due metri di distanza
- Fotocamera da 5MP e registrazioni video a 720p
- Audio a conduzione ossea
- Wi-Fi - 802.11 b/g
- Bluetooth
- 12 GB di memoria utilizzabile, sincronizzato con Google Cloud Storage compatibile con qualsiasi cellulare Bluetooth

3.2 Microsoft HoloLens

Microsoft HoloLens, noto anche come "Project Baraboo" in fase di sviluppo, sono degli smart glass a realtà mista, prodotti dalla Microsoft, la cui prima versione, detta "Development Edition", è disponibile da marzo 2016, pensata soprattutto per sviluppatori.

HoloLens viene considerato il primo display che è stato montato sulla testa che eseguiva la piattaforma "Windows Mixed Reality"[17] con il sistema operativo Windows 10.

La tecnologia che è stata usata per il tracciamento HoloLens può far risalire la sua discendenza a Kinect[18], un componente aggiuntivo per la console di gioco Xbox sempre di casa Microsoft introdotto precedentemente nel 2010.

17 Windows Mixed Reality, in precedenza noto come Windows Holographic, è una piattaforma basata sul sistema operativo Windows 10 per visori di realtà aumentata e Realtà virtuale ideata da Microsoft.

18 Kinect, inizialmente conosciuto con il nome in codice di Project Natal, è un accessorio sviluppato da Microsoft per la console Xbox 360, sensibile al movimento del corpo umano.

Figura 8 - HoloLens

Microsoft HoloLens è il dispositivo che si tratta di un visore senza cavi, un HMD, che monta al proprio interno il sistema operativo Windows 10.

Le lenti usano dei sensori avanzati, oltre ad un display ottico 3D ad alta definizione, che grazie ad un sistema di scansione spaziale dei suoni che consentono a chi li indossa di poter utilizzare applicazioni di realtà aumentata attraverso un'inedita interfaccia olografica con la quale è possibile interagire attraverso lo sguardo, la voce o i gesti delle mani.

Dal punto di vista tecnico HoloLens usa dei sensori con una telecamera di profondità ad alta efficienza energetica, con un campo di 120°x120° di visuale. Altre funzionalità fornite dai sensori sono rappresentate dalla possibilità di effettuare *head-tracking*[19], cattura video, e cattura del suono.

HoloLens dispone, oltre ad una CPU[20] ed una GPU[21], di una HPU, Holographic Processing Unit, ovvero un coprocessore che integra i dati dai vari sensori e

19 L'head tracking basato sull'orientamento è la forma di base e rileva solo la direzione delle rotazioni della vostra testa: da sinistra a destra, dall'alto al basso e il giro simile alla rotazione dell'orologio.
20 Una unità centrale di elaborazione (o central processing unit, sigla CPU), in elettronica e informatica indica l'unità logica e fisica che svolge le funzionalità logiche di elaborazione principali del computer.
21 L'unità di elaborazione grafica (o processore grafico, in inglese graphics processing unit, sigla GPU) è un circuito elettronico progettato per accelerare la creazione di immagini in un frame buffer, destinato all'output su un dispositivo di visualizzazione.

gestisce attività come la mappatura del territorio, il riconoscimento dei gesti ed il riconoscimento vocale.

Una seconda versione di HoloLens, ovvero HoloLens 2, è stata annunciata al nel febbraio 2019, e tra le varie migliorie apportate, il campo visivo è stato raddoppiato, il tracciamento è più affidabile, l'interazione risulta più intuitiva ed è stata incrementata l'ergonomia.

Microsoft ha sviluppato questo dispositivo soprattutto per un campo di impiego industriale.

Capitolo 4: Tools

Un fattore trainante della popolarità delle applicazioni di realtà aumentata è il fatto che esistono numerosi SDK disponibili ed anche la loro facilità d'uso è fondamentale.

Allo stesso tempo la loro facilità di utilizzo ha reso possibile una grande porzione di sviluppo di applicazioni, molto spesso prodotta da sviluppatori con scarsa esperienza nel settore.

Ad oggi esistono diverse piattaforme disponibili su cui poter procedere nella progettazione, e pertanto sono disponibili diverse scelte fornite allo sviluppatore che garantiscono diversi tipi di strumenti per lo sviluppo, che siano sia open source, ma anche altre commerciali, che si estendono in ambienti di runtime.

Vi sono diversi esempi di Tools, e tra i più comuni ed utilizzati come strumenti di sviluppo di applicazioni di realtà aumentata, sono:

- Vuforia Studio, molto probabilmente la più diffusa ed utilizzata, la quale è disponibile per varie piattaforme quali Android, IOS, UWP, Unity;
- ARCore, disponibile sia per Android che per IOS;
- ARKit, disponibile solo per IOS;

- Wikitude, infine, che copre anch'essa diverse piattaforme come Android, IOS, Microsoft Tablet e Smart Glasses.

4.1 Vuforia

Vuforia è uno dei kit di sviluppo software di realtà aumentata (SDK) per dispositivi mobili più utilizzati, il quale permette lo sviluppo di applicazioni per la realtà aumentata.

Figura 9 - Vuforia Logo

Questo SDK utilizza una tecnologia per la visione artificiale in modo da identificare e rappresentare delle immagini planari e degli oggetti tridimensionali in tempo reale.
La sua capacità di rilevazione delle immagini permette agli sviluppatori di poter posizionare e di poter orientare tutti oggetti virtuali, come ad esempio i modelli 3D, in rapporto agli oggetti del mondo reale quando questi verranno visualizzati attraverso la fotocamera di un dispositivo mobile.
L'oggetto virtuale, pertanto, traccia la posizione oltre che l'orientamento dell'immagine in tempo reale, cosicché la prospettiva dello spettatore sull'oggetto corrisponde alla prospettiva sull'obiettivo, ottenendo così un effetto

realistico come se l'oggetto virtuale faccia perfettamente parte della scena del mondo reale.

Vuforia supporta diversi tipi di target 2D e 3D, ed ha come altre funzionalità:

- l'identificazione del dispositivo con 6 gradi di libertà nello spazio,
- il rilevamento dell'occlusione localizzato mediante "pulsanti virtuali",
- la selezione del target dell'immagine in runtime,
- la capacità di creare e riconfigurare i set di target a livello della programmazione in fase di runtime.

Fornisce inoltre le sue API[22] per diversi linguaggi, tra cui C ++, Java, Objective-C++ e .NET tramite un'estensione del motore di gioco Unity, consentendo il supporto per lo sviluppo nativo per iOS, Android e UWP, permettendo anche lo sviluppo di applicazioni di realtà aumentata in Unity, rendendo facile il trasferimento su entrambe le piattaforme.

4.2 ARCore

ARCore è la piattaforma Android per la creazione di applicazioni AR per i dispositivi mobili.

Figura 10 - ARCore Logo

22 Application programming interface (API) indica, in un programma informatico, un insieme di procedure (in genere raggruppate per strumenti specifici) atte all'espletamento di un dato compito, e spesso tale termine designa le librerie software di un linguaggio di programmazione.

Si basa quasi esclusivamente sulla fotocamera principale e sui sensori di movimento che sono integrati nel dispositivo, ciò comporta che gli sviluppatori di applicazioni di realtà aumentata hanno molta più flessibilità nei motori di sviluppo che scelgono di utilizzare e possono persino sviluppare esperienze per dispositivi iOS.

A differenza dei dispositivi più nuovi o più costosi, che consentono esperienze AR più robuste, la maggior parte dei telefoni Android funzionanti sono compatibili con ARCore, infatti funziona utilizzando la fotocamera del dispositivo per rilevare "punti caratteristici" nell'ambiente circostante.

I gruppi di punti caratteristici vengono utilizzati per identificare i piani probabili nel mondo fisico, dove i piani sono superfici continue come muri o pavimenti e soffitti, o parti di strutture più grandi come i piani dei tavoli.

Accedendo alle informazioni sulla posizione, l'orientamento e il movimento del dispositivo mobile, ARCore abilita un processo chiamato Localizzazione e mappatura simultanee , il quale prevede la contestualizzazione della posizione del telefono e dell'ambiente circostante per consentire delle esperienze dinamiche.

ARCore prevede anche la stima della luce per cercare di rendere più credibile il posizionamento di oggetti digitali, inoltre, Google ha recentemente sviluppato un'API Depth che funziona sulla maggior parte dei dispositivi Android prodotti.

4.3 ARKit

ARKit è la piattaforma iOS, sviluppata dalla Apple, per la creazione e l'esperienza di app AR.

Figura 11 - ARKit Logo

Gli ultimi aggiornamento hanno migliorato tutte le esperienze, sia quelle vecchie che quelle nuove, attraverso l'integrazione delle funzionalità LiDAR, disponibili per gli ultimi dispositivi IOS.
La dipendenza da una unica tecnologia, e quindi l'esclusività può rendere certe esperienze di realtà aumentata per ARKit, molto più coinvolgenti e ottenere strumenti più precisi, il che significa allo stesso tempo, maggiori difficoltà per gli sviluppatori nella creazione di applicazioni AR più accessibili.

LiDAR è un acronimo di "Light Detection and Ranging", e serve per determinare la distanza tra un dispositivo e una superficie utilizzando il tempo necessario affinché un impulso luminoso si sposti dal dispositivo alla superficie e viceversa, i quali impulsi di luce generano un unico punto.
Le raccolte di questi punti, denominate "nuvole di punti", vengono successivamente utilizzate per la creazione di una mappa topografica dell'ambiente circostante dell'utente.
Questo metodo crea mappe molto più dettagliate rispetto al modello di Android, oltre al fatto di consentire un posizionamento più rapido dei modelli digitali, richiede anche hardware dedicato che renda i dispositivi più grandi e più costosi.

Il sistema LiDAR di Apple migliora anche l'occlusione degli oggetti e l'occlusione delle persone, consentendo agli tutti oggetti digitali di apparire posizionati dietro sia ad oggetti fisici e alle persone, le capacità di misurazione, di acquisizione del movimento e della fisica degli oggetti all'interno delle applicazioni di realtà aumentata, il che permette la realizzazione di un prodotto migliore.

Come nel caso di ARCore, ARKit gestisce la mappatura del mondo (che Apple chiama "Geometria della scena") tramite interfacce con la posizione del dispositivo e l'hardware di posizionamento. ARKit offre anche integrazioni che consentono di utilizzare contemporaneamente entrambe le telecamere di un dispositivo.

4.4 Wikitude

Wikitude SDK, nato nel 2008, è un framework di sviluppo di applicazioni di realtà aumentata, attraverso l'utilizzo del riconoscimento e del tracciamento delle immagini e di tecnologie per la geolocalizzazione.

Figura 12 - Wikitude Logo

Include il rendering di modelli 3D, overlay dei video, AR basata sulla posizione.

Nel 2017 Wikitude ha lanciato la sua tecnologia SLAM (Simultaneous Localization And Mapping) che consente il riconoscimento e il tracciamento degli oggetti, oltre al tracciamento istantaneo senza marker.
Questa tecnologia permette agli sviluppatori di poter mappare molto facilmente gli ambienti e di poter visualizzare i contenuti di realtà aumentata sprovvisto della necessità di immagini di destinazione (marker).

Object Recognition è l'ultima novità di questo SDK, basata su SLAM, la cui idea alla base è molto rassomigliante all' Image Tracking, con la differenza che invece di riconoscere immagini e superfici planari, essa può lavorare con delle strutture tridimensionali e con gli oggetti fisici, come ad esempio degli strumenti, dei giocattoli, o dei macchinari.

Wikitude è disponibile sia per i sistemi operativi Android, sia per iOS, ma anche per Windows, essendo ottimizzato anche per diversi dispositivi smart eyewear, esse può essere inoltre considerata la prima applicazione resa disponibile in modo pubblico, che utilizzava un approccio del tutto basato sulla posizione alla realtà aumentata.

4.5 Differenze ARCore ARKit

Quando si confrontano due cose qualsiasi, è facile chiedersi se una è "migliore" dell'altra, e quindi come nella maggior parte dei casi, compresi i casi di ARCore e ARKit, non è esattamente molto semplice.

Partendo dalle similitudini, sia ARCore che ARKit fanno un uso di tecniche per il riconoscimento ambientale, per consentire agli sviluppatori di poter lavorare direttamente al netto di sofisticati algoritmi basati su gradienti di luci e calcolo della profondità.

Entrambe le SDK interpretano quindi i gradienti di luce del mondo reale per capire quali sono le sorgenti di luce reale, per poi poter illuminare in maniera analoga gli oggetti virtuali, proprio come se essi fossero disposti nel mondo reale.
inoltre, entrambi i framework consentono l'aggiunta e l'alterazione delle diverse sorgenti di luce virtuale, anche in modo dinamico, ed offrono diversi modi per i dispositivi mobili di mappare, comprendere e ampliare il mondo.

Dal punto di vista delle differenze, quelle tecniche sono minimali il che necessiterebbe di scendere in dettagli implementativi per capire meglio eventuali pro e contro delle scelte adottate da Apple e Google per i rispettivi algoritmi di motion tracking, plane detection e lighting estimation.

Macroscopicamente si può verificare che ARCore riesce a mappare mondi reali abbastanza vasti all'interno di una stessa scena virtuale, senza bisogno di riavviare la sessione AR, identificando semplicemente più feature point nel mondo reale, e facendolo più rapidamente.

ARKit è solitamente più accurato per quanto riguarda la scelta dei feature point, ed inoltre, dal momento che è sul mercato da più tempo ha a suo vantaggio il fatto di avere un'API più matura e con una documentazione più adeguata, ed è per questo spesso considerato il framework preferito da coloro che vogliono cimentarsi nella realtà aumentata.

Tuttavia, la forte dipendenza a LiDAR di ARKit rende anche i dispositivi compatibili con l'edizione più recente più grandi e più costosi, mentre ARCore utilizza una tecnologia già su un telefono standard, rendendo così i dispositivi compatibili non necessariamente dovranno essere più grandi o più costosi per poter supportare la piattaforma.

Oggi ogni dispositivo mobile sul mercato possiede una fotocamera RGB, pertanto quasi tutti saranno compatibili con ARCore ed in una certa misura questo include anche i dispositivi Apple.

Gli sviluppatori che utilizzano ARCore, pertanto, possono sviluppare sia per dispositivi Android e Apple, questo è un vantaggio che gli sviluppatori di ARKit non hanno.

Bisogna tener presente che dal punto di vista della maggior parte degli utenti di dispositivi mobili, uno scontro tra ARKit e ARCore non dovrebbe essere sufficiente per la scelta di un sistema operativo o di un altro, ben diverso è per gli sviluppatori, nel cui caso l'argomento richiede maggior approfondimento.

Quindi il problema non ricade sulla qualità delle piattaforme o sul fatto che una sia di gran lunga migliore dell'altra, ma il tutto si riduce anche al tipo di applicazioni che si desidera sviluppare, a come si vuole/deve svilupparle e ai dispositivi con cui si desidera che l'applicazione sia compatibile.

Non esiste quindi una risposta universalmente ed oggettivamente corretta. Tendenzialmente gli sviluppatori tendono ad intraprendere l'uso di una soluzione, semplicemente scegliendo una strada piuttosto che un'altra.

L'unica cosa che andrebbe fatta è quella di imparare a lavorare su entrambe le piattaforme, in modo da potersi approcciare sia i dispositivi iOS che a quelli Android, indipendentemente dalla propria preferenza di base.

Capitolo 5: Architettura

L'obiettivo della realtà aumentata è principalmente quello di fondere oggetti virtuali nel mondo reale, cercando mi migliorare la percezione dell'utente collegando i due "mondi".

Gli oggetti virtuali visualizzano informazioni che l'utente non può rilevare con i propri sensi, e le informazioni che vengono fornite da essi aiutano l'utente a eseguire attività nel mondo reale.
La realtà aumentata è un chiaro esempio di come utilizzare il computer come strumento per rendere più agevole un'attività per un essere umano.

I componenti che sono alla base di una architettura di un progetto della realtà aumentata sono: il generatore di scansione, il sistema di tracciamento ed il display.
Grazie a questi sistemi i dispositivi di realtà aumentata funzionano in modo abbastanza efficiente, attraverso l'uso principalmente di due approcci basati sui marker e sul posizionamento.
Nel primo approccio, ovvero quello basato sui markers (marcatori), in esso questi elementi verranno utilizzati come riconoscimento per il software che

lavorerà come uno scanner in modo da indentificarli permettere lo svolgimento di determinate funzioni prestabilite.

I markers possono essere ad esempio dei codici a barre, delle figure oppure dei simboli.

Per quanto riguarda l'approccio basato sulla localizzazione, verrà sfruttata la capacità di un particolare dispositivo di riuscire a registrare la sua posizione nell'ambiente e di riuscire quindi ad offrire i dati e le informazioni utili per la posizione stessa.

La figura sotto riportata (figura 13) mostra il semplice sistema di realtà aumentata.

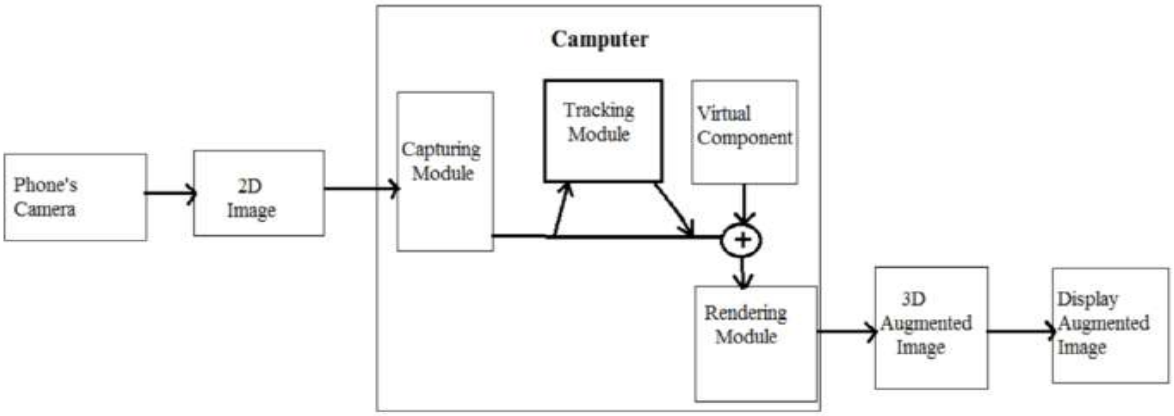

Figura 13 - Architettura applicazione AR [12]

È possibile notare tutte le principali fasi che rappresentano, in modo sintetico, il funzionamento di una applicazione di realtà aumentata.

La suddivisione viene fatta in alcuni blocchi/moduli, in cui è possibile osservare da sinistra verso destra:

- il modulo di capturing (cattura) dell'immagine del mondo reale, realizzato tramite la fotocamera del dispositivo mobile, la quale viene elaborata e trasformata in formato digitale in una immagine bidimensionale;
- il modulo di tracking (tracciamento), calcola la posizione e l'orientamento per la sovrapposizione dell'immagine virtuale;

- componenti virtuali, che unite al modulo di rendering, permetteranno di ottenere un'immagine tridimensionale di realtà aumentata, che verrà poi visualizzata nel display del dispositivo mobile.

5.1 Tracking

Il modulo di tracciamento gioca un ruolo molto importante nel sistema di una architettura di realtà aumentata, il quale viene utilizzato per il calcolo della posizione in tempo reale.

Il modo più semplice per il calcolo della posizione è quello di utilizzare un indicatore, attraverso il quale, grazie al modulo di rendering, verrà successivamente utilizzato per sovrapporre l'immagine virtuale all'immagine presente fisicamente nel mondo reale.

Un marker è quindi un'immagine, un indicatore, che il sistema di tracciamento dovrà rilevare e che utilizzerà per l'elaborazione delle immagini virtuali, attraverso il riconoscimento del particolare pattern[23] usato per il marker.

Le seguenti immagini (figura 15) sono alcuni esempi di marker utilizzati:

Figura 14 – A sinistra un generico esempio di marker, a destra il marker di default di Vuforia

[23] Nella computer graphics, immagine o motivo decorativo che è possibile replicare all'infinito, spec. nella creazione di sfondi.

Nella figura seguente (figura 14) viene raffigurata quella che viene considerata la posizione corretta del marker in relazione al punto di vista della fotocamera, in cui dovranno poi essere calcolate le dimensioni spaziali dell'immagine.

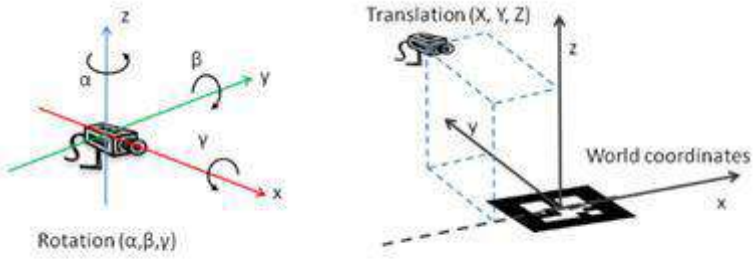

Figura 15 - Posizionamento ottimale di un marker [12]

Quindi il tracciamento del marker non è altro che la definizione di stato e posizione corretti, e l'obiettivo del rilevamento dei marker sarà quello di cercare e rilevare i contorni significativi dei potenziali marker.

L'obiettivo è quindi quello di controllare che i contorni del marker combacino con l'elemento rilevato, il quale rilevamento prevede: [12]

- l'acquisizione dell'immagine;
- l'elaborazione dell'immagine acquisita, in cui viene convertita in una scala di grigi ed eseguito un adattamento delle linee per formare gli angoli;
- il rilevamento dei marker, In questa fase si esegue una rapida controllo per la valutazione di potenziali marker e scartare i non marker;
- il Pattern Matching, che esegue una corrispondenza tra il modello dei marker, identificando i componenti virtuali che vanno posizionati;
- il calcolo della posa, in cui esegue il calcolo della posa per la fotocamera.

5.2 Diagramma delle classi

Il diagramma delle classi in UML mostra tutti i dettagli del modello di riferimento e delle relazioni che vi sono tra i vari componenti dell'architettura.

Le connessioni che vi sono tra le classi sono appuntate tramite delle etichette che mostrano il tipo di dati che vengono scambiati, oltretutto per mantenere la complessità del diagramma, possono essere tracciati solo alcuni dettagli e info scambiate tra i vari diagrammi dettagliati nei singoli sottosistemi presi in considerazione.

Nelle seguenti immagini (figura 16 e figura 17) vengono mostrati il diagramma delle classi inerente, nel primo caso ad una struttura di base generale del modello di riferimento, mentre ne secondo la struttura di riferimento per la realtà aumentata, arricchita da ulteriori dettagli.

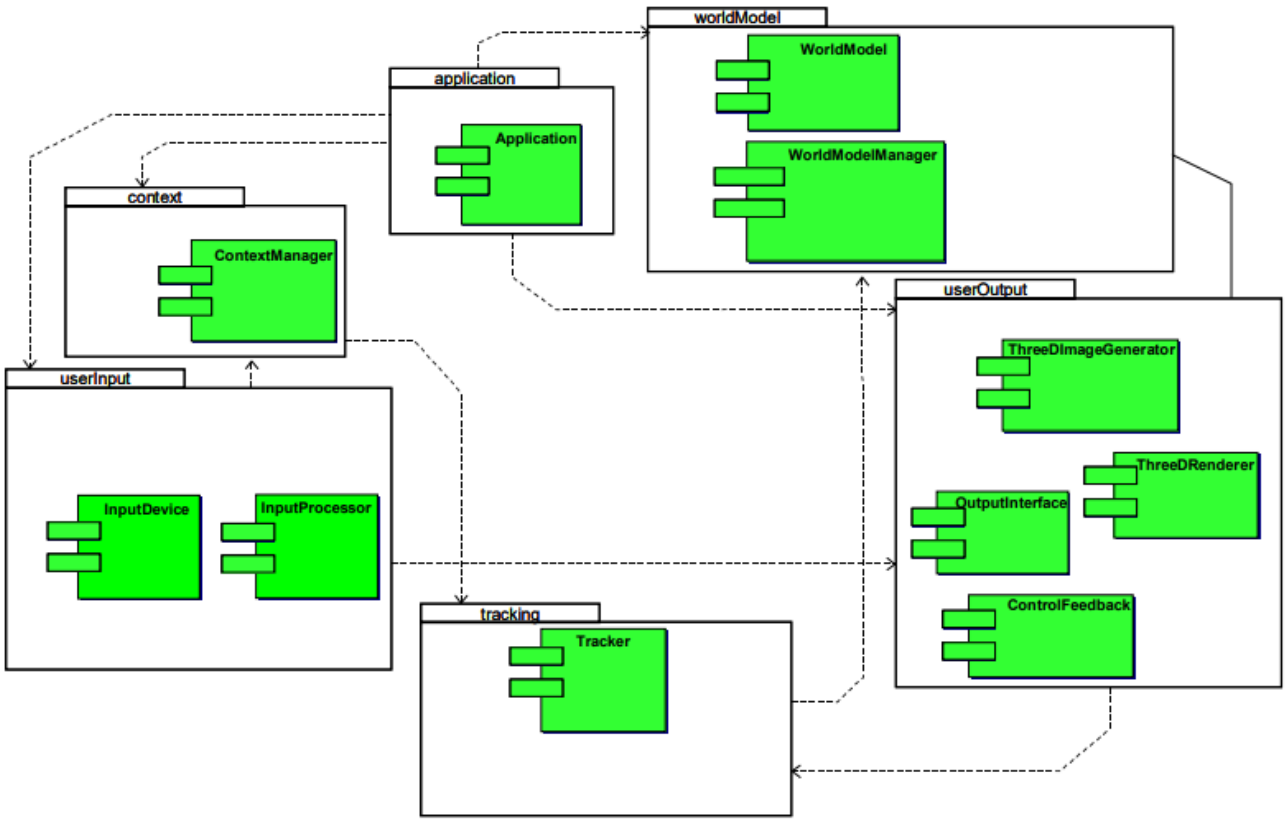

Figura 16 - Diagramma delle classi generico [13]

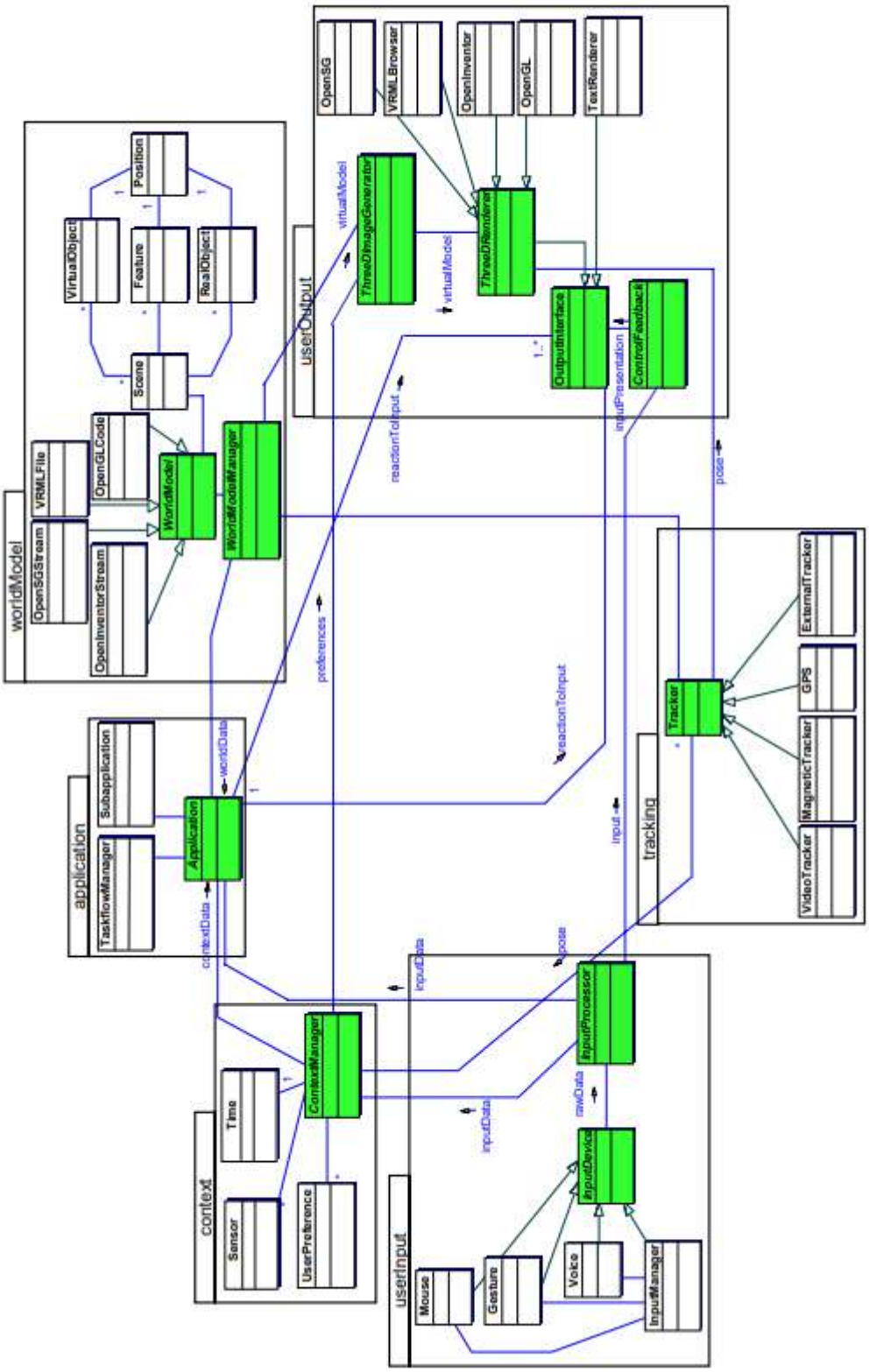

Figura 17 - Diagramma delle classi architettura Realtà Aumentata [13]

5.3 Sottosistema di applicazione

Il sottosistema dell'applicazione è composto da:

- codice specifico dell'applicazione,
- dati di configurazione,
- dati di contenuto.

In questo diagramma è possibile quindi vedere (figura 18) come sono collegate le varie funzionalità fornite dall'utente finale ed è responsabile del bootstrap[24].

L'applicazione dell'utente finale può essere, in linea generale, costituita da molte altre applicazioni secondarie oltre che da ulteriori componenti che forniscono poi anche le varie funzionalità per l'utente.

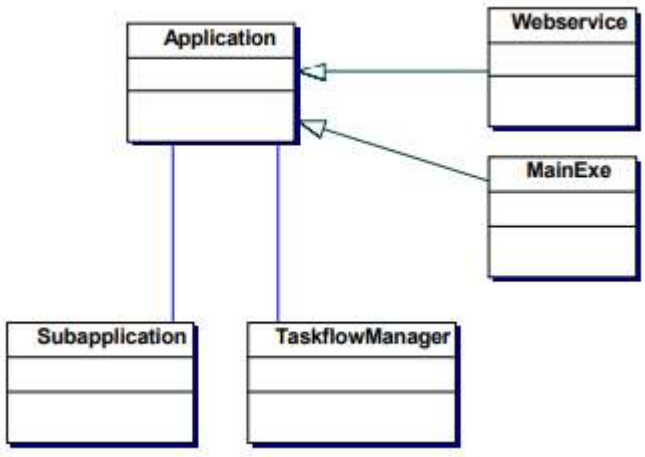

Figura 18 - Sottosistema dell'applicazione

Questo sottosistema mostrato nel precedete diagramma non è specifico per la realtà aumentata, ma comunica con altri sottosistemi, in particolare i sottosistemi di tracciamento e rendering.

[24] Bootstrap è una raccolta di strumenti liberi per la creazione di siti e applicazioni per il Web. Essa contiene modelli di progettazione basati su HTML e CSS, sia per la tipografia, che per le varie componenti dell'interfaccia, come moduli, pulsanti e navigazione, così come alcune estensioni opzionali di JavaScript.

5.3 Sottosistema di tracciamento

Il tracciamento è uno dei sottosistemi di fondamentale importanza per quanto riguarda la Realtà Aumentata, il quale può essere ottenuto attraverso l'utilizzo di diverse tecniche.

I sistemi di tracciamento più gettonati sono:

- quelli basati sul riconoscimento video,
- mediante l'utilizzo di traker magnetici o inerziali,
- geolocalizzazione tramite GPS,
- tracker totalmente esterni dall'ambiente,
- combinazioni, che generano dei tracker ibridi.

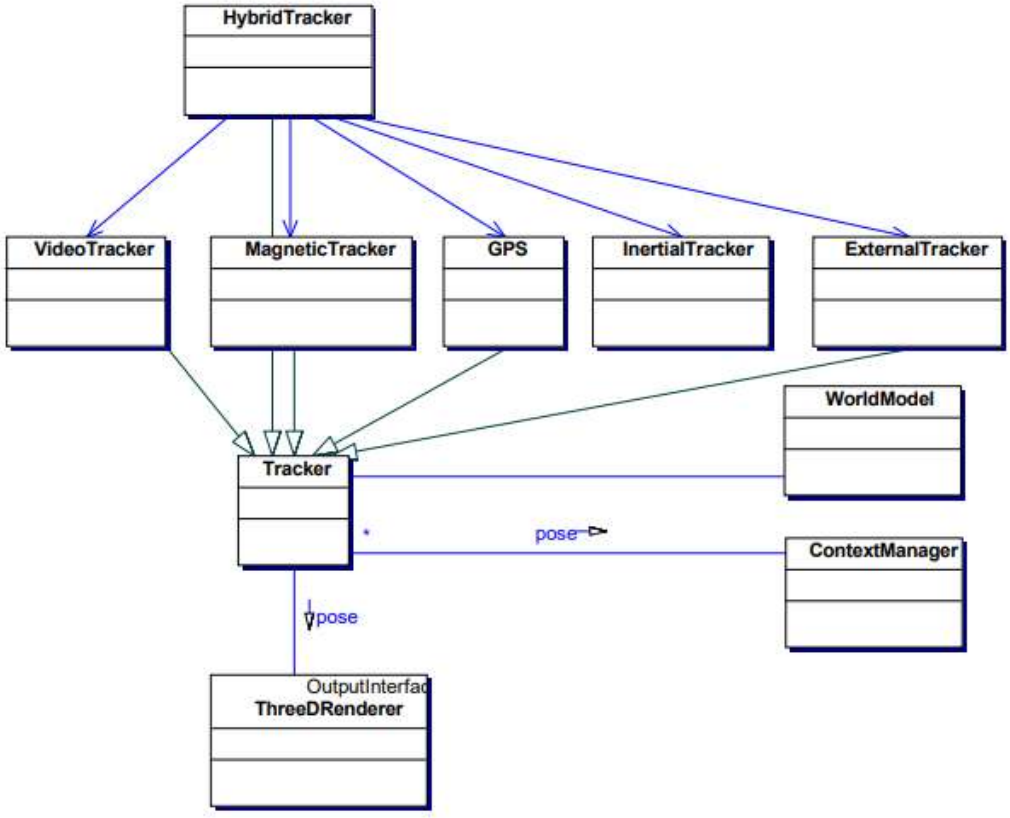

Figura 19 - Sottosistema di riferimento del modello tracking [13]

Come è possibile vedere nella figura precedente (figura 19), oltre alle possibili tecniche di tracciamento, sono raffigurate tutte le possibili funzionalità che i vari tracker possono utilizzare, per ottenere informazioni sull'ambiente in cui

stanno lavorando, come il "World Model", e poi il risultato del processo di tracciamento, la posa, verrà inoltrato sia al componente "ThreeDRendering", il quale aggiorna la visualizzazione e anche al componente "Context Manager" per l'utilizzo in altri sottosistemi.

5.4 Sottosistema per l'Input dell'utente

In questo caso vengono prese in considerazione come input, tutto quello che l'utente fa in modo consapevole per controllare il sistema, ma non è da considerarsi come input il sottosistema di tracciamento.

Il tracciamento della posizione fatto dall'utente diventa un input solamente dopo che è esso verrà interpretato come "un gesto".

In linea generale l'input dell'utente si ottiene mediante:

- dispositivi di input, come il mouse, la tastiera, etc.,
- gesti o voce,
- combinazioni di questi, attraverso un componente che consente la gestione degli input.

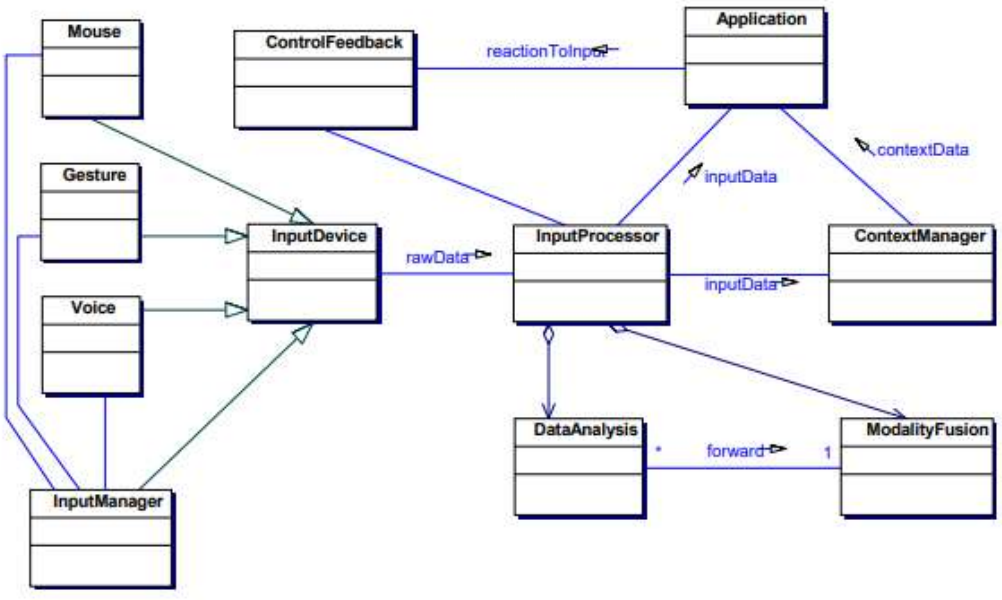

Figura 20 - Sottosistema degli input dell'utente [13]

Come mostrato in figura precedente (figura 20), i dati, che vengono considerati come input, sono inoltrati ad un processore di input "Input Processor", il quale potrà utilizzare a sua volta ulteriori funzionalità, quali:

- l'analisi dei dati, interpretati poi dal componente "Modality Fusion",
- inoltro dell'input all'applicazione,
- inoltro dell'input al gestore del contesto "Context Manager", per un'ulteriore elaborazione.

Il componente, che sta sotto il nome di "Control feedback", visualizza l'input ricevuto dall'applicazione, come ad esempio il click di un bottone sulla tastiera, il quale verrà riconosciuto e potrà essere inoltrato al sistema di output dell'utente.

5.5 Sottosistema per l'output dell'utente

Il sottosistema di output utente è quella che viene considerata l'interfaccia tra l'uomo e la macchina, il quale può essere ottenuto attraverso diversi canali, come:

- grafica 3D,
- testo,
- voce,
- suono.

Nell'ambito della Realtà Aumentata, ovviamente la presentazione grafica, in particolare quella tridimensionale, è quella considerata la più importante, pertanto, come è possibile notare nel diagramma sottostante (figura 21), la fonte di questo sistema è il componente "ThreeDRenderer", ossia un'interfaccia di output che disegna i vari oggetti virtuali che permetteranno l'aumento

della percezione da parte dell'utente, garantendo la corretta distanza, posizione e orientamento.

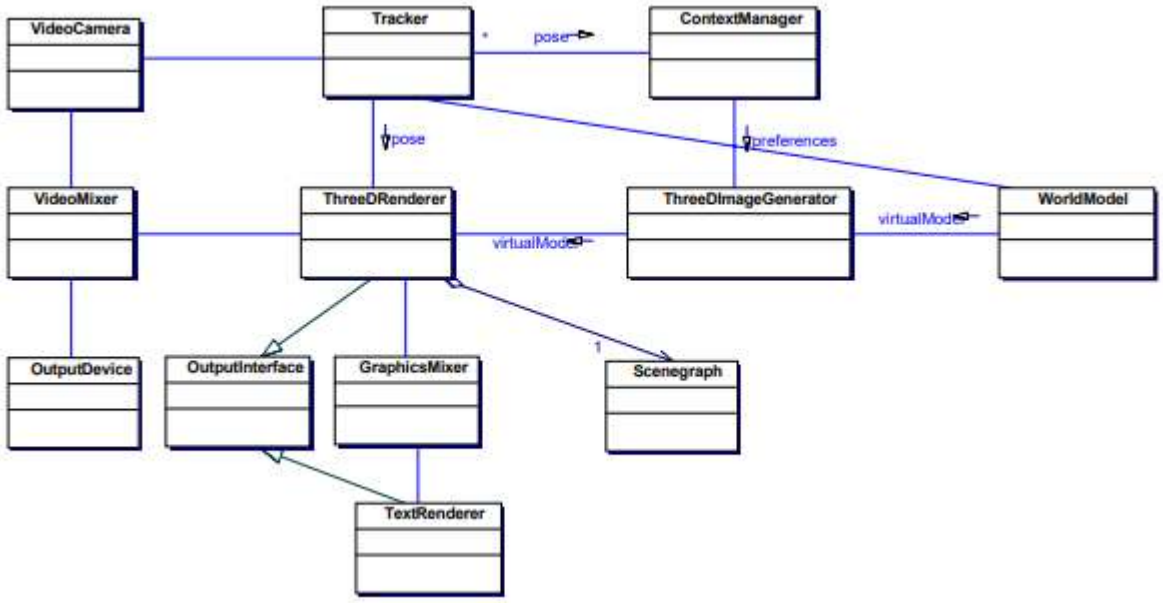

Figura 21 - Sottosistema per gli output dell'utente [13]

Sono inoltre presenti ulteriori componenti, a partire dal "ThreeDImageGenerator" che crea gli oggetti virtuali, per poi adattare i modelli del componente "WorldModel" alle informazioni del componente "Context Manager".

Per la funzionalità di rendering nella versione tridimensionale di solito viene utilizzato un componente "SceneGraph", mentre se vanno disegnati oggetti in due dimensioni viene utilizzato il "TextRenderer".

Il componente "GraphicsMixer" infine integra i dati 3D e 2D in un'unica raffigurazione.

5.6 Sottosistema del contesto

La modellazione del contesto dell'utente dipende dell'applicazione in questione, poiché vi saranno dei componenti diversi, dove ognuno necessario per i diversi tipi di informazioni.

Pertanto, il sottosistema del contesto, in linea generale, può essere visto come un sistema astratto con un componente "ContextManager", che raccoglie diversi tipi di informazioni sul contesto, come:

- preferenze dell'utente,
- dati del sensore e dell'ora,
- dati di tracciamento,
- informazioni sulle risorse,
- conoscenza del dominio,
- conoscenza dello stato attuale dell'utente.

Questo componente memorizza queste informazioni e le rende accessibili ad altri componenti, tra cui il "ContextProcessors" che può leggere le informazioni di contesto, elaborarle e generare nuove informazioni di contesto.

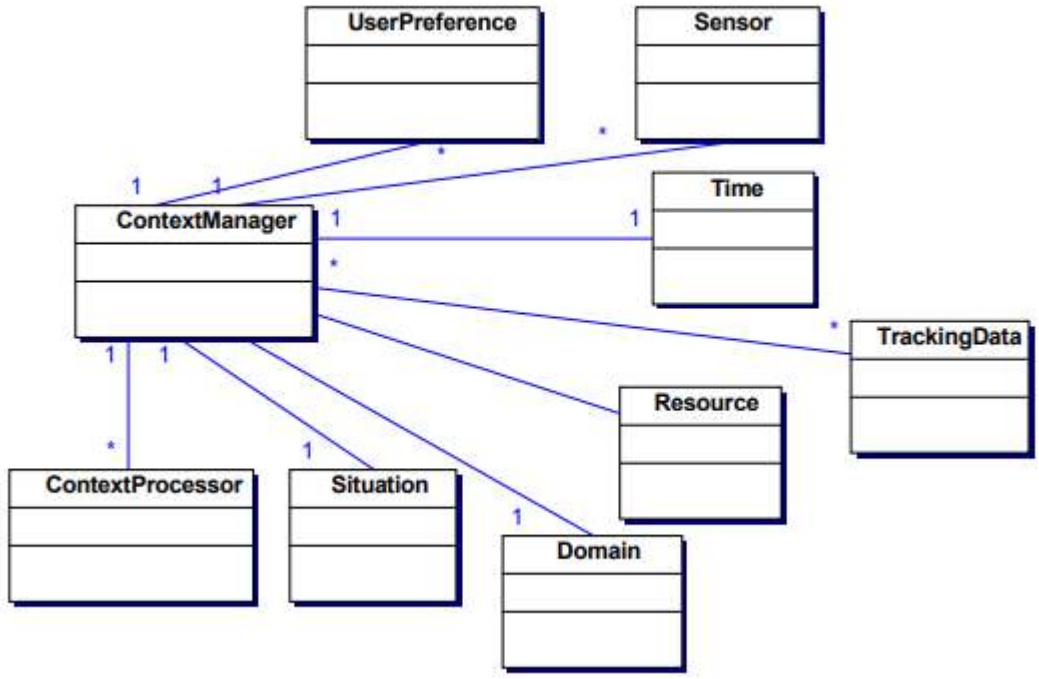

Figura 22 - Sottosistema del contesto

5.7 Sottosistema del World Model

L'ultimo sottosistema che può essere raffigurato è quello del World Model, il quale memorizza e fornisce le informazioni sul mondo che si trova intorno all'utente.

Questo può essere rappresentato, come mostrato in figura 23, in svariati formati, ossia attraverso:

- un modello VRML,
- con codice OpenGL,
- un flusso OpenInventor.

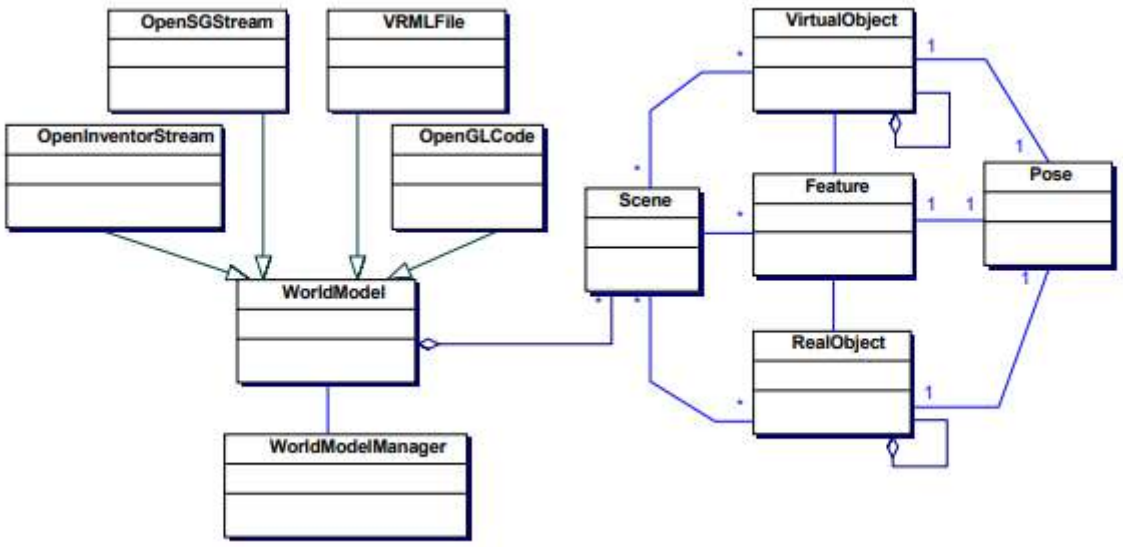

Figura 23 - Sottosistema World Model [13]

Durante la fase di esecuzione, il componente "World Model manager" controlla l'accesso al world model.

Un modello tratto dal mondo reale è composto da:

- scene,
- una raccolta di oggetti virtuali,
- informazioni sulle funzionalità per i tracker,
- rappresentazioni di oggetti del mondo reale.

Capitolo 6: Rendering

Il Rendering, nelle applicazioni in generale di computer grafica, ma anche nel campo della realtà aumentata è una fase dello sviluppo del progetto di fondamentale importanza, soprattutto per l'impatto estetico ed accattivante di quello che gli sviluppatori stanno creando.

Quindi indica un'operazione capace di riprodurre una raffigurazione di un oggetto, di un personaggio o di una struttura cercando di raggiungere una qualità accettabile, quanto più paragonabile e simile al mondo reale.

Con questa si vuole descrivere il processo di creazione di un'immagine partendo da una descrizione matematica di una scena tridimensionale, nel quale vengono interpretati degli algoritmi che ne definiranno il colore e le sfumature di ogni punto della creazione digitale.

Pertanto, può essere considerato uno dei passaggi temi più importanti del lavoro grafico ed è sempre in relazione con tutte le altre fasi, che fornirà l'ultimo stadio al modello e all'animazione finale.

6.1 Blender

Tra i software leader nel settore abbiamo Blender, prodotto open source ed è principalmente utilizzato per la modellazione, la creazione di texture, di animazioni e soprattutto di rendering.

Figura 24 - Blender Logo

È stato sviluppato dalla Neo Geo consente la realizzazione di modelli e applicazioni tridimensionali e fornisce la possibilità, di importare ed esportare i modelli, aiutando in questo modo l'interazione con altri software, oltre che consentire la riusabilità del lavoro svolto, data la grande compatibilità con altre piattaforme.

Punto di forza del software è quello di permettere la gestione visiva di elementi riuscendo a simulare anche lo stato liquido e gassoso, oltre le collisioni e la gravità.

Queste sono principalmente le caratteristiche che rendono un programma molto versatile e molto usato data la sua grande flessibilità, garantendo inoltre la possibilità non solo di modellare, ma di creare da zero le animazioni, rendendo Blender un programma "completo".

Come è possibile vedere nella figura sottostante (figura 25), Blender fornisce una interfaccia in cui sono raggruppate molte e svariate funzionalità, sono diversi menù che possono essere personalizzati secondo le volontà dello sviluppatore stesso.

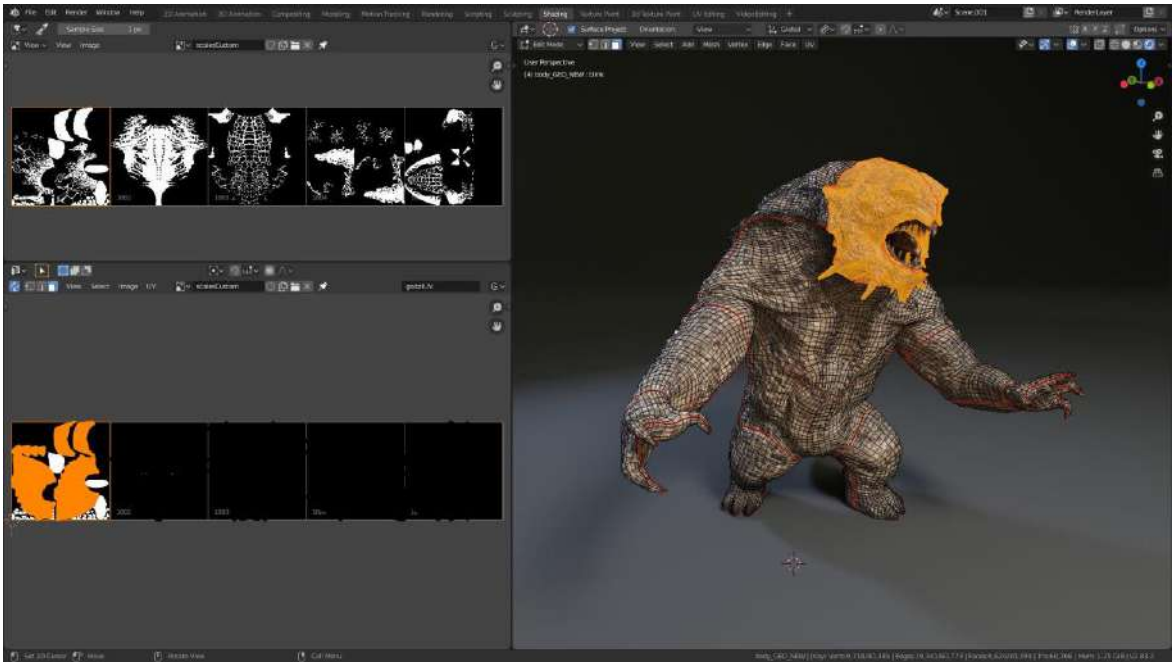

Figura 25 - Esempio di animazione in Blender [14]

Questa interfaccia permette attraverso la barra spaziatrice nella finestra principale, di far comparire un menù che consente di ricercare e di individuare quindi in modo rapido i comandi di cui si ha bisogno.

L'interfaccia, come accennato prima, è totalmente personalizzabile ed è possibile suddividerla in diverse sezioni, teoricamente illimitate, ma in pratica 4 sono sufficienti, così da avere diverse angolazioni dello stesso oggetto a cui si sta lavorando.

I comandi possono essere effettuati tramite le "scorciature da tastiera", ovvero gli Shortcut, il che velocizza notevolmente lo sviluppo del progetto.

Sarà possibile selezionare il tipo di elemento/oggetto base, come un cubo, una sfera o ancora alcuni tipi di curve, da cui partire per poi effettuare dei lavori più complessi, e passare dalla selezione di un oggetto all'altro mediante l'utilizzo del tasto destro del mouse.

Vi sono due principali modalità per sviluppare:

- Object mode, che viene utilizzata per manipolare i singoli oggetti, muovere all'interno dello spazio, scalare o ruotare gli elementi.

- Edit mode, altra modalità utilizzata per modificare gli elementi di un oggetto, permettendo di operare su facce, vertici e spigoli.

Il sistema di fisica è possibile impostare diversi parametri per ottenere l'effetto desiderato, ed è possibile selezionare un sistema metrico per visualizzare la lunghezza dei singoli lati o l'ampiezza degli angoli.

Nel caso dei liquidi è possibile gestire le collisioni e l'erogazione, oltre a specifiche caratteristiche a seconda del fluido che si vuole emulare, operazione però che risulta tra le più pesanti da calcolare e riprodurre.

Da annoverare anche le Action, che sono le azioni compiute da un oggetto, le quali sono modificabili e soprattutto riassegnabili anche ad altri oggetti.

6.2 Cinema 4D

Cinema 4D è un software 3D sviluppato dalla società tedesca Maxon.

Figura 26 - Cinema 4D Logo

Questo software è uno dei migliori che vengono utilizzati oggi soprattutto quando si tratta di lavori molti complessi e che necessitano di essere gestiti sotto diversi punti di vista, discorso che può essere affrontato anche in diversi discorsi.

In questo caso la differenza non è fatta dal motore di lavoro ma da chi lo utilizza perché sono gli utenti stessi a decidere la tipologia di impostazione da dare al progetto.

Il suo punto di forza di questo software è la possibilità di scaricare numerosi plugin che permettono di incrementare le potenzialità dello stesso, come ad esempio:

- Vray che è un plugin usato anche per Rhinoceros e che serve come motore di rendering,
- MOCCA per il mapping dei personaggi animati,
- Dynamics usato per la gestione dei corpi rigidi.

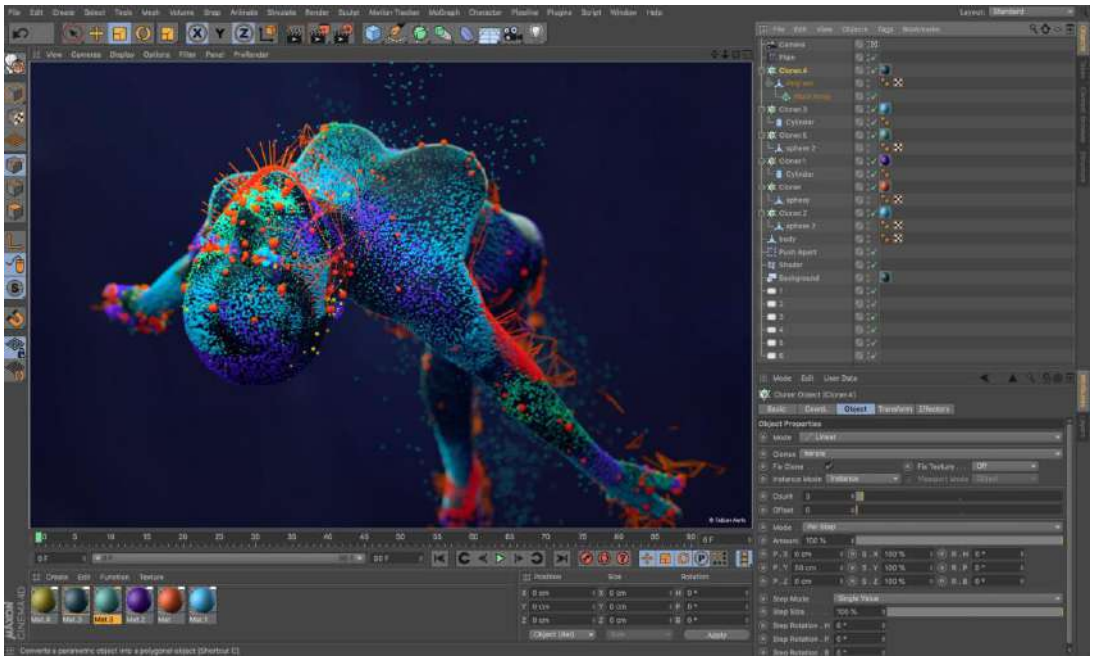

Figura 27 - Interfaccia Cinema 4D [15]

Come mostrato nella figura precedente (figura 27), l'interfaccia si presenta inizialmente è molto simile a quella di Blender, in alto si trovano le varie mo-

dalità di utilizzo, tra cui le simulazioni, lo sculpting, motion tracker ed altri tool per il setting di altri parametri di base.

Nella parte sinistra invece si trovano i node editor, presenti anche in Blender, mentre in basso una timeline che serve a monitorare le animazioni, mentre nella parte destra sono presenti ulteriori parametri che riguardano i livelli e le loro proprietà.

Anche questo software, come nel caso di Blender, presenta le cosiddette "scorciatoie da tastiera" in modo da velocizzare il lavoro.

Vi sono quindi diversi shortcut, dove tra quelli più importanti vengono ricordati più facilmente quelli che si trovano nella colonna degli editor e degli shading, questi suggerimenti sono solo degli aiuti, ma nulla impedisce il singolo utente di andare a cercesi il comando manualmente.

È uno strumento di lavoro molto potente per un principiante, nel vero senso del termine, potrebbe risultare un approccio brusco, proprio per la quantità di comandi presenti e per la gestione non molto intuitiva delle varie scene.

Fortunatamente si possono trovare in rete diverse risorse che permettono allo sviluppatore di conoscere e capire meglio da più punti di vista il software.

Una volta che la scena viene completata, viene effettuate anche la renderizzazione, nel dettaglio:

- vengono calcolate le ombre e le luci,
- i riflessi e le reazioni dei materiali alla luce.

Il file viene salvato come filmato o come immagine statica, il quale processo può richiedere pochi secondi, per scene semplici, oppure giorni, per scene più complesse.

6.3 Lumion

Lumion è anch'esso un programma per l'elaborazione tridimensionale, che consente il rendering di computer grafica 3D e di animazioni.

Principalmente si occupa dello sviluppo di elementi che riguardano elementi di architettura ed inerenti all'edilizia, oltre che al design in generale, usato anche per la creazione di paesaggi.

Figura 28 - Lumion Logo

A differenza di altri software, Lumion utilizza una GPU basata su HLSL[25].

L'interfaccia personalizzata viene renderizzata utilizzando la GPU e si basa principalmente su una visualizzazione della scena effettuata in tempo reale, ed essendo unica la misura dell'accuratezza, riduce al minimo le richieste dell'utente.

Il software gestisce il rendering in anteprima, in modo quasi del tutto definitivo, il tutto calcolato in base alla scheda grafica su cui si sta progettando, ad una frequenza che va da 1 a 150 frame al secondo.

[25] L'High Level Shader Language o HLSL è un linguaggio sviluppato da Microsoft per la creazione di shader da usare in Direct X, ed è molto simile al linguaggio Cg di NVIDIA. L'HLSL permette di scrivere complessi calcoli grafici che possono essere eseguiti molto velocemente dalla GPU, e rappresenta inoltre il primo passo per una pipeline grafica completamente programmabile. Un linguaggio analogo, il GLSL (OpenGL Shading Language), è presente nelle librerie grafiche OpenGL.

Lumion è disponibile per il sistema operativo Windows, ma può essere eseguito anche su macOS, utilizzando la utility boot camp[26].

Le caratteristiche di base del programma sono la semplicità di controllo e gestione, grazie al rendering effettuato in tempo reale, e dato che l'intero programma viene eseguito su una scheda grafica, ciò lo rende molte volte più veloce dei programmi che utilizzano la tecnologia CPU.

A verifica di questo sono la grande quantità di dettagli che si possono scorgere dai prodotti finali, come mostrato ad esempio nella figura sottostante (figura 29).

Figura 29 - Esempio di un prodotto finale [16]

In questa figura è possibile notare come i fili d'erba, le foglie sugli alberi, che si muovono in base all'intensità del vento, sembrino totalmente realistici.

Consente quindi di poter di aggiungere ambienti, materiali, luci, oggetti, foglie e molti altri effetti.

[26] Boot Camp è una utility integrata nel Mac e ti permette di spostarti tra macOS e Windows.

Il programma funziona su quasi tutte le schede grafiche che soddisfano i requisiti minimi del programma, però i computer da gaming grazie alle loro caratteristiche sulla carta migliori, sono quelli che forniscono migliori prestazioni per le operazioni di rendering.

Il programma si sforza anche di ridurre la necessità di ulteriori regolazioni negli editor grafici, quindi sono disponibili molti effetti, come la nitidezza, Sagome 3D, inserimento di bitmap, stili artistici e molti altri effetti.

Figura 30 - Interfaccia di lavoro Lumion [16]

Come mostrato nella precedente immagine (figura 30), l'interfaccia del software presenta la possibilità attraverso diversi menù di poter gestire l'ambientazione a cui si sta lavorando.

Il menu è suddiviso in delle categorie, ciascuna con il proprio sottomenu specifico, come ad esempio, nel menu del controllo meteo possono essere gestiti:

- il movimento del sole,
- la quantità di luce,
- la quantità di nuvole e il loro tipo.

Come impostazione di default, Lumion funziona sulla base delle versioni standard, senza la necessità di dover applicare alcun aggiornamento per l'esecuzione del programma.

Vi sono numerosi vantaggi che rendono questo software un dei migliori:

- può essere facilmente usato anche da sviluppatori non esperti, poiché consente a chiunque di creare filmati ed immagini senza la necessità di dover seguire corsi di formazione, trattandosi di un prodotto funzionale, intuitivo e veloce,
- compatibilità con tutti i programmi di software di progettazione 3D, quali Revit, SketchUp e ArchiCAD,
- velocità di applicazione, poiché è possibile lavorare e modificare tutto in tempo reale, consentendo un grande risparmio in termini di tempo,
- vasta libreria che consente la presenza di molti effetti visivi molto realistici, che possono essere utilizzati dal software.

Capitolo 7: Motore di gioco

Il motore di gioco detto anche motore grafico è la base software di un videogioco o di una applicazione di realtà aumentata, a patto di avere una grafica gestita in tempo reale.

Questo fornisce tutte le tecnologie che sono alla base di un progetto di una applicazione, le quali permettono di semplificarne lo sviluppo oppure di permettere al gioco/applicazione stesso/a di funzionare ad esempio diversi sistemi operativi o dispositivi mobili.

Le funzionalità, che in generale vengono fornite da un motore di gioco, prevedono:

- la gestione del rendering per le grafiche sia tridimensionali che bidimensionali,
- un rilevatore di collisioni detto motore fisico,
- gestione del suono e delle animazioni,
- scripting,
- intelligenza artificiale.

7.1 Unity

Unity è forse il motore di gioco ad oggi più completo ed usato nel mondo dello sviluppo di applicazioni ti tutte le tipologie, a partire dai videogiochi da console o per dispositivi mobili, per arrivare alle applicazioni di realtà aumentata.

Figura 31 - Unity Logo

È una multipiattaforma che consente lo sviluppo di videogiochi e altri contenuti interattivi, come le visualizzazioni di architetture o animazioni tridimensionali in tempo reale.

Questo ambiente di sviluppo gira sia su sistemi operativi Windows, macOS e Linux, così come le applicazioni che verranno sviluppate da esso, potranno essere compatibili, e quindi essere eseguite, su diverse console e sistemi operativi come:

- Xbox 360, PlayStation 3, PlayStation Vita, Wii, PlayStation 4, Xbox One, Wii U e Switch,
- iPad, iPhone, Android, Windows Mobile,
- Windows, macOS, Linux.

Unity presenta un editor, il quale può essere utilizzato sia un per la progettazione dei contenuti, ma presenta anche un motore di gioco attraverso il quale è possibile gestire l'esecuzione del prodotto finale.

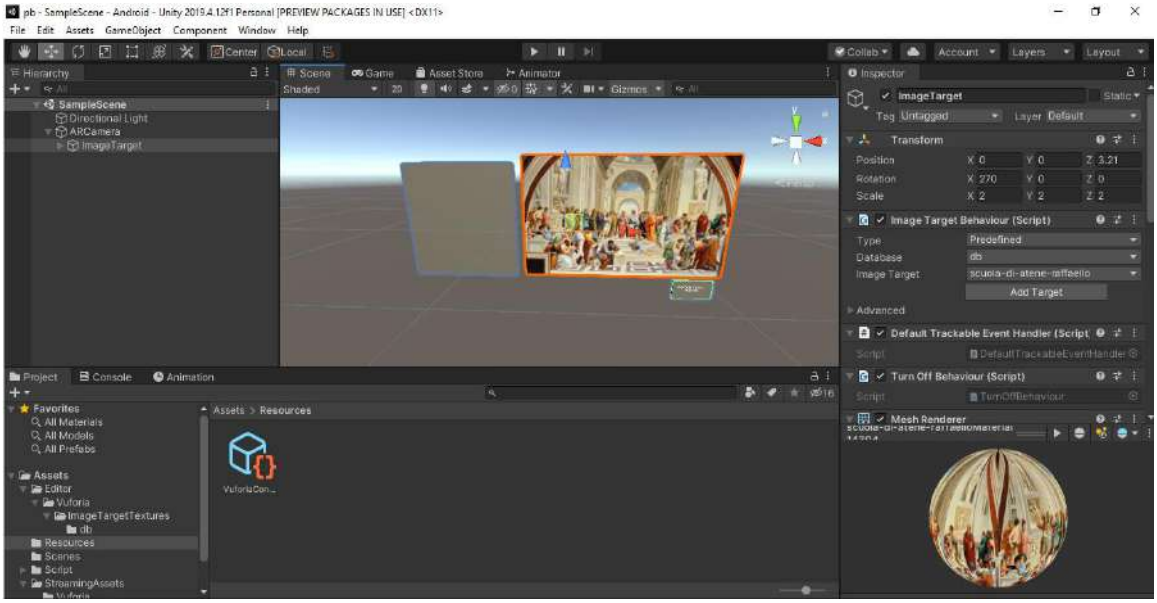

Figura 32 - Interfaccia di lavoro Unity

In Unity è inoltre possibile utilizzare un ambiente grafico del tutto integrato come metodo principale di sviluppo per le applicazioni.

Con la versione 2019 oltre alle innumerevoli modifiche che sono state apportate al motore grafico in modo da renderlo quanto più professionale possibile oltre che al passo con i tempi, grazie all'inserimento di nuove funzioni, dove la principale novità è la compatibilità con il visore per la realtà virtuale presente nell'apposito kit di Nintendo Labo[27].

[27] Nintendo Labo è un concetto di giocattoli che prendono vita sviluppato da Nintendo e rilasciato nell'aprile 2018. Labo è composto da due parti, una parte è un gioco e una parte è costituita da più fogli di cartone.

7.2 Interfaccia

L'area di lavoro di Unity è suddivisa in finestre come è possibile vedere nella seguente figura:

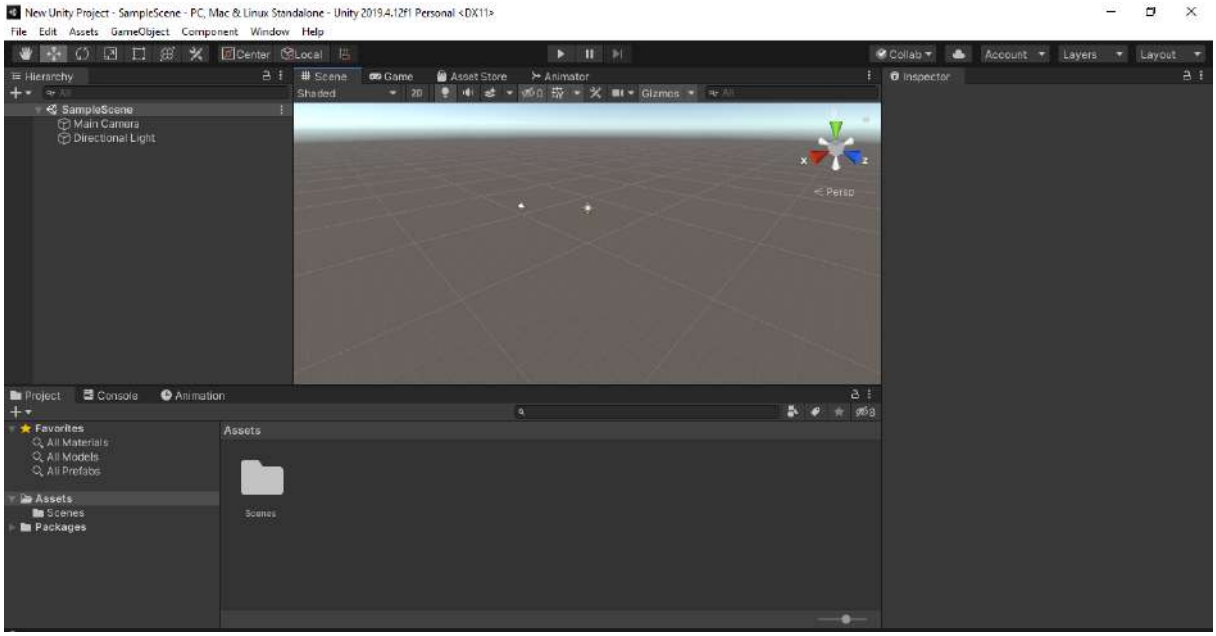

Figura 33 - Interfaccia base di un progetto Unity

Come è possibile notare l'interfaccia iniziale viene suddivisa nelle seguenti sezioni:

- "Hierarchy", finestra presente di default nella parte sinistra dell'interfaccia, attraverso la quale vengono elencati tutti gli oggetti presenti nella scena, o che verranno poi inseriti in essa, e grazie ad essa è possibile ordinare gli oggetti e selezionarli. Da annoverare il fatto che un oggetto stesso può contenere ulteriori oggetti all'interno di esso;
- "Scene", posizionata nella parte centrale, indica la finestra all'interno della quale è possibile lavorare sulla visuale, della scena appunto, su cui si sta lavorando attualmente. In questa finestra è possibile, pertanto, sistemare, spostare, ruotare, ridimensionare e scalare i vari oggetti presenti nello spazio creandone l'ambiente. È impostata di default sulla vi-

sta 3D, ma tramite un bottone in alto indicato con il simbolo "2D" si può selezionare la scena con una visualizzazione bidimensionale nell'eventualità in cui si voglia realizzare un gioco a sole due dimensioni;

- "Game", è la finestra che permette la simulazione del gioco, e tutto quello che viene visualizzato dalla camera viene reso disponibile e visibile in questa finestra, questa finestra viene usata generalmente per la fase di testing di una scena;
- "Inspector" consente la visualizzazione di tutte le proprietà e dei componenti del game object che è stato selezionato;
- "Project", in questa finestra è possibile visualizzare le risorse del progetto e gestirne l'organizzazione;
- "Console", infine, è la finestra in cui vengono visualizzati tutti i messaggi contenenti gli eventuali errori/warnings di compilazione.

7.3 Creazione progetto

Quando si crea un nuovo progetto con Unity si deve prendere in considerazione tutte le risorse di cui l'applicazione necessita, ed all'interno di esso possono essere definite ulteriori scene, scripts e prefabs[28].

All' avvio del software (figura 34), comparirà una schermata in cui sarà possibile visionare tutti i progetti esistenti, oppure crearne dei nuovi, e nel caso si volesse creare un nuovo progetto, basterà cliccare sul bottone "New" per poi vedere la nuova schermata che apparirà (figura 35), in cui sarà possibile sce-

[28] Il sistema prefabs (prefabbricato) di Unity consente di creare, configurare e archiviare un file di tipo GameObject completo di tutti i suoi componenti, valori di proprietà e GameObject figlio come risorsa riutilizzabile. L' asset prefabs funge da modello da cui è possibile creare nuove istanze prefabbricare nella scena.

gliere il tipo di progetto, tra quelli proposti nella sezione a sinistra, che si vuole create.

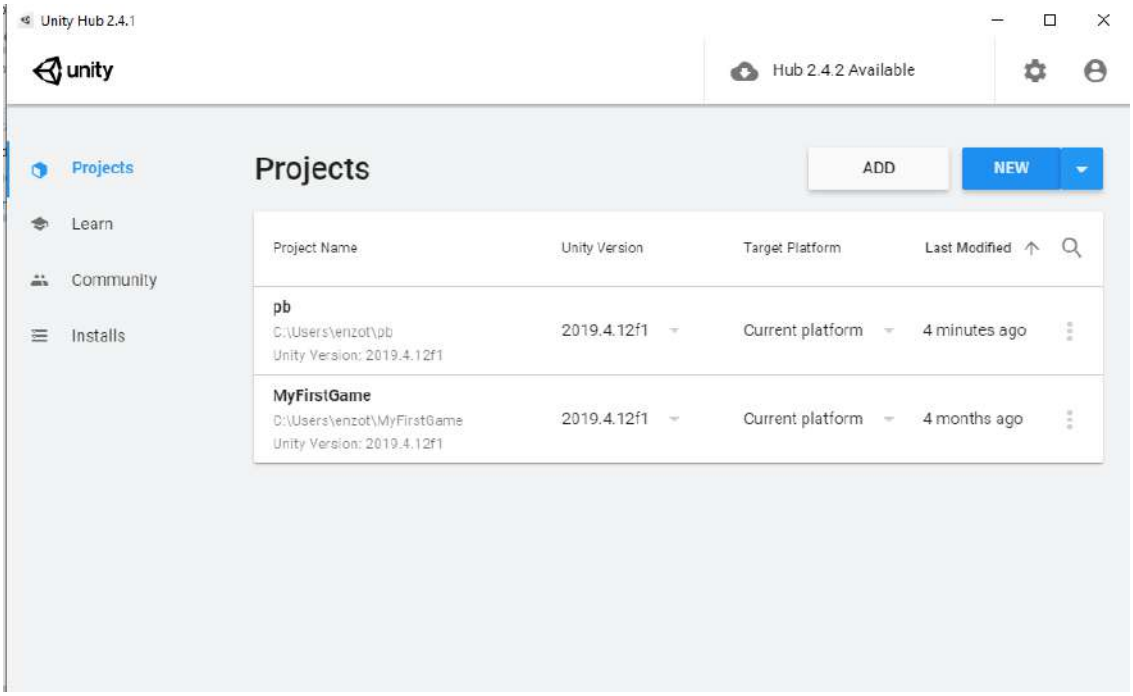

Figura 34 - Creazione nuovo progetto Unity (pt.1)

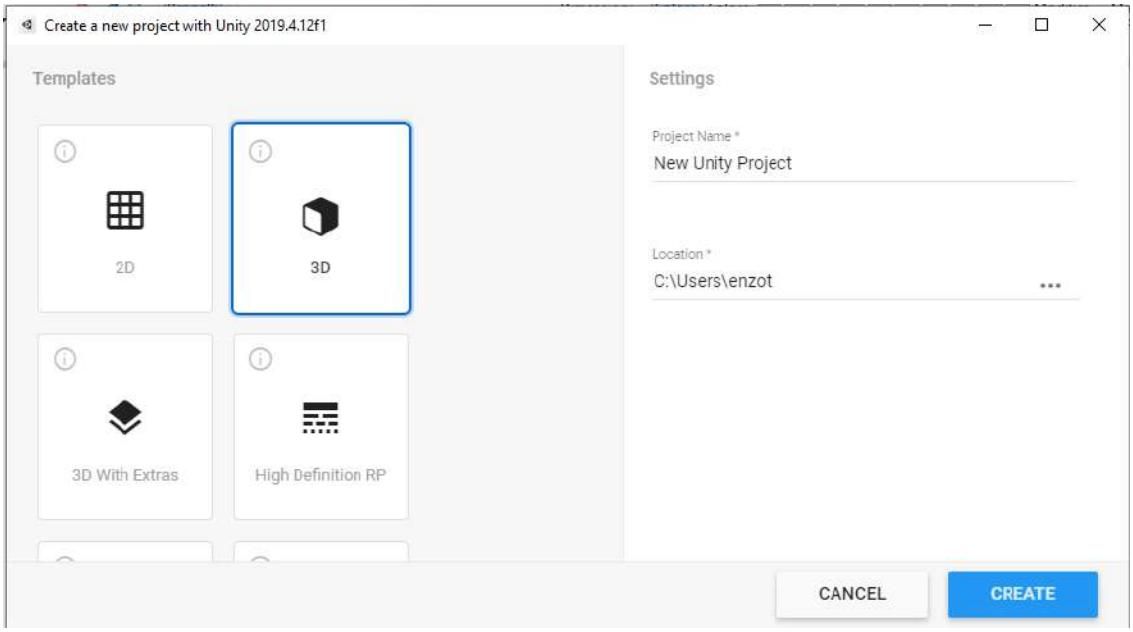

Figura 35 - Creazione nuovo progetto Unity (pt.2)

Una volta configurati tutti i settaggi iniziali, comparirà l'interfaccia mostrata nel paragrafo precedente (7.2 Interfaccia), attraverso la quale sarà possibile lavorare allo sviluppo dell'applicazione, nel nostro caso di realtà aumentata.

7.4 Componenti del progetto

La scena, come quella ad esempio mostrata nella figura sottostante (figura 36), a cui si intende lavorare può essere definita come un ambiente "virtuale" totalmente isolato, in cui in essa vengono definiti tutti gli oggetti ed i marker che ne comporranno il livello.

Nel momento in cui essa verrà caricata, l'engine (motore di gioco) allocherà tutti gli oggetti al suo interno in memoria, e nel caso eventuale in cui si dovesse voler cambiare scena e quindi switchare da una scena all'altra, tutti gli oggetti che erano presenti nella scena precedente verranno automaticamente deallocati.

Ciò nonostante, esistono comunque diverse tecniche che permettono di mantenere i GameObject tra una scena e l'altra.

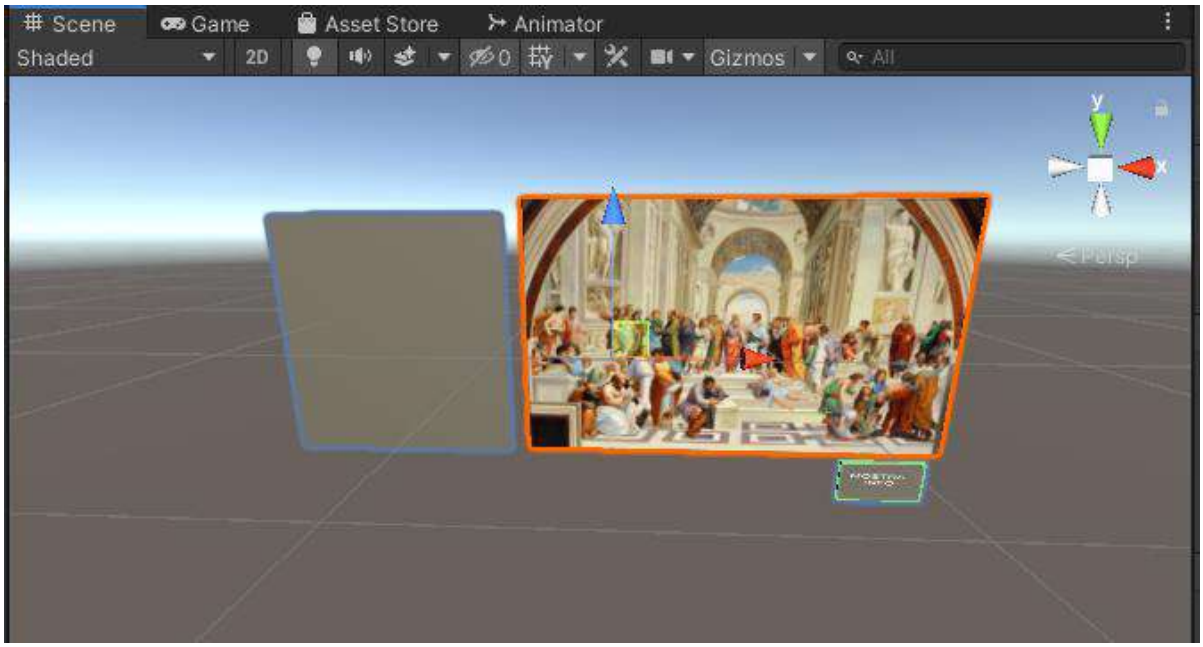

Figura 36 - Scena di un progetto Unity

Il "GameObject" è uno degli elementi principali e fondamentali dell'engine a cui è possibile aggiungere ulteriori componenti che ne specificano i diversi comportamenti e le loro relative proprietà.

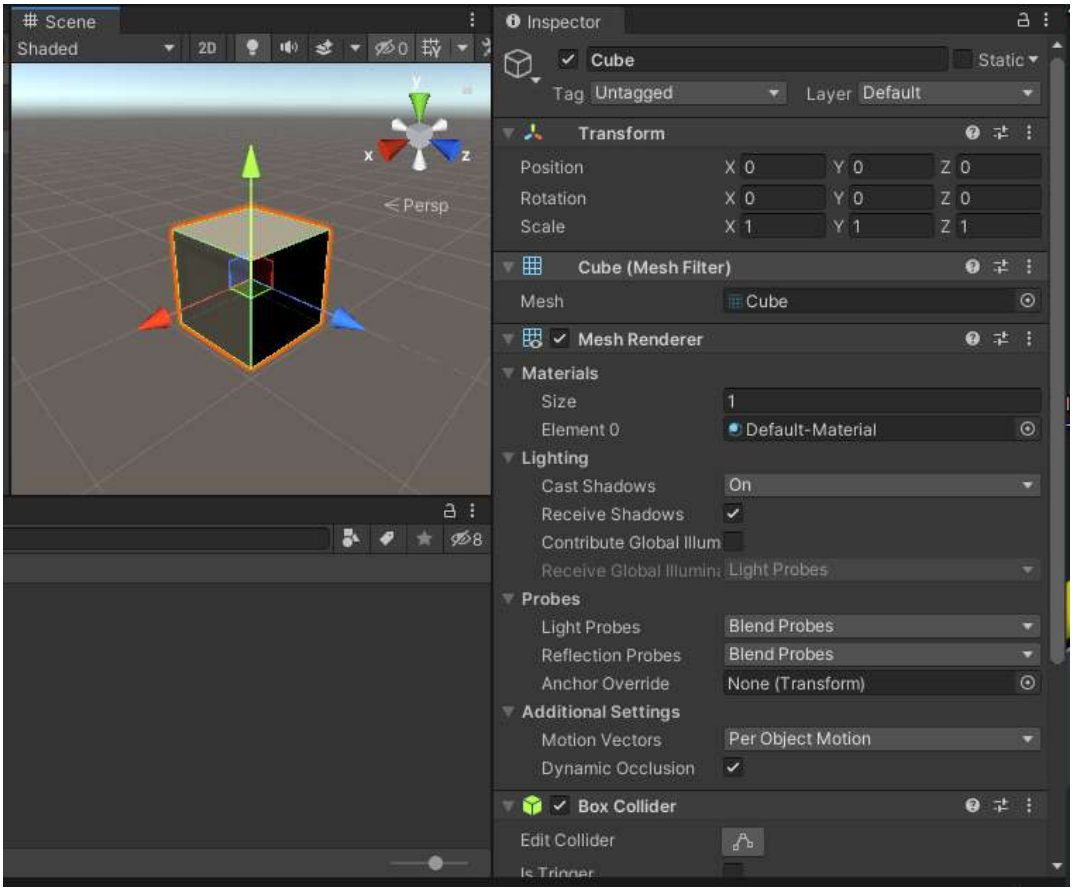

Figura 37 - Esempio di GameObject

Come mostrato in figura 37, ogni GameObject ha una sezione dedicata dove poter modificare le relative proprietà, visibile nella schermata a destra, dove è possibile definire la posizione, rotazione e grandezza dell'oggetto secondo un sistema di coordinate relativo agli assi xyz.

Il "Component", in italiano componente, può essere considerato come un ulteriore oggetto/file che va inserito all'interno di un GameObject in base alle esigenze progettuali, per poter aggiungere delle proprietà specifiche all'oggetto oppure dei comportamenti che il "GameObject" deve avere al verificarsi di un determinato evento.

I più importanti ed utilizzati sono:

- Sorgenti audio;
- Animazioni;

- Scripts.

È possibile, inoltre, inserire in un oggetto uno Script semplicemente trascinando lo script dalla finestra Project alla finestra Hierarchy sull'oggetto desiderato, ed attraverso la sezione dell'Inspector sarà poi è possibile abilitare, disabilitare o eliminare oppure impostare il valore di una variabile all'interno di uno script.

Unity permette il salvataggio di oggetti creati nella scena tramite l'utilizzo dei Prefab, elementi che formano un collegamento tra loro, ed il vantaggio di utilizzare un Prefab si ha nel fatto, non solo di poter salvare all'interno del progetto i vari GameObject più comuni, in modo da poterli riutilizzare all'interno della scena o di altre scene, ma anche nel permettere che una modifica effettuata su di esso si ripercuota su tutte le copie di quell'oggetto, rendendo molto più facile la gestione di tutte le copie.

Capitolo 8: Testing

Al giorno d'oggi sono presenti sul mercato diversi software che consento la creazione e lo sviluppo di applicazioni di realtà aumentata, e come visto nei capitoli precedenti, vi sono diversi tool che consentono di gestire al meglio il lavoro, secondo le preferenze degli sviluppatori.

Vi si può progettare applicazioni su diverse piattaforme, e molte sono le aziende che investono nella produzione e nello studio di nuovi dispositivi e strumenti che possono adempire al meglio gli obiettivi che si prefigge la realtà aumentata, soprattutto in ambito di avanguardia nel lavoro dell'uomo.

Quindi, mentre gli strumenti di sviluppo e l'applicazione della realtà aumentata stanno diventando sempre più abbondanti, allo stesso tempo gli strumenti e le pratiche di test specifici per la realtà aumentata sono ancora in ritardo.

Tuttavia, sebbene la tecnologia, che riguarda la realtà virtuale, esista da diversi anni, rispetto alla realtà aumentata, non esistono delle pratiche comuni che consentono di gestire in modo del tutto automatizzato il testing per queste applicazioni.

Come è possibile osservare nella figura sottostante (figura 38), non vi sono il livello di somiglianza tra le due realtà, quella aumentata e quella virtuale, come si interfacciano, tra l'ambiente reale e quello virtuale.

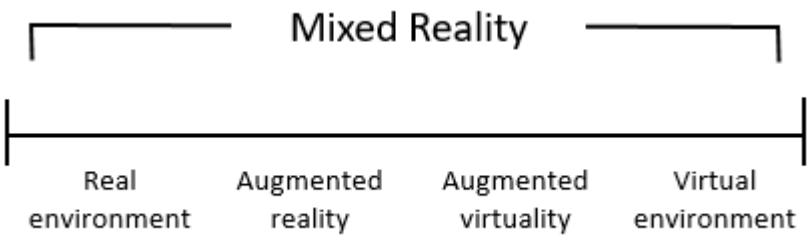

Figura 38 - Differenze tra ambiente reale e virtuale [17]

Sebbene diverse ricerche abbiano identificato un discreto potenziale, quasi nullo nelle pratiche di test per nuove interfacce o software, ciò ovviamente non vuol dire che gli sviluppatori di applicazioni di realtà aumentata, non facciano testing o che esso non sia necessario, e che quindi non le cercheranno o si baseranno su di esse.

La valutazione della qualità e del corretto funzionamento di un'applicazione di realtà aumentata è più complessa di quella di un tipico prodotto software tradizionale, infatti i primissimi sviluppatori di queste applicazioni, hanno riscontrato bug nelle versioni di produzione.

I rapporti con problemi di qualità riscontrati nelle applicazioni includono il mancato posizionamento dei modelli su superfici fisiche lucide, dettagli del modello bassi, contorni del modello innaturali e distinguibili e il mancato rispetto delle dimensioni proporzionali di un modello all'ambiente fisico.

Questi problemi erano spesso comuni in diverse applicazioni dovute principalmente alle condizioni di scarsa illuminazione, dove i modelli galleggiavano o si allontanavano dal punto di ancoraggio originale.

La pianificazione dei test per il rilascio di un'applicazione di realtà aumentata ha molti fattori contestuali ambientali, tra cui l'oggetto virtuale deve riuscire a comportarsi entro dei limiti accettabili dell’ambiente reale.

A seconda di come è strutturata l'applicazione, potrebbero essere necessarie tecnologie cloud che prevedono dei punti di ancoraggio condivisi, il che può anche portare ad avere requisiti di qualità funzionali e non funzionali ovviamente più complessi.

Le soluzioni per testare in modo automatizzato per Web e dispositivi mobili sono esistenti e disponibili, sviluppati sia da comunità open source che da aziende commerciali, ma ciononostante la ricerca per strumenti di automazione dei test specifici per le applicazioni di realtà aumentata non sono facili da trovare.

8.1 Caratteristiche

Un'applicazione di realtà aumentata ben progettata riesce a fondere il mondo virtuale e quello fisico in un'esperienza che facilita la prospettiva, la presenza, l'interazione e l'immersione.

Nel dettaglio queste caratteristiche possono essere descritte nel seguente modo:

- la prospettiva è uno dei fattori fondamentali, necessaria per coinvolgere oggetti virtuali in modo realistico all'interno di ambienti fisici dinamici,
- la presenza è la capacità dell'utente di essere all'interno del proprio ambiente fisico e di partecipare al modello o alla scena virtuale,
- l'interazione che si verifica quando viene manipolato il modello oppure quando cambia in relazione alla prospettiva dell'ambiente reale,
- l'immersione indica la capacità di essere presenti sia fisicamente che virtuali allo stesso tempo, e per scalare AR, l'applicazione deve avere anche la persistenza.

Queste principali caratteristiche consentono ai modelli e alle scene di realtà aumentata di essere sperimentati da altri utenti dell'applicazione, ridimensionando l'esperienza della realtà aumentata, e richiedono agli sviluppatori di andare oltre i tradizionali problemi di test.

Vi sono ulteriori considerazioni sulle caratteristiche che vanno fatte soprattutto riguardanti ad esempio la scalabilità, ovvero in questo caso si fa riferimento al ridimensionamento delle esperienze di realtà aumentata tra più utenti, che richiede la centralizzazione di mappe fisiche e ancoraggi di oggetti virtuali in modo che più utenti possano condividere l'immersione.
ARKit e ARCore hanno entrambi queste capacità.
Altra caratteristica è la persistenza, che gioca un ruolo importante sulla qualità che riguarda gli scenari per un singolo utente, dove vi si posiziona l'oggetto virtuale e giorni dopo ci si aspetta di vederlo incorporato sempre nella stessa posizione fisica.
La persistenza viene considerato è anche un requisito che consente di aumentare la capacità degli utenti di condividere l'esperienza aumentata, infatti diversi casi di test richiedono una quantità di complessità nel numero di tester che si impegnano in uno scenario ed esperienza condivisi, nonché la durata del tempo coperto dal caso di test.

8.2 Contesto ambientale

La funzionalità di un'applicazione di realtà aumentata deve essere testata per due requisiti distinti, occlusione e collisione.

In generale gli utenti delle applicazioni di realtà aumentata hanno bisogno di interagire in modo realistico tra gli oggetti fisici e virtuali, significa che gli oggetti hanno ombre e profondità adeguate e non devono occupare lo stesso spazio allo stesso tempo.

8.2.1 Occlusione

L'occlusione richiede che se un oggetto virtuale è distanziato da oggetti fisici, durante il rendering gli oggetti fisici nascondano o occludano parti o l'oggetto virtuale interamente in modo realistico, questa è difficile perché i motori grafici di realtà aumentata devono rappresentare correttamente il mondo fisico, anche se la relazione con esso cambia a seconda del movimento fisico effettuato.

Il motore grafico deve ricostruire il modello tridimensionale in base alle proprietà del mondo fisico e rendere l'interfaccia utente dinamica in modo appropriato.

È necessario quindi creare dei casi di test che rappresentino i possibili scenari di ambienti fisici in cui verrà eseguita l'applicazione, per poi caricare e coinvolgere gli oggetti virtuali per testare la propensione a fallire l'occlusione o creare un'esperienza che non rappresenti la realtà.

Figura 39 - Esempio di occlusione [18]

L'immagine precedente (figura 39) mostra un esempio di occlusione, applicata tramite dispositivo mobile (tablet) nell'ambiente reale, mentre nelle figure sottostanti (figura 40), possiamo notare la differenza tra una corretta occlusione ed un caso in cui essa fallisce.

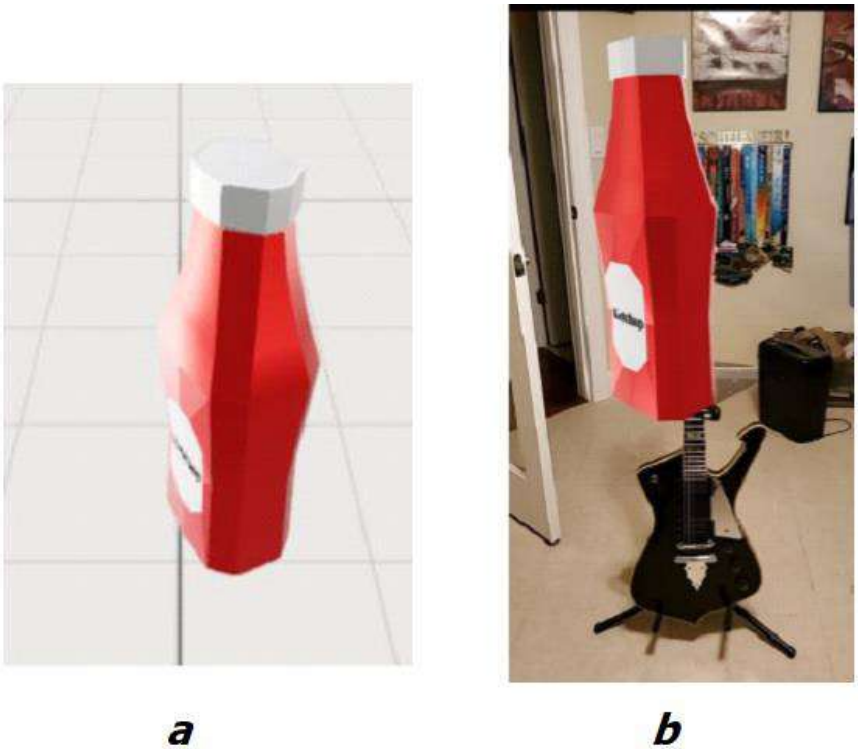

Figura 40 - (a) Esempio di corretta occlusione. (b) Esempio di occlusione fallita [17]

8.2.2 Collisione

Una collisione è il fenomeno che si verifica quando un oggetto virtuale si avvicina a un oggetto fisico e tenta di occupare lo stesso spazio tridimensionale nello stesso momento, infatti molte applicazioni di realtà aumentata consentiranno di manipolare l'oggetto virtuale così come appare nel mondo fisico.

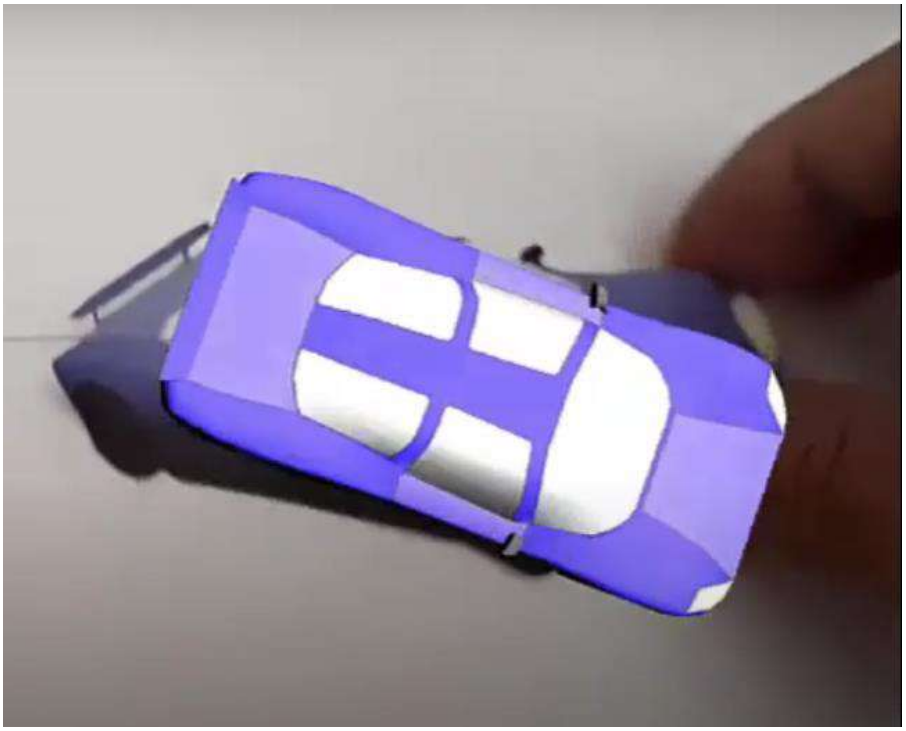

Figura 41 - Esempio di collisione

Come è possibile vedere nell'immagine precedente (figura 41), l'oggetto virtuale, in questo caso l'automobile, non dovrebbe occupare lo stesso spazio degli oggetti fisici, ovvero le dita dell'utente, nonostante la manipolazione, ma il rilevamento delle collisioni deve essere testato in un'implementazione.

I casi di test per prevenire le collisioni dovrebbero includere anche il modo in cui il colore e la luce modificano la capacità di rilevare e prevenire tali collisioni.

8.3 Test di distinzione

Gli ambienti del mondo reale rendono abbastanza difficile la distinzione del gradiente i quali ridurranno l'usabilità delle applicazioni, poiché le applicazioni di realtà aumentata possono perdere la distinzione visiva in base al mondo fisico in cui vengono immerse.

Infatti, coinvolgere gli oggetti virtuali è difficile ed alcuni casi sono facilmente inclini al fallimento soprattutto nelle aree dove la luce è intensa, oppure dove il colore bianco è dominante.

Vi sono anche casi in cui il modello e la scena reale sembrano di colore camaleontico, o quando le superfici sono molto lisce e pertanto presentano la caratteristica di essere riflettenti.

Il test di distinzione, quindi, deve avere più casi di test che alterano l'ambiente fisico a questi tipi di estremi, per comprendere il comportamento dell'applicazione e per presentare potenzialmente all'utente casi di studio che vanno al di fuori dell'interfaccia utente di realtà aumentata, come un'interfaccia basata su menu tradizionale, ma anche nel caso di mancato rendering degli oggetti virtuali in modo tale che siano distinti.

8.4 Test delle prestazioni

Il rendering tridimensionale che viene effettuato durante il funzionamento dell'hardware della fotocamera e l'unione dell'immagine in un'interfaccia utente è dal punto di vista computazionale considerato molto oneroso.

Come capitava nelle prime versioni di ARCore, al momento del rilascio venivano fornite delle note che suggerivano che gli ancoraggi, ovvero i punti ca-

ratteristici che rappresentano la presenza di oggetti virtuali nel mondo reale, dovevano essere scollegati o eliminati per evitare costosi costi della CPU.

Le applicazioni di realtà aumentata sono spesso implementate utilizzando un'architettura distribuita, in cui sono presenti sia dei client locali, ma anche ulteriori servizi ad esempio di tipo cloud.
Pertanto, andranno eseguite delle analisi prestazionali su tutte le risorse principali del progetto, con la consapevolezza che le capacità di calcolo di tali risorse possono essere estese quando molti clienti si impegnano nei servizi.
Molto spesso capita che i dispositivi client stessi risultino avere problemi di prestazioni, come ad esempio l'esaurimento della batteria in seguito a lunghi tempi di funzionamento.
I tradizionali test delle prestazioni risolvono una buona parte di queste analisi e coloro che utilizzano le risorse cloud dovranno aggiungere la convalida della complessità e del costo delle risorse ridimensionate automaticamente ai piani di test.
Le prestazioni del cloud possono essere ridotte quando ad esempio vi è un traffico di rete elevato verso il data center del cloud, causando quindi del traffico che potrebbe risultare intenso in un'implementazione di un'applicazione AR distribuita.

8.5 Test automatizzati con robot

L'automazione del testing aiuta innanzitutto ad accelerare i test in modo da poter considerare il rilascio di produzione prima e anche in modo da poter trovare quanti più bug possibili in un periodo di tempo più breve.

Analogamente a come le pratiche di test tradizionali non considerano nuove interfacce utente come la realtà aumentata, mancano anche gli strumenti di test in quest'area.

Un caso interessante di studio è quello effettuato dalla “SmartBear”, che ha pensato e progettato un particolare approccio al testing, sfruttando le potenzialità della robotica.

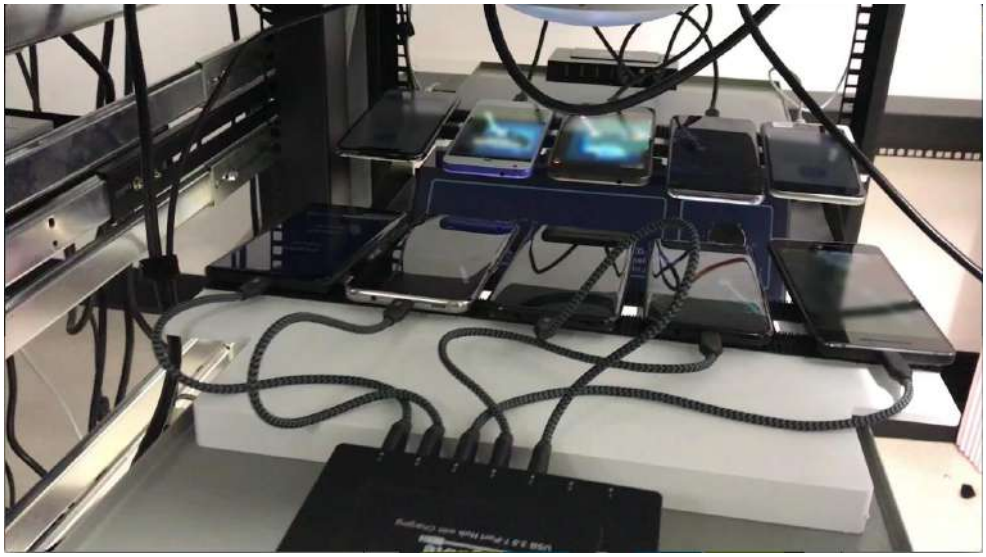

Figura 42 - Test robotizzato [19]

Il sistema prevedeva un laboratorio mobile reale e l’uso quindi della robotica fisica che riposizionano i dispositivi consentendo alle loro telecamere di zoomare e ripristinare adeguatamente per incorporare il modello 3D nel mondo fisico.

Questo studio non descrive però come è stata testata la funzionalità e l'esperienza dell'utente, né in particolare come sono stati testati gli aspetti che prevedono sia la collisione che l'occlusione.

Questo approccio prevedeva di creare una disposizione fisica in cui tutti i dispositivi mobili fossero posizionati su un piano, in modo equidistante tra loro, su un "tavolo" tridimensionale a circa 60 cm dai dispositivi.

Inoltre, il tavolo che veniva utilizzato, prevedeva delle "trame" diverse in modo che la piattaforma di realtà aumentata fosse capace di allineare gli oggetti tridimensionali sovrapposti sopra la superficie del tavolo.

Infine, sono state garantite delle condizioni di illuminazione quanto più uniformi possibile nella parte superiore e intorno al tavolo in modo che le ombre o le aree scure, non presentassero dei problemi di interferenza al processo di test.

Il tutto è stato ottenuto tramite il collegamento di luci LED[29] grandangolari che puntano il tavolo da quattro direzioni creando una coperta leggera e uniforme. [19]

Altro requisito di fondamentale importanza, era la calibrazione dell'applicazione di realtà aumentata, per cui l'automazione deve spostare i dispositivi per un po' di tempo al fine di riuscire a calibrare sia l'input video che l'accelerometro, procedura che va fatta ogni volta che l'applicazione viene reinstallata.

Quindi per risolvere questo problema, i dispositivi mobili sono stati montati su piani oscillanti utilizzato un controller personalizzato per far oscillare i piani tramite chiamate API.

In questo modo lo sviluppatore può avviare e arrestare l'oscillazione direttamente da script di test e avere il pieno controllo dell'intero flusso di test automatizzato.

I test stessi sono test standard gestiti tramite Cloud-Side Appium[30] eseguiti contemporaneamente su tutti i dispositivi mobili sia IOS che Android, in modo

[29] In elettronica il LED (*Light Emitting Diode)* o diodo a emissione di luce è un dispositivo optoelettronico che sfrutta la capacità di alcuni materiali semiconduttori di produrre fotoni attraverso un fenomeno di emissione spontanea quando attraversati da una corrente elettrica.

[30] Appium è uno strumento di automazione open source per l'esecuzione di script e il test di applicazioni native, applicazioni web mobile e applicazioni ibride su Android o iOS utilizzando un webdriver.

da garantire che i risultati dei test siano confrontabili, dove l'output dei test è una semplice informazione di "pass / fail".

Capitolo 9: Strumenti per il testing

Esistono diversi strumenti di sviluppo, progettati per assistere gli sviluppatori nella creazione di applicazioni di realtà aumentata, e per testarne il corretto funzionamento.

9.1 Airtest

AirtestProject è un framework di testing automatizzato creato dalla Netease Games, ed esso è composto da diversi strumenti di cui gli sviluppatori possono usufruire.

Figura 43 - Airtest Project Logo

Quindi principalmente Airtest è un framework di test automatizzato dell'interfaccia utente multipiattaforma basato sull'identificazione di immagini ed oggetti presenti in essa, adatto sia per giochi che per applicazioni di realtà aumentata, e che permettere di avere una compatibilità come molte piattaforme, ovvero supporta Windows, Android e iOS.

AirtestIDE è l'editor di test che permette di automatizzare funzioni integrate per Airtest, che ti consente di scrivere rapidamente e facilmente codice per entrambi questo framework, ed è un potente strumento GUI che può aiutarti a registrare ed eseguire il debug degli script di test.

Come ulteriore strumento abbiamo anche AirLab, anch'essa piattaforma di test automatizzata, ma in questo caso di tipo cloud, utilizzate per macchine reali, attualmente fornisce test di compatibilità con dispositivi mobili, anche di ultima generazione.

9.1.1Caratteristiche

Airtest potendo eseguire l'automazione dei test sia di giochi che di applicazioni di realtà aumentata su quasi tutte le piattaforme, consentendo di essere definito come un software "Cross-platform".

Come requisiti per il sistema ospitante, funziona sia su Windows, su MacOS e Linux, e necessita una versione di Python nella versione minima di 2.7.

Questo progetto è basato sul linguaggio Python, dove tutti gli script scritti sono codice Python.

È inoltre un framework con una buona scalabilità poiché tramite l'uso della riga di comando e l'interfaccia API python fornita da Airtest, gli script possono essere facilmente eseguiti su cluster di apparecchiature su larga scala.

Grazie anche alla precisione dei report HTML, che vengono forniti, è possibile analizzare i passaggi operativi in modo dettagliato, che possono permettere così di individuare rapidamente il punto di errore.

Fornisce inoltre un API multipiattaforma, dove vi è inclusa l'installazione di applicazioni, input analogico e svariate asserzioni.

Dato che la tecnologia di riconoscimento delle immagini viene utilizzata per individuare elementi dell'interfaccia utente, i giochi e le applicazioni possono essere automatizzati anche senza incorporare alcun codice.

9.2 AltUnity Tester

Grazie ad AltUnity Tools, viene fornito una possibilità di soluzione per l'automazione del test dell'interfaccia utente per i giochi Unity in generale, ma anche di applicazioni di realtà aumentata.

AltUnity Tester è una risorsa open-source gratuita, il cui obiettivo fondamentale è quello di abilitare l'automazione dei test dell'interfaccia utente, strumentando i giochi per ottenere l'accesso e controllare a livello di codice gli oggetti Unity.

AltUnity Inspector è invece un'applicazione desktop che consente agli sviluppatori di poter ispezionare la struttura e la gerarchia degli oggetti in modo da interagire con la propria applicazione al di fuori dell'editor di Unity.

9.2.1 Panoramica

AltUnity Tester è quindi uno strumento di automazione dei testing basato sull'interfaccia utente open-source che permette agli sviluppatori di trovare oggetti nel all'interno dell'applicazione create e sviluppata tramite il motore grafico di Unity.

Può inoltre interagire con essi utilizzando script di test scritti in diversi linguaggi di programmazione, tra cui:

- C #,
- Python,
- Java.

Possono essere eseguiti i test su dispositivi reali, sia dispositivi mobili che computer ma anche all'interno dello stesso editor di Unity, oltre che ispezionare la gerarchia degli oggetti di gioco in modo da ottenere facilmente tutte le proprietà, caricando qualsiasi scena o livello del progetto, per controllare ad esempio la velocità dell'applicazione per il debug e per la progettazione del test.

9.2.2 Caratteristiche principali

Questo strumento permette quindi di trovare gli elementi e di ottenere tutte le loro proprietà, di dominio pubblico come le coordinate, il testo, i valori, oppure componenti di Unity.

È possibile, inoltre, utilizzare e modificare anche qualsiasi metodi e/o proprietà degli elementi di Unity, in modo da simulare qualsiasi tipo di funzionamento, come gli input del dispositivo.

Come detto precedentemente è possibile eseguire, dei test in C #, Python o Java utilizzando un qualsiasi IDE[31] e controllare il gioco in esecuzione su un dispositivo o all'interno dell'Editor stesso di Unity, in modo da visualizzare le azioni di input durante l'esecuzione del test, per poi controllare e vedere i risultati dei test e i report sempre all'interno dell'editor di unity.

Vi è la possibilità di essere integrato con i test Appium per la capacità di interagire con elementi nativi.

9.2.3 Come funziona

Il framework di AltUnity Tester contiene e prevede i seguenti moduli, ovvero:

[31] Un IDE, ambiente di sviluppo integrato, è un ambiente di sviluppo ovvero un software che, in fase di programmazione, supporta i programmatori nello sviluppo e debugging del codice sorgente di un programma: spesso l'IDE aiuta lo sviluppatore segnalando errori di sintassi del codice direttamente in fase di scrittura, oltre a tutta una serie di strumenti e funzionalità di supporto alla fase stessa di sviluppo e debugging.

- AltUnity Server, il quale è un modulo che viene utilizzato in modo da mostrare l'accesso a tutti gli oggetti presenti nella gerarchia di Unity, viene aperta una connessione socket TCP sul dispositivo che sta eseguendo l'applicazione ed attende che un client AltUnity si connetta dopo l'avvio dell'applicazione.
- AltUnity Client, questo modulo viene utilizzato per connettersi ad AltUnity Server, così da accedere a tutti gli oggetti che Unity mette a disposizione, e di poter interagire con essi tramite test scritti in C #, Java o Python.
- Finestra dell'editor di AltUnity Tester, che è la l'interfaccia grafica che viene utilizzata per la strumentazione di una applicazione di Unity e l'esecuzione di test C # direttamente da Unity Editor.

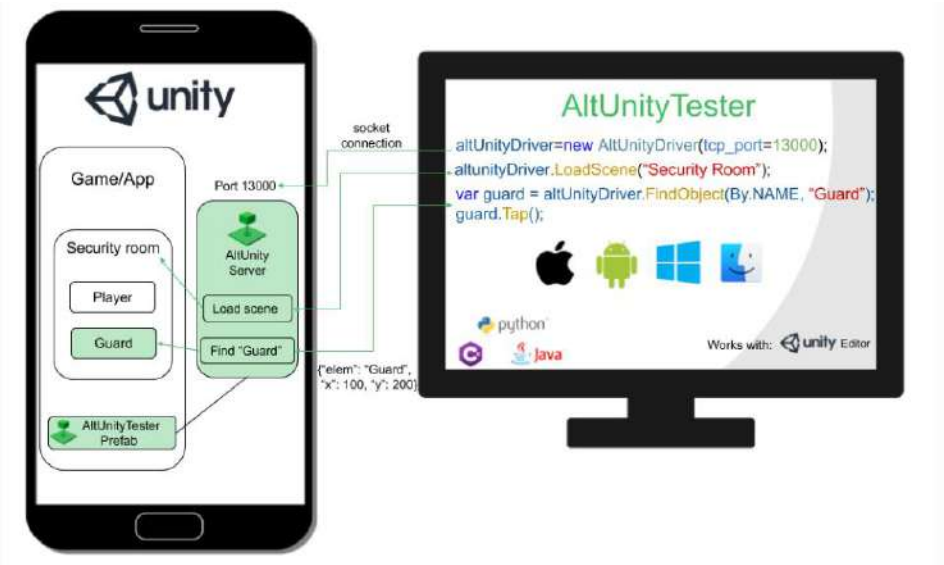

Figura 44 - AltUnity Tester Connessione [20]

9.2.4 Elenco dei test

Come possibile vedere nella figura sottostante (figura 45) è possibile visualizzare tutti i test disponibili direttamente presenti nella cartella del progetto.

Lo sviluppatore può selezionare quali test eseguire, semplicemente selezionando la casella di controllo accanto al nome del test, e può inoltre controllare ogni test individualmente o controllare l'intera classe di test.

I test avranno esito positivo sono contrassegnati con un segno di spunta verde, mentre i test che falliscono presentano una spunta rossa.

È inoltre presente il registro dei test, che contiene un riepilogo delle motivazioni per cui un test è fallito.

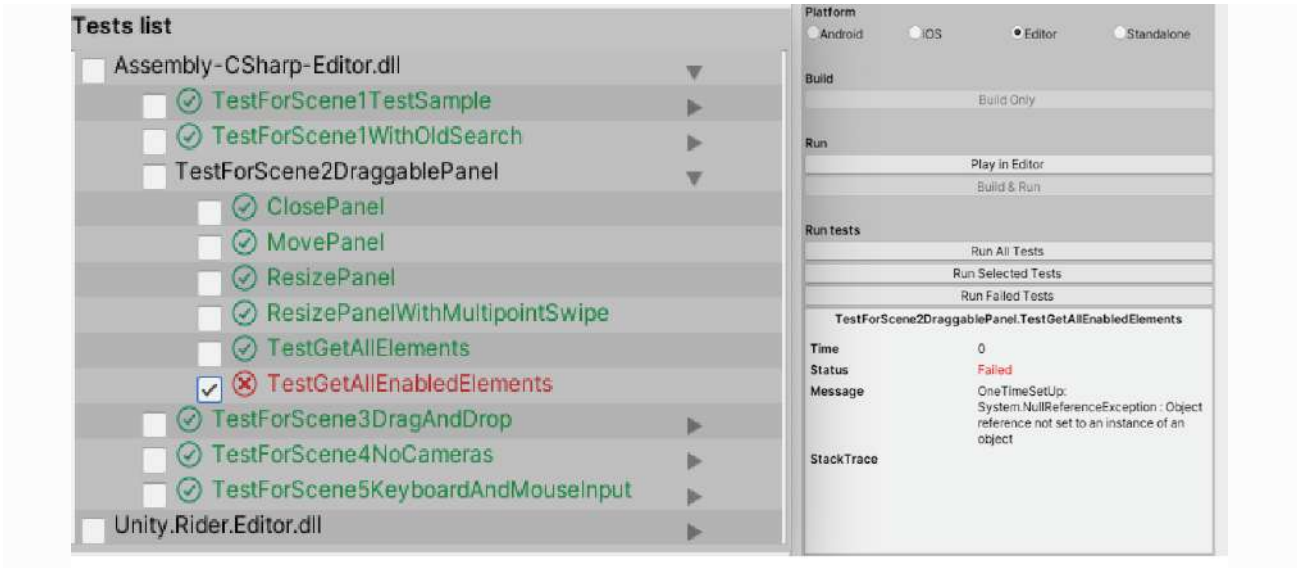

Figura 45 - Pannello di esempio dei test [20]

9.3 Integrazione con Appium per l'esecuzione dei test

Quando si tratta di test di automazione mobile, sono disponibili diverse scelte per uno strumento di test e per poterli eseguire.

Una delle opzioni più popolare ed utilizzata è quella di scegliere Appium, un progetto open-source che consente di eseguire test automatizzati su vari dispositivi mobili sia Android che iOS.

Appium è uno strumento open source per l'automazione di applicazioni native, Web mobile e ibride su piattaforme mobili iOS, mobili Android e desktop Windows.

Ci sono diversi motivi per cui conviene utilizzare Appium insieme ad Altunity Tester, tra le motivazioni più importante vi è quella che AltUnity Tester non può avviare un'applicazione su un dispositivo.

Pertanto, se si vuole eseguire test in una pipeline o utilizzando ad esempio servizi cloud, è possibile creare uno script che avvierà l'applicazione oppure puoi utilizzare Appium stesso prima dell'esecuzione dei test.

Altra motivazione è che AltUnity Tester non è in grado di eseguire alcune tipologie di azioni, come l'interazione con i popup nativi che potrebbe avere la tua app o mettere l'app in background e riprenderla.

In ognuno di questi casi, quindi, sarà possibile utilizzare Appium per svolgere determinate funzioni che AltUnity Tester non può eseguire.

Capitolo 10: Analisi dati di progetti di Realtà Aumentata

In questo capitolo si è affrontata l'analisi dei dati sui progetti di realtà aumentata che sono già sviluppati, in modo da valutare l'andamento di questo nuovo mondo e capire se vi è o meno una espansione sia dal punto di vista applicativo, ma anche e soprattutto di interesse da parte degli sviluppatori di queste applicazioni e dei tester per il controllo e la correzione di eventuali errori magari anche da un punto di vista automatizzato.

E' stato valutato quindi anche se ci sono particolari forme di test che vengono effettuate, e soprattutto come vengono gestite da sviluppatori anche non necessariamente professionisti del settore.

I progetti che sono stati considerati sono quelli che vengono messi a disposizione dagli sviluppatori stessi in modalità open-source sulla piattaforma GitHub[32].

Figura 46 - Logo Github

[32] GitHub è un servizio di hosting per progetti software. Il sito è principalmente utilizzato dagli sviluppatori, che caricano il codice sorgente dei loro programmi e lo rendono scaricabile dagli utenti. Questi ultimi possono interagire con lo sviluppatore tramite un sistema di issue tracking, pull request e commenti che permette di migliorare il codice del repository risolvendo bug o aggiungendo funzionalità.

10.1 Salvataggio su Github

Innanzitutto, da annoverare che per poter reperire tutti gli script che verranno mostrati, ed ampiamente spiegati nei capitoli successivi, sono stati messi a disposizione nel seguente repository, percorribile tramite la ricerca del seguente link:

https://github.com/enzot95/pythonQueryGithub

In questo repository sono stati inseriti tutti i file contenenti le funzioni e gli script progettati per l'analisi dei repositories di Github sulla realtà aumentata.

Inoltre, sono presenti anche i file Json ottenuti come output dell'esecuzione dei vari script, i quali sono stati poi successivamente stati utilizzati come base per l'analisi svolta tramite ulteriori script fatti con il software Matlab.

Per poter eseguire il codice python al suo interno, è stato utilizzato un Ide specifico, con editor di testo, per il linguaggio stesso ovvero PyCharm.

Nel caso in cui non si dovesse avere installato sul proprio pc un Ide simil PyCharm, lo stesso potrà esser eseguito tranquillamente tramite riga di comando, seguendo questi passaggi.

Innanzitutto, come prima cosa bisogna verificare di avere installata una versione di Python sul proprio dispositivo, digitando "py" nel Prompt dei comandi, come mostrato nelle figure seguenti (figura 47-48):

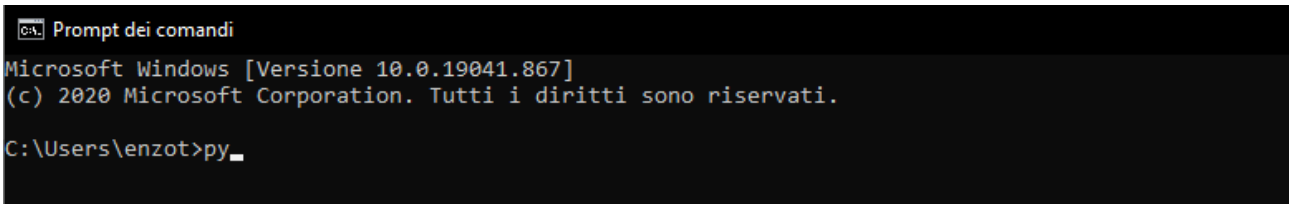

Figura 47 - Verifica presenza di Python (pt.1)

Come possibile vedere nella figura 48, nel caso dovesse esser installata già una versione, comparirà a video la versione presente.

In questo caso vi è la 3.9, ma per l'esecuzione di questi script sarà sufficiente anche possedere la versione dalla 3.6.

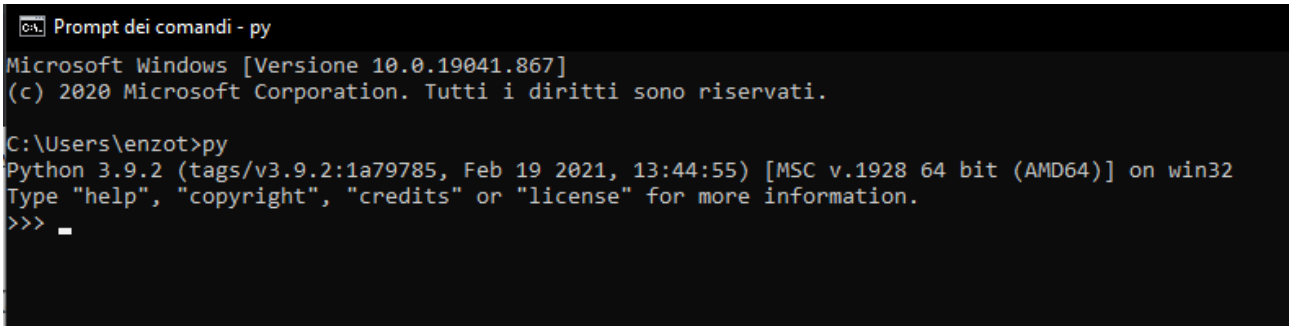

Figura 48 - Verifica presenza di Python (pt.2)

Nel caso in cui non dovesse essere stata installata alcuna versione di python, sarà necessario provvedere alla sua installazione, a seconda del sistema operativo che si ha sul proprio pc.

Una volta avuta a disposizione sul proprio dispositivo una versione sufficiente di python, bisognerà semplicemente eseguire, sempre da terminale, il seguente comando:

python persorso_cartella_script\nome_script.py

ovvero far seguire alla parola chiave "python" il path del file che si vuole eseguire, come mostrato nelle figure successive.

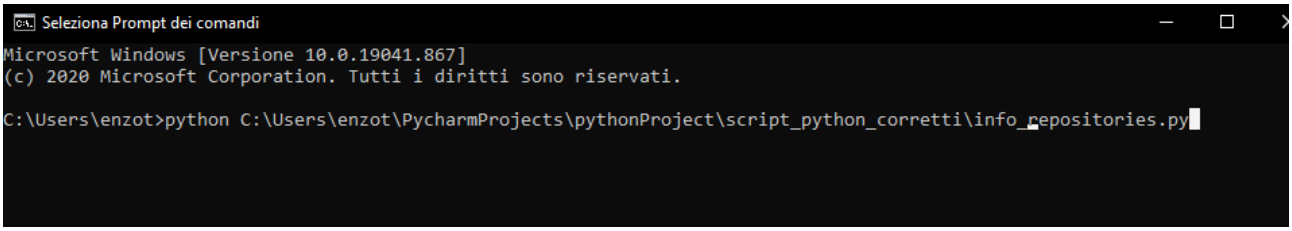

Figura 49 - Inserimento comando per esecuzione script

Una volta inserito il comando basterà premere "invio" ed attendere il termine dell'esecuzione dello script per vedere i risultati.

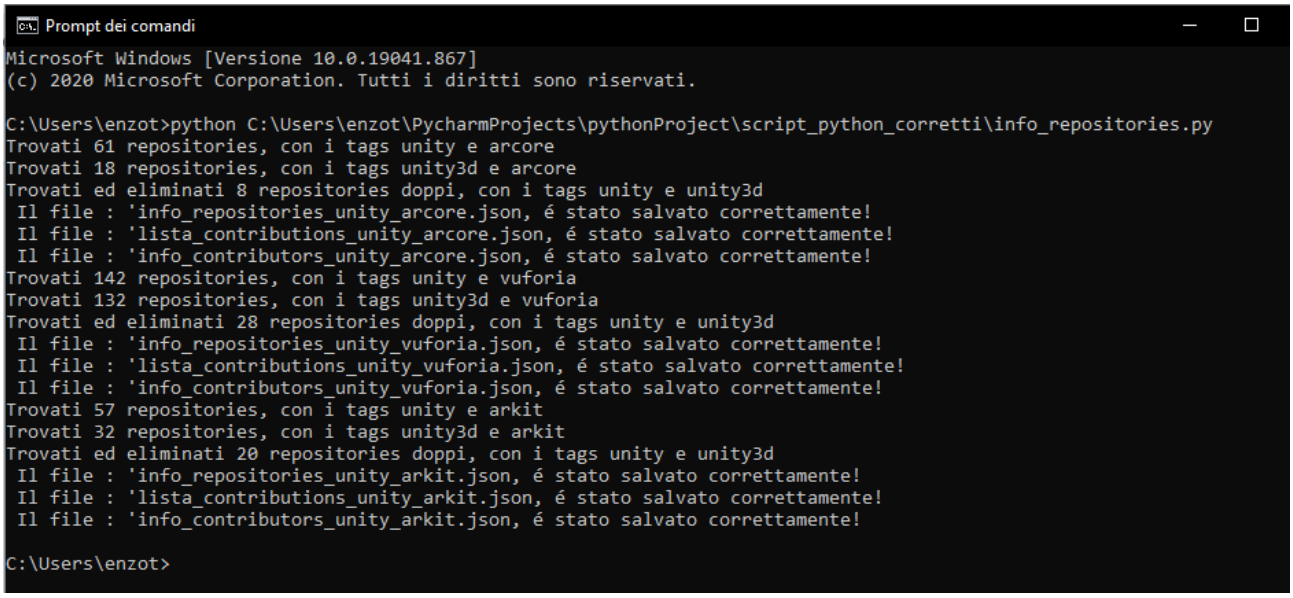

Figura 50 - Risultato esecuzione dello script

10.2 Ricerca dei progetti

Per lo svolgimento di questa analisi sono stati presi in considerazione tutti i progetti che presentano Tag che definiscono il riconoscimento delle varie categorie di appartenenza, in modo da studiare soltanto quei lavori che risultano essere ben ordinati e catalogati dai rispettivi sviluppatori.

Per effettuare questa ricerca sono state fatte dapprima delle rapide ricerche manuali per valutare se ci fossero appunto i presupposti numerici per poter effettuare una analisi statistica sensata, a cui fosse possibile trarre delle conclusioni.

Una volta svolta questa fase di pre-ricerca, sono stati quindi scelti i tags più opportuni per poi effettuare una ricerca delle informazioni facendo uso delle API messe a disposizione da Github stesso.

Facendo quindi una cernita tra i Tags, quelli che sono sembrati più utili alla causa sono stati:

- Unity,
- Unity3D,

- Augmented-Reality,
- AR,
- AugmentedReality,
- Vuforia,
- ArCore,
- Arkit.

Le ricerche che sono state poi effettuate, sono state sia singole che combinate, eliminando gli opportuni progetti doppi.

Come accennato, anche e soprattutto la combinazione di alcuni di questi Tag, hanno suscitato grande interesse e conseguentemente portato a trovare dei progetti, sempre più affini allo scopo della ricerca, ovvero quanto più caratterizzanti nella realtà aumentata.

Nella successiva tabella (tabella 1) è possibile vedere un breve resoconto dei progetti trovati su GitHub:

	TAG	*PROGETTI*
1	UNITY	13077
2	UNITY3D	7966
3	AUGMENTED-REALITY	1605
4	AR	509
5	AUGMENTEDREALITY	76
6	VUFORIA	299
7	ARCORE	220
8	ARKIT	660

Tabella 1 - Riepilogo progetti

Facendo quindi una rapida analisi nel merito della tabella soprariportata (tabella 1), bisogna però dire che i seguenti progetti, che sono stati trovati, presentano fondamentalmente già una sorta di scrematura.

Ovvero, facendo questo tipo di ricerca sono stati indirettamente scartati tutti quei progetti, ai quali gli sviluppatori non hanno applicato l'utilizzo dei tags per permettere una loro migliore classificazione e facile ricerca.
Questo fattore può essere considerato a tutti gli effetti un pregio, poiché essendo ben etichettati, in linea del tutto generale, possono indicare una maggiore cura da parte del/degli sviluppatori, portando a considerarli come dei progetti di qualità migliore di quelli che non li hanno.

Per quanto riguarda i progetti che presentano il tag "Unity" e Unity3d", necessitano di una ulteriore scrematura, nel senso che tra di essi, potrebbero essere presenti non solo progetti facenti parte dell'ambito della realtà aumentata, ma anche di ambiti come videogiochi per smartphone etc.
Per questo sono state fatte anche delle ricerche incrociate, tramite delle funzioni che fanno uso di query in python, pensate e sviluppate ad-hoc per questo tipo di ricerca.

Nel dettaglio sono stati, quindi, combinati i primi due tag sia con quelli che riguardano la realtà aumentata, ossia augmented-reality, ar e augmentedreality, ed in modo parallelo anche i tag vuforia, arcore ed arkit.
Queste due ricerche sono state condotte col fine di analizzare i progetti che sono stati sviluppati con i più famosi ed utilizzati tool, quindi Vuforia, Arcore ed Arkit, ma ciononostante è stata condotta anche una ricerca del tutto generale per poter tener conto anche di come vengono trattate e sviluppate queste applicazioni tramite l'utilizzo di altre piattaforme.

Nella seguente tabella (tabella 2) viene riepilogato brevemente quanto accennato precedentemente:

	COMBINAZIONE DEI TAG	PROGETTI
1 & 3	UNITY & AUGMENTED-REALITY	253
1 & 4	UNITY & AR	104
1 & 5	UNITY & AUGMENTEDREALITY	15
2 & 3	UNITY3D & AUGMENTED-REALITY	229
2 & 4	UNITY3D & AR	54
2 & 5	UNITY3D & AUGMENTEDREALITY	23
1 & 6	UNITY & VUFORIA	141
1 & 7	UNITY E ARCORE	59
1 & 8	UNITY & ARKIT	56
2 & 6	UNITY3D & VUFORIA	133
2 & 7	UNITY3D E ARCORE	35
2 & 8	UNITY3D & ARKIT	31

Tabella 2 - Riepilogo ricerche combinate

Come è possibile notare ovviamente i numeri presenti in questa tabella adesso sono molto più contenuti e presentano quindi con maggior dettaglio le informazioni dei progetti ad essi relativi per quanto riguarda la ricerca.

Per mantenere una ulteriore correzione sono stati eliminati tutti progetti doppi dalle ricerche intrecciate, ed allo stesso tempo non sono stati considerati quelli che erano presenti in più categorie, perché numericamente poco significativi per poterne fare una analisi.

Combinando i tag simili sono stati riscontrati, come mostrato nella tabella seguente, alcuni progetti ripetuti:

	COMBINAZIONE DEI TAG	PROGETTI
1 & 2 & 3	UNITY & UNITY3D & AUGMENTED-REALITY	67
1 & 2 & 4	UNITY & UNITY3D & AR	36
1 & 2 & 5	UNITY & UNITY3D & AUGMENTEDREALITY	10
1 & 2 & 6	UNITY & UNITY 3D & VUFORIA	30
1 & 2 & 7	UNITY & UNITY 3D & ARCORE	17
1 & 2 & 8	UNITY & UNITY 3D & ARKIT	20

Tabella 3 - Riepilogo progetti doppi

Pertanto, si è provveduto quindi alla rimozione dei progetti doppi, ottenendo così la tabella finale, che mostra il numero di progetti a seconda delle relative categorie di appartenenza.

La successiva tabella (tabella 4), mostra quindi i progetti che sono stati "ripuliti" da doppioni:

	COMBINAZIONE DEI TAG	***PROGETTI***
1 & 3	UNITY & AUGMENTED-REALITY	185
1 & 4	UNITY & AR	68
1 & 5	UNITY & AUGMENTEDREALITY	5
2 & 3	UNITY3D & AUGMENTED-REALITY	158
2 & 4	UNITY3D & AR	18
2 & 5	UNITY3D & AUGMENTEDREALITY	12
1 & 6	UNITY & VUFORIA	110
1 & 7	UNITY E ARCORE	42
1 & 8	UNITY & ARKIT	36
2 & 6	UNITY3D & VUFORIA	102
2 & 7	UNITY3D E ARCORE	18
2 & 8	UNITY3D & ARKIT	12

Tabella 4 - Riepilogo progetti senza doppioni

Nelle ricerche che sono state fatte tramite gli script Python, in modo più approfondito e dettagliato, bisogna considerare il fatto che i tag "Unity" ed "Unity3d" sono stati accorpati, in modo da avere dati meno frastagliati, dato che sono comunque due etichette che inglobano progetti di realtà aumentata.

Come si può evincere da questa analisi iniziale, il tool più utilizzato sulla piattaforma di Unity è Vuforia, che copre un totale del 66% dei progetti.

Seguono Arcore con il 19% dei progetti ed Arkit con il 15%.

Il grafico sottostante (figura 51) riassume quindi questi numeri:

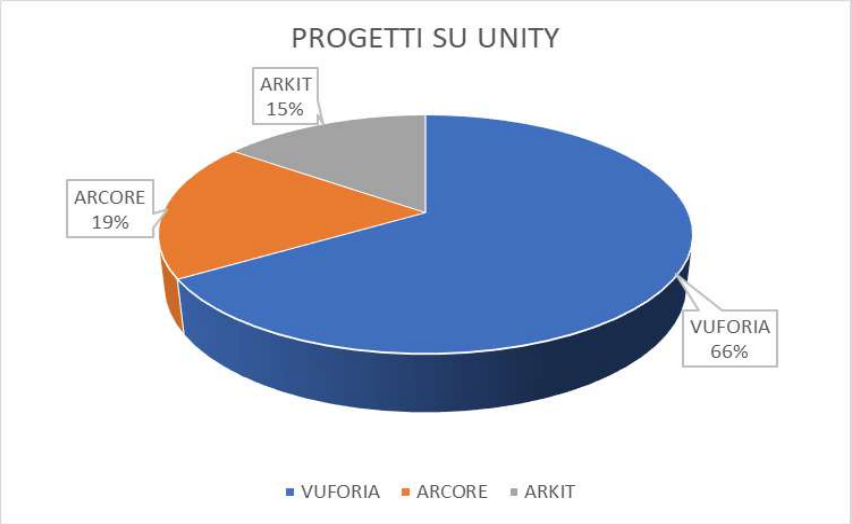

Figura 51 - Tools più usati

Per quanto riguarda l'analisi che verrà esplicitata più dettagliatamente nei paragrafi successivi, vengono presi in considerazione soltanto i progetti che presentano come tag uno dei tre tools, poiché sui 446 progetti totali, in 126 di essi non viene specificato il tool che viene utilizzato (tramite tag), oppure utilizzano altri strumenti come ad esempio Wikitude, ma come descritto precedentemente, i numeri sono troppo esigui per poterne eseguire una stima dettagliata dei progetti.

Per poter comunque classificare ed analizzare questi progetti di realtà aumentata, sono stati quindi analizzati i progetti in linea del tutto generale, facendo uso dei generici tag citati prima, ovvero "AR", "augmentedreality" e "augmented-reality".

10.3 Script per le ricerche

Per condurre le ricerche riportate nel paragrafo precedente, come accennato, sono stati quindi sviluppati degli script scritti in linguaggio Python, facendo

uso dell'ambiente di sviluppo Pycharm per poter svolgere la ricerca delle informazioni tramite le API[33] messe a disposizione dal sito di github.

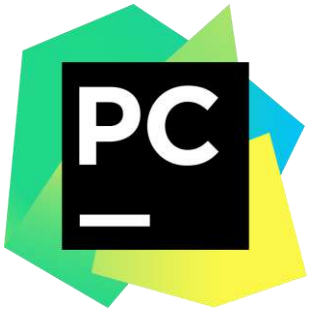

Figura 52 - Logo Pycharm

Successivamente tramite l'ambiente di calcolo numerico e statistico, ovvero Matlab[34], è stato possibile creare anche qui degli script che automatizzassero gli studi analitici delle informazioni che verranno messe a disposizione dopo l'esecuzione degli script Python per la ricerca, salvati in formato JSON[35].

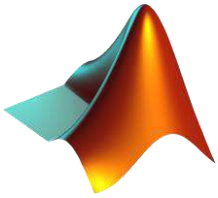

Figura 53 - Logo Matlab

10.3.1 Ricerca dei repositories

Per quanto riguarda la primissima ricerca effettuata, descritta nel paragrafo precedente, per poter permettere l'analisi dei repositories presenti sulla piat-

[33] Le API (acronimo di Application Programming Interface, ovvero Interfaccia di programmazione delle applicazioni) sono set di definizioni e protocolli con i quali vengono realizzati e integrati software applicativi.

[34] MATLAB (abbreviazione di Matrix Laboratory) è un ambiente per il calcolo numerico e l'analisi statistica scritto in C, che comprende anche l'omonimo linguaggio di programmazione creato dalla MathWorks. MATLAB consente di manipolare matrici, visualizzare funzioni e dati, implementare algoritmi, creare interfacce utente, e interfacciarsi con altri programmi.

[35] JSON (JavaScript Object Notation) è uno standard formato di file, e il formato dei dati di interscambio, che usi leggibile testo per memorizzare e trasmettere gli oggetti dati costituito da coppie attributo-valore e tipi di dati array (o qualsiasi altro valore serializzabile).

taforma di github, è stato prodotto il seguente script composto da diverse funzioni di base.

Nelle tabelle sottostanti, ovvero la 5 e 6, viene descritta in modo sintetico una inquadratura generale delle funzioni presenti nella figura 54.

	ELEMENTO	***DESCRIZIONE***
NOME	SEARCH REPOSITORIES ONE TAG	Questa funzione permette di effettuare delle ricerche tramite la query di Requests, sfruttando le Api di Github.
TIPOLOGIA	FUNZIONE	
INPUT	TAG1	Elemento identificativo che viene ultilizzato per poter effettuare la ricerca della query in modo del tutto generico, rendendo questa funzione riutilizzabile.
OUTPUT	REPOSITORIES	Lista delle informazioni ottenute dall'esecuzione della query.
ESEMPIO D'USO	funzioni.search_repositories_one_tag(tag1)	

Tabella 5 - Informazioni della funzione "search_repositories_one_tag"

	ELEMENTO	***DESCRIZIONE***
NOME	SEARCH REPOSITORIES TWO TAG	Questa funzione permette di effettuare delle ricerche tramite la query di Requests, sfruttando le Api di Github, utilizzando come elemento della ricerca 2 parole chiave (tags) che rendono la funzione riutilizzabile.
TIPOLOGIA	FUNZIONE	
INPUT	TAG1	Elemento identificativo che viene ultilizzato per poter effettuare la ricerca della query in modo del tutto generico, rendendo questa funzione riutilizzabile.
	TAG2	Elemento identificativo che viene ultilizzato per poter effettuare la ricerca della query in modo del tutto generico, rendendo questa funzione riutilizzabile.
OUTPUT	REPOSITORIES	Lista delle informazioni ottenute dall'esecuzione della query.
ESEMPIO D'USO	funzioni.search_repositories_two_tag(tag1,tag2)	

Tabella 6 - Informazioni della funzione "search_repositories_two_tags"

Il codice delle funzioni appena descritte è visibile nell'immagine sottostante (figura 54), le quali vengono poi utilizzate all'interno di uno script per la ricerca delle informazioni sui repositories.

```
def search_repositories_one_tag(tag1):

    #ricerca dei repositories di github senza tenere in considerazione dei
progetti duplicati
    repositories = reque-
sts.get('https://api.github.com/search/repositories?q=%23' + tag1,
                                auth=('enzot95', ACCESS_TOKEN)).json()
    print("- " + str(repositories["total_count"]) + " repositories con il
tag " + tag1)

    return repositories

def search_repositories_two_tags(tag1,tag2):
    #intersezione dei sw con i tools (senza considerare gli eventuali du-
plicati)
    repositories = reque-
sts.get('https://api.github.com/search/repositories?q=%23' + tag1 + '+ %23'
+ tag2,
                                auth=('enzot95', ACCESS_TOKEN)).json()
    #print("- " + str(repositories["total_count"]) + " repositories con i
tag " + tag1 + ' e ' + tag2)

    return repositories
```

Figura 54 - Query per le ricerca dei repositories

```
import funzioni

#TAG PER LA RICERCA DEI REPOSITORY DI GITHUB
sws = ['unity','unity3d']
types = ['augmentedreality','augmented-reality','ar']
tools = ['vuforia','arcore','arkit']

# RICERCA AD 1 TAG
print("Ricerca di 1 tag")
print("Sono presenti in totale: ")
for sw in sws:
    funzioni.search_repositories_one_tag(sw)

print("Sono presenti in totale: ")
for type in types:
    funzioni.search_repositories_one_tag(type)

print("Sono presenti in totale: ")
for tool in tools:
    funzioni.search_repositories_one_tag(tool)

# INTERSEZIONE DI 2 TAGS
print("Intersezione di 2 tags")
print("Sono presenti in totale: ")
for sw in sws:
     for tool in tools:
        funzioni.search_repositories_two_tags(sw, tool)
```

```
print("Sono presenti in totale: ")
for sw in sws:
    for type in types:
        funzioni.search_repositories_two_tags(sw, type)
```

Figura 55 – Script per l'esecuzione delle ricerche

Nell'immagine precedente (figura 55) viene mostrato il codice dello script che utilizzerà le due funzioni illustrate per poter eseguire le query su github, mentre nella seguente tabella 7 vengono sintetizzate le informazioni inerenti il medesimo script sulla ricerca delle informazioni.

	ELEMENTO	***DESCRIZIONE***
NOME	RICERCA REPOSITORIES	Questo script consente di poter eseguire le funzioni "Search repositories One Tag" e "Search repositories two Tags", utilizzando diverse parole chiave per poter effettuare diverse ricerche in modo parallelo, così da ottenere diverse informazioni.
TIPOLOGIA	SCRIPT	
INPUT	ELENCO TAG SOFTWARE	Elenco dei tag inerenti ai software di realtà aumentata
	ELENCO TAG AR	Elenco dei tag inerenti alla realtà aumentata
	ELENCO TAG TOOLS	Elenco dei tag inerenti ai tool di realtà aumentata.
FUNZIONI USATE	SEARCH REPOSITORIES ONE TAG	Funzione di ricerca tramite un tag per volta
	SEARCH REPOSITORIES TWO TAG	Funzione di ricerca tramite due tag per volta
OUTPUT	INFORMAZIONI	Elenco di informazioni visibili tramite console inerenti ai repositories ricercati.
ESEMPIO D'USO	ricerca_repositories.py (da riga di comando)	

Tabella 7 - Informazioni dello script "ricerca_repositories"

Una volta che viene mandato in esecuzione lo script, da riga di comando oppure utilizzando il software Pycharm[36], nelle rispettive console viene mostrato il seguente output (figure 56 e 57) con i seguenti risultati.

```
Ricerca di 1 tag
Sono presenti in totale:
- 13077 repositories con il tag unity
- 7966 repositories con il tag unity3d
Sono presenti in totale:
- 76 repositories con il tag augmentedreality
- 1605 repositories con il tag augmented-reality
- 509 repositories con il tag ar
Sono presenti in totale:
- 299 repositories con il tag vuforia
- 220 repositories con il tag arcore
- 660 repositories con il tag arkit
```

Figura 56 - Risultato (pt.1)

```
Intersezione di 2 tags
Sono presenti in totale:
- 141 repositories con i tag unity e vuforia
- 59 repositories con i tag unity e arcore
- 56 repositories con i tag unity e arkit
- 133 repositories con i tag unity3d e vuforia
- 35 repositories con i tag unity3d e arcore
- 31 repositories con i tag unity3d e arkit
Sono presenti in totale:
- 15 repositories con i tag unity e augmentedreality
- 253 repositories con i tag unity e augmented-reality
- 104 repositories con i tag unity e ar
- 23 repositories con i tag unity3d e augmentedreality
- 229 repositories con i tag unity3d e augmented-reality
- 54 repositories con i tag unity3d e ar
```

Figura 57 - Risultato (pt.2)

I risultati sono gli stessi presenti nella tabella 1 del capitolo precedente, sia per quanto riguarda la ricerca a singolo tag, che per quanto riguarda la ricerca combinata a due tags.

[36] PyCharm è un ambiente di sviluppo integrato (IDE) utilizzato nella programmazione di computer, specificamente per il linguaggio Python.

10.3.2 Ricerca informazioni ed eliminazione progetti doppi

Nel seguente paragrafo viene descritto uno script che permette di effettuare la ricerca dei repositories, e più nel dettaglio delle informazioni e delle issues riguardanti i progetti che sono presenti su Github con i tags “Unity”, “Unity3d”, “Vuforia”, “Arcore” e “Arkit”.

Per la realizzazione di questo script sono state sviluppate le funzioni che permettono di poter effettuare questa ricerca, grazie anche e soprattutto alle query utilizzate in esse.

Queste sono ovviamente necessarie per effettuare la ricerca delle informazioni dei vari repositories, che verranno poi analizzate negli script Matlab.

Sono stati inoltre eliminati tutti quei progetti “doppi” che erano presenti in più ricerche, e per ogni informazione ottenuta delle ricerche effettuate viene creato e salvato un file JSON che le contiene.

Il codice delle funzioni e delle query che sono state utilizzate per l’esecuzione di questa ricerca, vengono raffigurate nelle seguenti immagini (da figura 58 a figura 64).

	ELEMENTO	***DESCRIZIONE***
NOME	SALVA REPOSITORIES TWO TAG	Questa funzione permette di poter salvare in un file json tutte le informazioni che si è ottenuti tramite le opportune ricerche.
TIPOLOGIA	FUNZIONE	
INPUT	INFO	Informazioni e dati da salvare in un file json
	TITLE	Nome che si vuole dare al file Json da creare
	TAG1	Tag1 identificativo, da utilizzare all'interno del nome del file Json
	TAG2	Tag2 identificativo, da utilizzare all'interno del nome del file Json
OUTPUT	FILE JSON	File Json contenete le informazioni da salvare

ESEMPIO D'USO	Funzioni.salva_repositories_two_tag(info,title,tag1,tag2)

Tabella 8 - Informazioni funzione "salva repositories two tags"

La funzione in figura 54 serve per la creazione ed il salvataggio della lista di informazioni in formato JSON:

```
def salva_repositories_two_tag(info, title, tag1,tag2):

    # salvataggio dei repositories in formato json
    with open(title+"_" + tag1 + "_" + tag2 + ".json", "w") as outfile:
        json.dump(info, outfile, indent=4, sort_keys=True)
    print(" Il file : '"+title+"_" + tag1 + "_" + tag2 + ".json, é stato
salvato correttamente!")
```

Figura 58 - Funzione "salva_repositories_two_tag"

Nelle figure 59 e 60 viene mostrato il codice della funzione che permette di ricercare le informazioni riguardanti i repositories, quali:

- Id, ossia l'identificativo del progetto;
- Language, attributo che indica il linguaggio che è maggiormente presente, ed utilizzato all'interno del progetto;
- Open issues, che indica il numero di quante issues sono aperte per quel determinato progetto;
- Created_at, che indica la data del primo caricamento effettuato sul progetto;
- Updated_at, che indica la data dell'ultimo caricamento effettuato sul progetto.

	ELEMENTO	DESCRIZIONE
NOME	RICERCA INFO	Questa funzione permette di poter effettuare delle ricerche delle informazioni inerenti ai repositories di github, sfogliando le pagine del sito, in modo da riuscire a prendere tutte le informazioni ed i dati.
TIPOLOGIA	FUNZIONE	
INPUT	TAG1	Tag1 identificativo, da utilizzare all'interno della query delle API di github
	TAG2	Tag2 identificativo, da utilizzare all'interno della query delle API di github
	REPOSITORIES	Elenco dei nomi dei repositories in cui dover ricercare le informazioni.
OUTPUT	INFO	Lista contenente tutte le informazioni memorizzate ricercate tramite la query dei repositories
ESEMPIO D'USO	Funzioni.ricerca_info(tag1,tag2,repositories)	

Tabella 9 - Informazione funzione "ricerca_info"

```
def ricerca_info(tag1, tag2, total_repositories):

    ids = []
    linguaggi = []
    issues = []
    data_creazione = []
    data_aggiornamento = []
    visualizzazioni = []

    lista_pagine = []

    if (total_repositories["total_count"]) < 50:
        total_count_sws = 1
    else:
        total_count_sws = round(total_repositories["total_count"] / 100)+1
        #print(total_count_sws)

    for numero_pagina in range(total_count_sws):
        page = requests.get(
            'https://api.github.com/search/repositories?q=%23' + tag1 +
'+%23' + tag2 + '&page=' + str(
                numero_pagina+1) + '&per_page=100',
            auth=('enzot95', ACCESS_TOKEN)).json()
        lista_pagine.append(page)
```

Figura 59 - Funzione "ricerca_info" (pt.1)

```
    for i in page["items"]:
        ids.append(i["full_name"])
        if i["language"]!= "Objective-C++" and i["language"]!="C#" and
i["language"]!="Kotlin" and i["language"]!= "HTML" \
                and i["language"]!="ShaderLab" and i["language"]!="ASP.NET"
and i["language"]!="Java" and i["language"]!="ASP" \
                and i["language"]!="C++":
            linguaggi.append("null")
        else:
            linguaggi.append(i["language"])
        issues.append(i["open_issues"])
        data_creazione.append(i["created_at"])
        data_aggiornamento.append(i["updated_at"])
        visualizzazioni.append(i["watchers"])

info = []
info.append(ids)
info.append(linguaggi)
info.append(issues)
info.append(data_creazione)
info.append(data_aggiornamento)
info.append(visualizzazioni)

return info
```

Figura 60 - Funzione "ricerca_info" (pt.2)

La funzione in figura 61 e 62 mostra un'altra ricerca che viene effettuata, ovvero quella tramite identificativo del progetto.

	ELEMENTO	***DESCRIZIONE***
NOME	RICERCA INFO PER ID	Questa funzione permette di poter effettuare delle ricerche delle informazioni inerenti ai repositories di github, dato in ingresso soltanto il nome dei repositories.
TIPOLOGIA	FUNZIONE	
INPUT	LISTA ID	Elenco dei nomi dei repositories in cui dover ricercare le informazioni.
OUTPUT	INFO	Lista contenente tutte le informazioni memorizzate ricercate tramite la query dei repositories
ESEMPIO D'USO	Funzioni.ricerca_info_per_id(lista_id)	

Tabella 10 - Informazioni funzione "ricerca info per id"

```
def ricerca_info_per_id(lista_id):

    repositories = []
    for id in lista_id[0]:
        repositories.append(requests.get('https://api.github.com/repos/' +
id ,auth=('enzot95', ACCESS_TOKEN)).json())

    ids = []
    linguaggi = []
    issues = []
    data_creazione = []
    data_aggiornamento = []
    visualizzazioni = []

    for i in repositories:
        ids.append(i["full_name"])
        if i["language"]!= "Objective-C++" and i["language"]!="C#" and
i["language"]!="Kotlin" \
                and i["language"]!= "HTML" and i["language"]!="ShaderLab"
and i["language"]!="ASP.NET" \
                and i["language"]!="Java" and i["language"]!="ASP" and
i["language"]!="C++":
            linguaggi.append("null")
        else:
            linguaggi.append(i["language"])
        issues.append(i["open_issues"])
        data_creazione.append(i["created_at"])
        data_aggiornamento.append(i["updated_at"])
        visualizzazioni.append(i["watchers"])
```

Figura 61 - Funzione "ricerca_info_per_id" (pt.1)

```
info = []
info.append(ids)
info.append(linguaggi)
info.append(issues)
info.append(data_creazione)
info.append(data_aggiornamento)
info.append(visualizzazioni)

return info
```

Figura 62 - Funzione "ricerca_info_per_id" (pt.2)

Altra funzione di ricerca è quella di figura 63 che permette di effettuare la ricerca delle informazioni inerenti ad i contributors, ovvero coloro che hanno preso parte con dei caricamenti al progetto in questione.

Essa prende in ingresso la lista degli id, a cui si vuole prelevare le informazioni.

	ELEMENTO	DESCRIZIONE
NOME	RICERCA CONTRIBUTORS	Questa funzione permette di poter effettuare delle ricerche delle informazioni inerenti ai contributors dei repositories di github, dato in ingresso soltanto l'elenco dei nomi dei repositories.
TIPOLOGIA	FUNZIONE	
INPUT	LISTA ID	Elenco dei nomi dei repositories in cui dover ricercare le informazioni.
OUTPUT	LISTA CONTRIBUTORS	Lista contenente tutte le informazioni principali sui contributors filtrate e memorizzate ricercate tramite la query dei repositories
	INFO CONTRIBUTORS	Lista contenente tutte le informazioni sui contributors memorizzate, tramite la ricerca della query dei repositories
ESEMPIO D'USO	Funzioni.ricerca_contributors(lista_id)	

Tabella 11 - Informazioni fuznione "ricerca_contributors"

```
def ricerca_contributors(lista_id):

    contributors = []
    lista_contributions=[]
    info_contributors=[]
    for id in lista_id[0]:
        info=requests.get('https://api.github.com/repos/' + id
+'/contributors' ,auth=('enzot95', ACCESS_TOKEN)).json()
        contributors.append(info)
        lista_contributions.append(len(info))

    #print(json.dumps(contributors,indent=4, sort_keys=True))
    for ids in contributors:
        if len(ids)!=0:
            for id in ids:
                info_contributors.append(id["type"])
                info_contributors.append(id["contributions"])


    return lista_contributions,info_contributors
```

Figura 63 - Funzione "ricerca_contributors"

Di fondamentale importanza è la funzione di figura 64, la quale serve per poter effettuare il controllo di eventuali progetti doppi in più categorie di ricerca.

Per poter essere gestita necessita di avere in ingresso due parametri che corrispondono alle sue liste di identificativi che si vuole controllare, ed eliminare.

	ELEMENTO	DESCRIZIONE
NOME	CONTROLLO PROGETTI DOPPI	Questa funzione permette di poter effettuare il controllo di due elenchi di informazioni in modo da eliminare elementi doppi cosi da avere un unico elenco di informazioni uniche.
TIPOLOGIA	FUNZIONE	
INPUT	INFO1	Elenco dei nomi dei repositories in cui dover ricercare le informazioni.
	INFO2	Elenco dei nomi dei repositories in cui dover ricercare le informazioni.
OUTPUT	INFO	Lista contenente tutte le informazioni filtrate e prive di doppioni.
ESEMPIO D'USO	Funzioni.controllo_progetti_doppi(info1,info2)	

Tabella 12 - Informazione funzione "controllo progetti doppi"

```
def controllo_progetti_doppi(info1,info2):

    # pulizia di progetti doppi
    count=0
    info = []
    info.append(info1[0])
    for id in info2[0]:
        for i in info[0]:
            if id == i:
                count=count+1
                info[0].remove(i)
    for id in info2[0]:
        info[0].append(id)

    return info
```

Figura 64 - Funzione "controllo_progetti_doppi"

	ELEMENTO	DESCRIZIONE
NOME	RICERCA REPOSITORIES	Questo script consente di poter eseguire delle funzioni che realizzano delle ricerche sui repositories e le relative informazioni su di essi, utilizzando diverse parole chiave per poter effettuare diverse ricerche in modo parallelo, così da ottenere diverse informazioni.
TIPOLOGIA	SCRIPT	
INPUT DATI	ELENCO TAG SOFTWARE	Elenco dei tag inerenti ai software di realtà aumentata
	ELENCO TAG TOOLS	Elenco dei tag inerenti ai tool di realtà aumentata.
FUNZIONI USATE	RICERCA INFO	Funzione di ricerca delle informazioni inerenti ai repositories
	CONTROLLO PROGETTI DOPPI	Funzione di controllo per verificare se le informazioni doppie all'interno di elenchi di informazioni sui repositories
	RICERCA INFO PER ID	Funzione di ricerca delle informazioni sui repositories, dato come input l'elenco dei nomi dei repositories
	RICERCA CONTRIBUTORS	Funzione di ricerca delle informazioni sui contributors, dato come input l'elenco dei nomi dei repositories
	SALVA REPOSITORIES TWO TAGS	Funzione di salvataggio delle informazioni ricercate, memorizzate in un apposito file json.
OUTPUT	INFORMAZIONI	Elenco di informazioni visibili tramite console inerenti ai repositories ricercati.
	FILE JSON	File Json creati contenenti le varie informazioni ricercate.
ESEMPIO D'USO	Ricerca_repositories.py (da eseguire da riga di comando)	

Tabella 13 - Informazioni script "ricerca repositories"

Quindi, le precedenti funzioni che sono state mostrate e brevemente descritte nelle tabelle precedenti, sono state utilizzate per l'esecuzione del seguente script (figura 65 e 66).

```
import funzioni

# token github
ACCESS_TOKEN = '68d3043d0154715a436f34adba3f3c490bcb5443'

#TAG PER LA RICERCA DEI REPOSITORY DI GITHUB
sws = ['unity','unity3d']
tools = ['arcore','vuforia','arkit']

# per svolgimento operazioni di ricerca delle informazioni nelle reposito-
ries
for tool in tools:
    #RICERCA DELLE INFO DEI REPOSITORIES
    info= funzioni.ricerca_info(sws[0], tool, funzio-
ni.search_repositories_two_tags(sws[0], tool))
    len_info=len(info[0])
    print("Trovati "+str(len_info)+" repositories, con i tags "+sws[0]+" e
"+tool)

    info_3d = funzioni.ricerca_info(sws[1], tool, funzio-
ni.search_repositories_two_tags(sws[1], tool))
    len_info_3d=len(info_3d[0])
    print("Trovati " + str(len_info_3d) + " repositories, con i tags " +
sws[1] + " e " + tool)

    #controllo ed eliminazione dei progetti doppi
    lista_id = funzioni.controllo_progetti_doppi(info, info_3d)
    print("Trovati ed eliminati "+str(len_info+len_info_3d-
len(lista_id[0]))+" repositories doppi, "

"con i tags "+sws[0]+" e "+sws[1])

    #ricerca dei repositories per id, e salvataggio su file json
    info= funzioni.ricerca_info_per_id(lista_id)
    funzioni.salva_repositories_two_tag(info, "info_repositories", "unity",
tool)
```

Figura 65 - Script info repositeries (pt.1)

```
#contributors presenti dei progetti
lista_contributions,info_contributors= funzio-
ni.ricerca_contributors(lista_id)
funzioni.salva_repositories_two_tag(lista_contributions, "li-
sta_contributions", "unity", tool)
funzioni.salva_repositories_two_tag(info_contributors, "info_contributors",
"unity", tool)
```

Figura 66 - Script info repositories (pt.2)

Una volta eseguito lo script precedentemente raffigurato, sarà possibile vedere nella console in output (figura 67), i seguenti risultati:

- Numero dei progetti trovati, a seconda della richiesta specificata nella query relativa, per i tags selezionati;
- messaggio di avvenuto controllo ed eliminazione dei progetti doppi;

- messaggio di avvenuta creazione e salvataggio delle informazioni in file JSON, le quali informazioni contenute in essi verranno in seguito utilizzate per l'analisi e la creazione dei grafici.

Figura 67 - Risultato script

10.3.3 Ricerca dei progetti generici di realtà aumentata

Altra ricerca che è stata condotta riguarda la selezione dei progetti di Github che presentano come tag quelli inerenti alla realtà aumentata, senza però specificare il tool specifico, ovvero Vuforia, Arcore o Arkit, ma solo quelli svolti nell'ambiente Unity, in modo da avere un quadro più generale.

	ELEMENTO	***DESCRIZIONE***
NOME	INFO ISSUES AUGMENTED REALITY	Questo script consente di poter eseguire delle funzioni che realizzano delle ricerche sui repositories e le relative informazioni riguardanti delle issues su di essi, utilizzando diverse parole chiave per poter effettuare diverse ricerche in modo parallelo, così da ottenere diverse informazioni.
TIPOLOGIA	SCRIPT	

INPUT DATI	ELENCO TAG SOFTWARE	Elenco dei tag inerenti ai software di realtà aumentata
	ELENCO TAG TOOLS	Elenco dei tag inerenti ai tool di realtà aumentata.
	TAG	Tag utilizzato per i salvataggi ovvero, "augmented_reality"
FUNZIONI USATE	RICERCA INFO	Funzione di ricerca delle informazioni inerenti ai repositories
	CONTROLLO PROGETTI DOPPI	Funzione di controllo per verificare se le informazioni doppie all'interno di elenchi di informazioni sui repositories
	RICERCA ISSUES	Funzione di ricerca delle informazioni sulle issues dei repositories, dato come input l'elenco dei nomi dei repositories
	RICERCA LABEL	Funzione di ricerca delle informazioni sulle label delle issues dei repositories, dato come input l'elenco dei nomi dei repositories
	RICERCA CONTRIBUTORS	Funzione di ricerca delle informazioni sui contributors, dato come input l'elenco dei nomi dei repositories
	SALVA REPOSITORIES TWO TAGS	Funzione di salvataggio delle informazioni ricercate, memorizzate in un apposito file json.
OUTPUT	INFORMAZIONI	Elenco di informazioni visibili tramite console inerenti ai repositories ricercati.
	FILE JSON	File Json creati contenenti le varie informazioni ricercate.
ESEMPIO D'USO	Info_issues_augmented_reality.py (da eseguire da riga di comando)	

Tabella 14 - Informazioni script "info issues repositories realtà aumentata"

Pertanto, lo script utilizzato è il seguente (figure 68, 69, 70 e 71):

```
import funzioni

# token github
ACCESS_TOKEN = '68d3043d0154715a436f34adba3f3c490bcb5443'

#TAG PER LA RICERCA DEI REPOSITORY DI GITHUB
sws = ['unity','unity3d']
types = ['augmentedreality','augmented-reality','ar']
tag = "augmented_reality"

# ricerca dei repositories
lista_info = []
for type in types:
    info= funzioni.ricerca_info(sws[0], type, funzio-
ni.search_repositories_two_tags(sws[0], type))
```

```
    info_3d= funzioni.ricerca_info(sws[1], type, funzio-
ni.search_repositories_two_tags(sws[1], type))
    lista_info.append(funzioni.controllo_progetti_doppi(info, info_3d))
    #print(lista_info)

info_tot = funzioni.controllo_progetti_doppi(lista_info[0], lista_info[1])
```

Figura 68 - Script ricerca info issues progetti realtà aumentata (pt.1)

```
#creazione e salvataggio dell'elenco degli id dei progetti, con eliminazio-
ne dei progetti doppi
lista_id = funzioni.controllo_progetti_doppi(lista_info[2], info_tot)
funzioni.salva_repositories_two_tag(lista_id, "ids", "unity", tag)

#creazione e salvataggio dell'elenco delle informazioni delle repositories
repositories = []
repositories.append(funzioni.ricerca_info_per_id(lista_id))
funzioni.salva_repositories_two_tag(repositories, "repositories", "unity",
tag)

#RICERCA DELLE ISSUES
issues_info_open = funzioni.ricerca_issues(lista_id, "aperte")
issues_info_close = funzioni.ricerca_issues(lista_id, "chiuse")
funzioni.salva_repositories_two_tag(issues_info_open, "issues_info_open",
"unity", tag)
funzioni.salva_repositories_two_tag(issues_info_close, "issues_info_close",
"unity", tag)

# SALVATAGGIO DELLE ISSUES APERTE DEI REPOSITORIES
issues_open = []
issues_open = funzioni.ricerca_label(issues_info_open)
funzioni.salva_repositories_two_tag(issues_open, "issues_open", "unity",
tag)

#SALVATAGGIO DELLE ISSUES CHIUSE DEI REPOSITORIES
issues_close = []
issues_close = funzioni.ricerca_label(issues_info_close)
funzioni.salva_repositories_two_tag(issues_close, "issues_close", "unity",
tag)
funzioni.salva_repositories_two_tag(lista_contributions, "li-
sta_contributions", "unity", "AR")
funzioni.salva_repositories_two_tag(info_contributors, "info_contributors",
"unity", "AR")
```

Figura 69 - Script ricerca info issues progetti realtà aumentata (pt.2)

```
#RICERCA DELLE ISSUES APERTE
issues_label_open = []
for progetto in issues_open:
    for issue in progetto:
        issues_label_open.append(issue["name"])

funzioni.salva_repositories_two_tag(issues_label_open, "label_open", "uni-
ty", tag)
```

Figura 70 - Script ricerca info issues progetti realtà aumentata (pt.3)

```
#RICERCA DELLE ISSUES CHIUSE
issues_label_close = []
```

```
for progetto in issues_close:
    for issue in progetto:
        issues_label_close.append(issue["name"])

funzioni.salva_repositories_two_tag(issues_label_close, "label_close",
"unity", tag)

#contributors presenti dei progetti
lista_contributions,info_contributors= funzio-
ni.ricerca_contributors(lista_id)
funzioni.salva_repositories_two_tag(lista_contributions, "li-
sta_contributions", "unity", "AR")
funzioni.salva_repositories_two_tag(info_contributors, "info_contributors",
"unity", "AR")
```

Figura 71 - Script ricerca progetti realtà aumentata (pt.4)

In questo script sono state utilizzate alcune funzioni mostrate anche nel paragrafo precedente, nel dettaglio quelle mostrate nelle figure da 58 a 64.

	ELEMENTO	***DESCRIZIONE***
NOME	RICERCA ISSUES	Questa funzione permette di poter effettuare delle ricerche delle informazioni inerenti alle issues dei repositories di github, dato in ingresso soltanto l'elenco dei nomi dei repositories.
TIPOLOGIA	FUNZIONE	
INPUT	INFO	Elenco dei nomi dei repositories in cui dover ricercare le informazioni.
	TIPOLOGIA	Parola chiave che permette di poter capire se si tratta di ricercare delle issues aperte o chiuse
OUTPUT	ISSUES INFO	Lista contenente tutte le informazioni principali sulle issues filtrate e memorizzate dalla ricerca tramite la query dei repositories
ESEMPIO D'USO	Funzioni.ricerca_issues(info,tipologia)	

Tabella 15 - Informazioni funzione "ricerca issues"

Inoltre, sono state sviluppate due ulteriori funzioni che servono per la ricerca delle informazioni riguardante le issues, e le etichette ad esse riferite.

```
def ricerca_issues(info,tipologia):

    issues_info = []
    if tipologia == "aperte":
        # ricerca delle issues aperte
        for id in info[0]:
            issues_open = requests.get('https://api.github.com/repos/' + id
+ '/issues?page=1&per_page=100',
                                       auth=('enzot95',
ACCESS_TOKEN)).json()
            if len(issues_open) != 0:
                issues_info.append(issues_open)
    else:
        #ricerca delle issues chiuse
        for id in info[0]:
            issues_close = requests.get(
                'https://api.github.com/repos/' + id +
'/issues?state=closed&page=1&per_page=100',
                auth=('enzot95', ACCESS_TOKEN)).json()
            if len(issues_close) != 0:
                issues_info.append(issues_close)

    return issues_info
```

Figura 72 - Funzione "ricerca_issues"

	ELEMENTO	DESCRIZIONE
NOME	RICERCA LABEL	Questa funzione permette di poter effettuare delle ricerche delle informazioni inerenti alle label delle issues dei repositories di github, dato in ingresso soltanto l'elenco dei nomi dei repositories.
TIPOLOGIA	FUNZIONE	
INPUT	ISSUES	Informazioni delle issues che vanno filtrate in modo da ottenere le label
OUTPUT	ISSUES LABEL	Lista contenente tutte le informazioni sulle label delle issues passate come input.
ESEMPIO D'USO	Funzioni.ricerca_label(issues)	

Tabella 16 - Informazioni funzione "ricerca label"

```
def ricerca_label(issues):

    #ricerca delle label nelle issues
    issues_label = []
    for progetto_open in issues:
        for issue_open in progetto_open:
            if len(issue_open["labels"]) != 0:
                # print(len(issue["labels"]))
                issues_label.append(issue_open["labels"])

    return issues_label
```

Figura 73 - Funzione "ricerca_label"

Queste funzioni consentono la ricerca delle issues (figura 72), sia aperte che chiuse dei vari progetti, con la selezione e creazione di liste di informazioni inerenti ad esse.

Inoltre, nella seconda delle due funzioni (figura 73), permette di creare una lista contenente tutte le labels trovate nelle issues dei progetti, da considerare che quest'ultime però non vengono tutte etichettate, pertanto ai fini statistici sono state considerate ed analizzate solo quelle che le presentano.

Eseguendo lo script di figura 68-71, nella console viene mostrato il seguente output (figura 74):

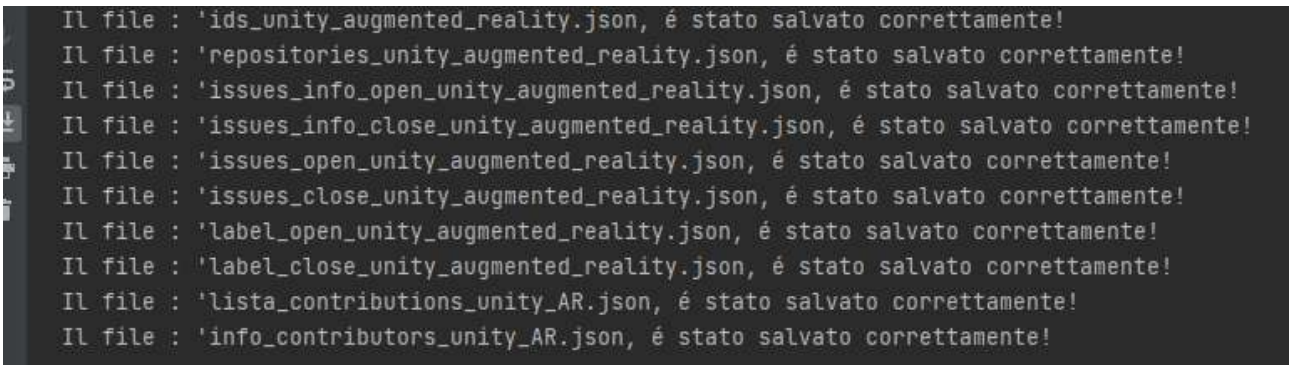

Figura 74 - Risultato script

Nell'output vengono mostrati dei messaggi che indicano la corretta creazione ed il corretto salvataggio dei file JSON.

10.3.4 Analisi delle issues

Nella seguente ricerca invece sono state sviluppate delle funzioni ad-hoc per poter permettere l'analisi delle issues, riguardanti i progetti sviluppati sulle piattaforme Vuforia, Arcore ed Arkit.

Le query che sono state utilizzate in esso sono quelle mostrate nei paragrafi precedenti ovvero quelle delle figure 72 e 73, per la ricerca delle informazioni sulle issues.

	ELEMENTO	DESCRIZIONE
NOME	ANALISI ISSUES	Questo script consente di poter eseguire delle funzioni che realizzano delle ricerche sulle issues aperte e chiuse dei repositories, utilizzando l'elenco dei repositories di realtà aumentato.
TIPOLOGIA	SCRIPT	
INPUT DATI	INFO	File json contenente le informazioni dei repositories di realtà aumentata
	ELENCO TAG SOFTWARE	Elenco dei tag inerenti ai software di realtà aumentata
	ELENCO TAG TOOLS	Elenco dei tag inerenti ai tool di realtà aumentata.
FUNZIONI USATE	RICERCA ISSUES	Funzione di ricerca delle informazioni sulle issues dei repositories, dato come input l'elenco dei nomi dei repositories
	RICERCA LABEL	Funzione di ricerca delle informazioni sulle label delle issues dei repositories, dato come input l'elenco dei nomi dei repositories
	SALVA REPOSITORIES TWO TAGS	Funzione di salvataggio delle informazioni ricercate, memorizzate in un apposito file json.
OUTPUT	INFORMAZIONI	Elenco di informazioni visibili tramite console inerenti ai repositories ricercati.
	FILE JSON	File Json creati contenenti le varie informazioni ricercate.
ESEMPIO D'USO	Analisi_issues.py (da eseguire da riga di comando)	

Tabella 17 - Informazioni script "analisi issues"

```
import json
import funzioni

# token github
ACCESS_TOKEN = '68d3043d0154715a436f34adba3f3c490bcb5443'

#TAG PER LA RICERCA DEI REPOSITORY DI GITHUB
sws = ['unity','unity3d']
tools = ['vuforia','arcore','arkit']

#LETTURA DA FILE JSON
for tool in tools:
    # LETTURA DEL FILE JSON CONTENENTE LE INFORMAZIONI DEI REPOSITORIES
    info = json.load(open("info_repositories_unity_" + tool + ".json"))
```

```
    # RICERCA DELLE ISSUES
    issues_info_open = funzioni.ricerca_issues(info, "aperte")
    issues_info_close = funzioni.ricerca_issues(info, "chiuse")
    funzioni.salva_repositories_two_tag(issues_info_open, "is-
sues_info_open", "unity", tool)
    funzioni.salva_repositories_two_tag(issues_info_close, "is-
sues_info_close", "unity", tool)

    # SALVATAGGIO DELLE ISSUES APERTE DEI REPOSITORIES
    issues_open = []
    issues_open = funzioni.ricerca_label(issues_info_open)
    funzioni.salva_repositories_two_tag(issues_open, "issues_open", "uni-
ty", tool)

    # SALVATAGGIO DELLE ISSUES CHIUSE DEI REPOSITORIES
    issues_close = []
    issues_close = funzioni.ricerca_label(issues_info_close)
    funzioni.salva_repositories_two_tag(issues_close, "issues_close", "uni-
ty", tool)
```

Figura 75 - Script analisi issues (pt.1)

```
#RICERCA DELLE ISSUES APERTE
issue_label_open = []
for progetto in issues_open:
    for issue in progetto:
        issue_label_open.append(issue["name"])

funzioni.salva_repositories_two_tag(issue_label_open, "label_open", "uni-
ty", tool)

#RICERCA DELLE ISSUES CHIUSE
issue_label_close = []
for progetto in issues_close:
    for issue in progetto:
        issue_label_close.append(issue["name"])

funzioni.salva_repositories_two_tag(issue_label_close, "label_close", "uni-
ty", tool)
```

Figura 76 - Script analisi issues (pt.2)

Pertanto, lo script precedentemente illustrato nelle figure 75 e 76, permette la ricerca delle informazioni sulle issues, mentre nella figura 77 è possibile vedere l'output dell'esecuzione dello script, che presenta dei messaggi di avvenuta creazione dei file JSON contenenti le informazioni che sono state poi utilizzate per l'analisi fatta in Matlab.

Figura 77 - Risultato dello script

10.4 Analisi con Matlab

Nel seguente paragrafo vengono descritti e mostrati gli script che sono stati sviluppati ed utilizzati per la creazione dei grafici utilizzati per l'analisi dei progetti di realità aumentata.

10.4.1 Analisi informazioni repositories

Per poter effettuare l'analisi dei dati ottenuti mediante le ricerche degli script Python, è stato necessario dover sviluppare delle funzioni che permettessero la lettura delle informazioni dai file JSON.

La prima funzione descritta è quella mostrata in figura 78, nella quale viene effettuata la lettura da file, per poi tramite una ulteriore funzione, mostrata nella figura 81, permettere il calcolo delle occorrenze, di quel determinato attributo che viene analizzato.

```
function analisi_json(tool)

if strcmp(tool,"augmented_reality")==0
    str = char(fread(fopen('info_repositories_unity_'+tool
+'.json'),inf)');
    fclose(fopen('info_repositories_unity_'+tool+'.json'));
    val = jsondecode(str);
else
    str = char(fread(fopen('repositories_unity_'+tool+'.json'),inf)');
    fclose(fopen('repositories_unity_'+tool+'.json'));
    val = jsondecode(str);
    val=val{1};
end

%INFORMAZIONI SUI LINGUAGGI
linguaggi = val(2);

linguaggi_tot = sort(string([linguaggi{1}]));
linguaggi=conta_occorrenze(length(linguaggi_tot),linguaggi_tot);
grafico(linguaggi,upper(tool),"ANALISI
 LINGUAGGI","Linguaggi","Progetti");

%ANALISI DATA CREAZIONE ED ULTIMO AGGIORNAMENTO DEI PROGETTI
data_creazione = val(4);

anni=["2016";"2015";"2014"];
data_creazione_tot = str2num(cell2mat(strtok ([data_creazione{1};...
    anni], '-' )));
media_creazione_tot = round(sum(data_creazione_tot)/length(...
    data_creazione_tot));
vettore=conta_occorrenze(length(data_creazione_tot),string(sort...
    (data_creazione_tot)));

data_aggiornamento = val(5);

data_aggiornamento_tot = sort(str2num(cell2mat(strtok (...
    [data_aggiornamento{1};anni], '-' ))));
media_aggiornamento_tot = round(sum(data_aggiornamento_tot)/...
    length(data_aggiornamento_tot));
vettore2=conta_occorrenze(length(data_aggiornamento_tot),string(sort...
    (data_aggiornamento_tot)));

grafico_doppio("ANNI DI PROGETTO",tool,vettore,vettore2);

end
```

Figura 78 - Funzione "analisi_json"

La funzione mostrata in figura 79, serve per l'analisi delle informazioni riguardanti i contributors, ovvero colo che hanno partecipato al progetto.

```
function analisi_contributors(tool)

if strcmp(tool,"augmented_reality")==1
    tool=string;
    tool="AR";
end

str = char(fread(fopen('lista_contributions_unity_'+tool
+'.json'),inf)');
fclose(fopen('lista_contributions_unity_'+tool+'.json'));
val = jsondecode(str);
val=sort(string(val));

vettore=conta_occorrenze(length(val),val);
vettore(1,:)=[];

totale=sum(double(vettore(:,2)));
perc=totale/100*1;
labels = [string zeros];
count=1;
for i=1:length(vettore(:,2))
    if double(vettore(i,2))>perc
        labels(count,1)=vettore(i,1);
        labels(count,2)=double(vettore(i,2));
        count = count +1;
    end
end

%PLOT DELLE INFORMAZIONI
figure;
x=categorical(labels(:,1));
y=double(labels(:,2))';
ymax=(find(y==max(y)));
explode = zeros(length(labels),1);
explode(ymax)=1;
pie3(y,explode);
label = {labels(:,1)}';
label= label{1};
lgd = legend(label);

title(lgd,'Legenda')
title({'NUMERO DI CONTRIBUTORS';'TAGS: UNITY e '+tool});


end
```

Figura 79 - Funzione "analisi_contributors"

La funzione mostrata in figura 80 serve per l'analisi delle informazioni riguardanti le issues.

```
function labels = analisi_issues(nome_file)

str = char(fread(fopen(nome_file),inf'));
fclose(fopen(nome_file));

info = sort(jsondecode(str));

label=string([]);
count=0;
for i=1:length(info)-1
    if strcmp(info(i),info(i+1))==0
        count=count+1;
        label(count)=info(i);
    end
end
dim_type = length(label);
dim_info = length(info);
type = label';

info = string(info);
vettore = [type zeros(length(type),1)];
for i = 1 : dim_type
    count = 0;
    for j = 1: dim_info
        if strcmp(info(j),type(i))==1
            count = count +1;
            vettore(i,2)=count;
        end
    end
end

% prendo soltanto i valori che superano almeno il 4% delle occorrenze
% totali
totale=sum(double(vettore(:,2)));
perc=totale/100*4;
labels = [string zeros];
count=1;
for i=1:length(vettore(:,2))
    if double(vettore(i,2))>perc
        labels(count,1)=vettore(i,1);
        labels(count,2)=double(vettore(i,2));
        count = count +1;
    end
end
```

Figura 80 - Funzione "analisi_issues"

```
function vettore=conta_occorrenze(dim_info, info)

%CALCOLO DELLE TIPOLOGIE
type = string([]);
count=1;
type(1)= info(1);
for i=1:length(info)-1
    if strcmp(info(i),info(i+1))==0
        count=count+1;
        type(count)=info(i+1);
    end
end

% CALCOLO EFFETTIVO DELLE OCCORRENZE
vettore = [type' zeros(length(type),1)];
for i = 1 : length(type)
    count = 0;
    for j = 1: dim_info
        if strcmp(info(j),type(i))==1
            count = count +1;
            vettore(i,2)=count;
        end
    end
end



end
```

Figura 81 - Funzione "conta_occorrenze"

10.4.2 Funzioni per la creazione dei grafici

In questa sezione vengono mostrate tre funzioni che permettono la creazione dei grafici in modo da poter rappresentare i risultati ottenuti.

```
function grafico(vettore, tools, title_name, x_name, y_name)

%PLOT DELLE INFORMAZIONI
figure;
hold on;
grid MINOR;
title({title_name;'TAGS: UNITY e '+tools});
ylabel(y_name),xlabel(x_name);
x=categorical(vettore(:,1));
y=double(vettore(:,2))';
b=bar(x,y,0.4);
labels1 = string(b(1).YData);
text(b.XData,b.YData,labels1,'VerticalAlignment','bottom',...
    'HorizontalAlignment','center')
b.FaceColor = 'flat';
b.CData(find(y==max(y)),:) = [.5 0 .5];

end
```

Figura 82 - Funzione "grafico"

Le tre funzioni si differenziano a seconda del tipo di grafico che serve, ossia se deve essere singolo (figura 82), oppure doppio (figura 83), il quale presenta una grafica divisa a metà con la contrapposizione delle due rappresentazioni.

```
function grafico_doppio(titolo,tools,issues_open,issues_close)

ymax_open=max(double(issues_open(:,2)));
ymax_close=max(double(issues_close(:,2)));
perc=max(ymax_open,ymax_close);

figure;
subplot(1,2,1);
title({'ANALISI '+titolo;' TAGS: UNITY e '+upper(tools)});
hold on; grid MINOR;
ylabel('Issues Aperte'),xlabel('Labels');
b=bar(categorical(issues_open(:,1)),double(issues_open(:,2)),0.4);
text(b.XData,b.YData,string(b(1).YData),'VerticalAlignment','bottom',...
    'HorizontalAlignment','center')
b.FaceColor = 'flat';
ymax=find(double(issues_open(:,2))==max(double(issues_open(:,2))));
for i=1:length(ymax)
    b.CData(ymax(i),:) = [.5 0 .5];
end

ymin=0;
ymax=perc+10;
ax=gca;
ax.YLim=[ymin ymax];
set(gca,'Ytick',[ymin:10:ymax])

subplot(1,2,2);
hold on; grid MINOR;
ylabel('Issues Chiuse'),xlabel('Labels');
b=bar(categorical(issues_close(:,1)),double(issues_close(:,2))',0.4);
text(b.XData,b.YData,string(b(1).YData),'VerticalAlignment','bottom',...
    'HorizontalAlignment','center')
b.FaceColor = 'flat';
ymax=find(double(issues_close(:,2))==max(double(issues_close(:,2))));
for i=1:length(ymax)
    b.CData(ymax(i),:) = [.5 0 .5];
end

ymin=0;
ymax=perc+10;
ax=gca;
ax.YLim=[ymin ymax];
set(gca,'Ytick',[ymin:10:ymax])
end
```

Figura 83 - Funzione "grafico_doppio"

Infine, abbiamo il grafico "a torta" in figura 84.

```
function grafico_torta(labels,titolo,tool)

%PLOT DELLE INFORMAZIONI
figure;
x=categorical(labels(:,1));
y=double(labels(:,2))';
ymax=(find(y==max(y)));
explode = zeros(length(labels),1);
explode(ymax)=1;
pie3(y,explode);
label = {labels(:,1)}';
label= label{1};
lgd = legend(label);

title(lgd,'Legenda')
title({titolo;'TAGS: UNITY e '+tool});
end
```

Figura 84 - Funzione "grafico_torta"

10.4.3 Main

Lo script che permette l'esecuzione di queste funzioni è mostrato nella figura 85 sottostante, il quale permette la realizzazione dei grafici per i file JSON collegati ai quattro tag in questione da analizzare.

```
clc; clear all; close all;

tags = ["vuforia";"arcore";"arkit";"augmented_reality"];

for i=1:length(tags)
    %LETTURA FILE JSON ED ANALISI DEI DATI LETTI
    analisi_json(tags(i));

    %ANALISI E RAPPRESENTAZIONE GRAFICA ISSUES
    issues_open = analisi_issues("label_open_unity_"+tags(i)+".json");
    issues_close =
 analisi_issues("label_close_unity_"+tags(i)+".json");
    grafico_doppio("LABEL",tags(i),issues_open,issues_close);

    %ANALISI E RAPPRESENTAZIONE DEI CONTRIBUTORS
    analisi_contributors(tags(i));
end
```

Figura 85 - Funzione "Main"

10.5 Analisi statistica dei progetti

A questo punto è possibile analizzare i progetti, recuperati tramite l'esecuzione degli script Python.

10.5.1 Progresso temporale

Innanzitutto, si è proceduto all'analisi temporale dei progetti, in modo da capire tendenzialmente negli anni come gli sviluppatori di applicazioni di realtà aumentata si sono dedicati a questi progetti.

Come è possibile vedere nei grafici successivi (figura 86, 87, 88 e 89), il trend degli ultimi anni è crescente, ovvero sempre più sviluppatori creano nuove applicazioni di realtà aumentata.

Infatti, per ogni immagine vi sono due grafici, quello di sinistra che indicano le date di inizio dei progetti esaminati, mentre quello di destra rappresenta gli anni in cui sono stati fatti gli ultimi aggiornamenti per ogni progetto.

Le colonnine in blu, indicano il numero di progetti iniziati in quel particolare anno, mentre la colonnina di colore viola, indica l'anno che presenta maggiori progetti tra tutti gli anni.

Per quanto riguarda i grafici "alla destra" ovviamente si tratta dell'ultima data in cui è stato apportato un aggiornamento al progetto, il che non rappresenta necessariamente una informazione sicura al 100% che quel determinato progetto sia stato abbandonato, ma può altresì indicare che sono terminati gli aggiornamenti, per quel progetto, sempre in quel determinato anno.

È inoltre possibile notare come i primissimi progetti siano stati sviluppati negli anni tra il 2014 ed il 2015, il che rappresenta la giovanissima età di questa tecnologia.

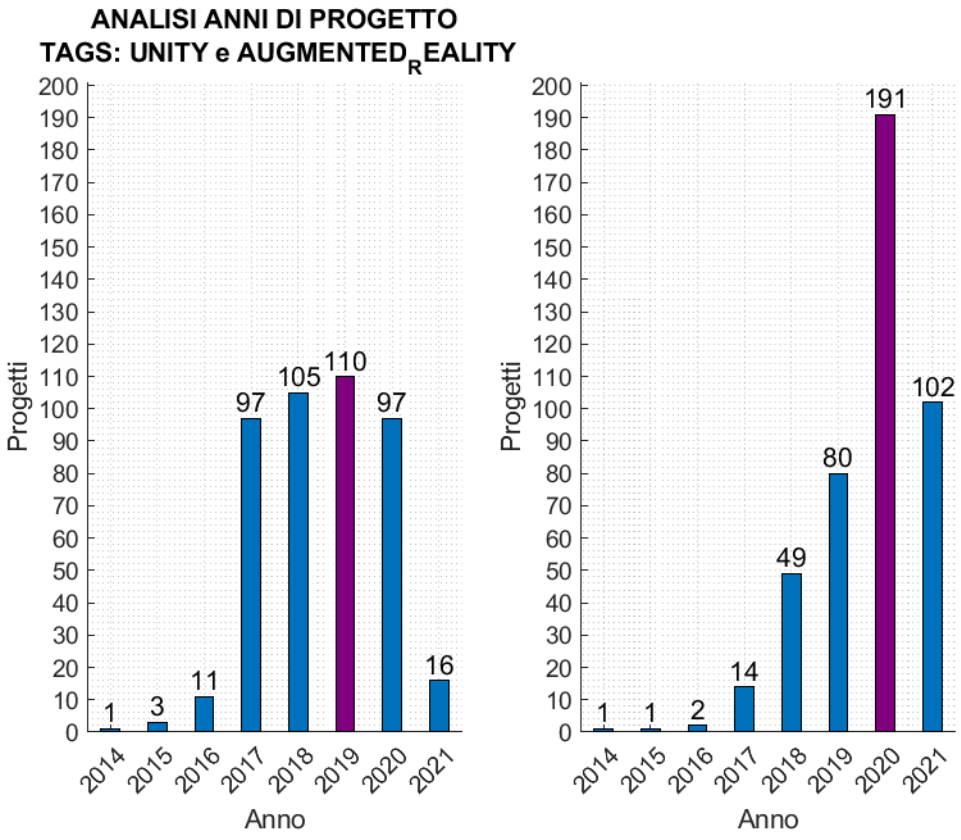

Figura 86 - Anni di progetto "Augmented Reality"

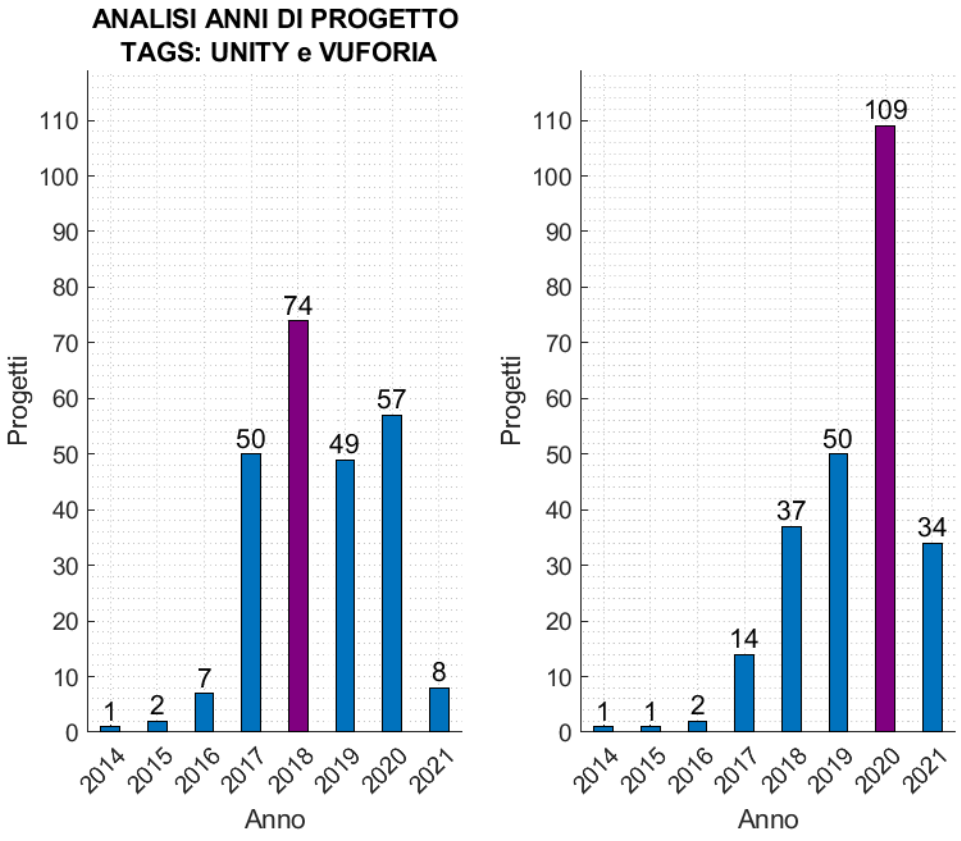

Figura 87 - Anni di progetto "Vuforia"

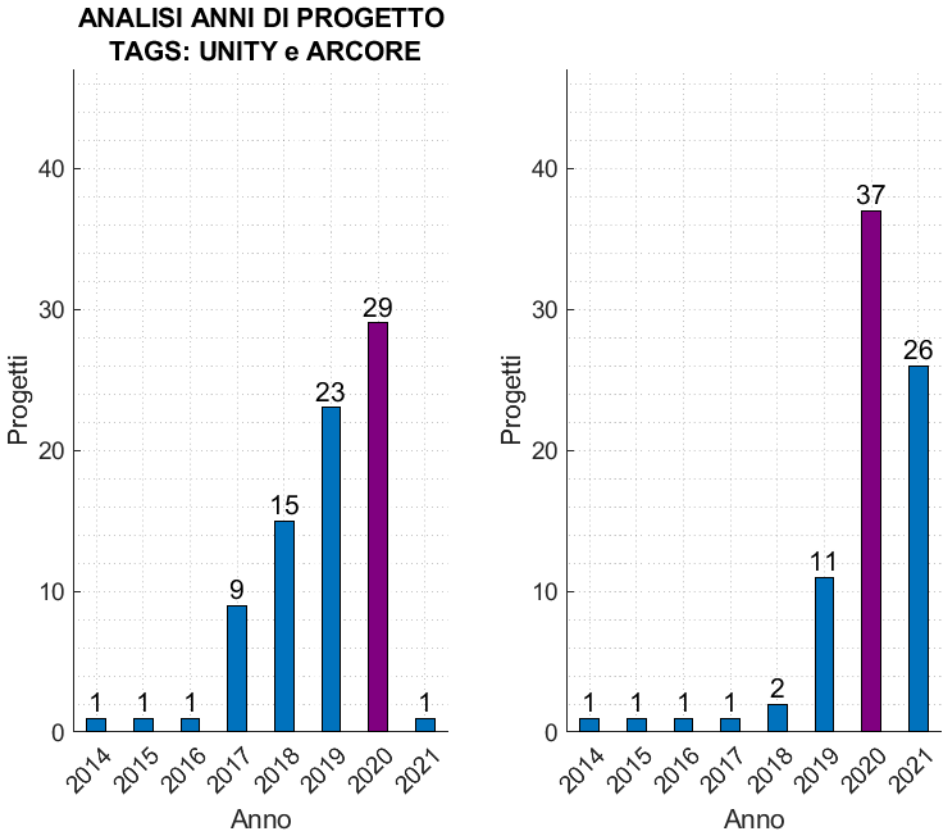

Figura 88 - Anni di progetto "Arcore"

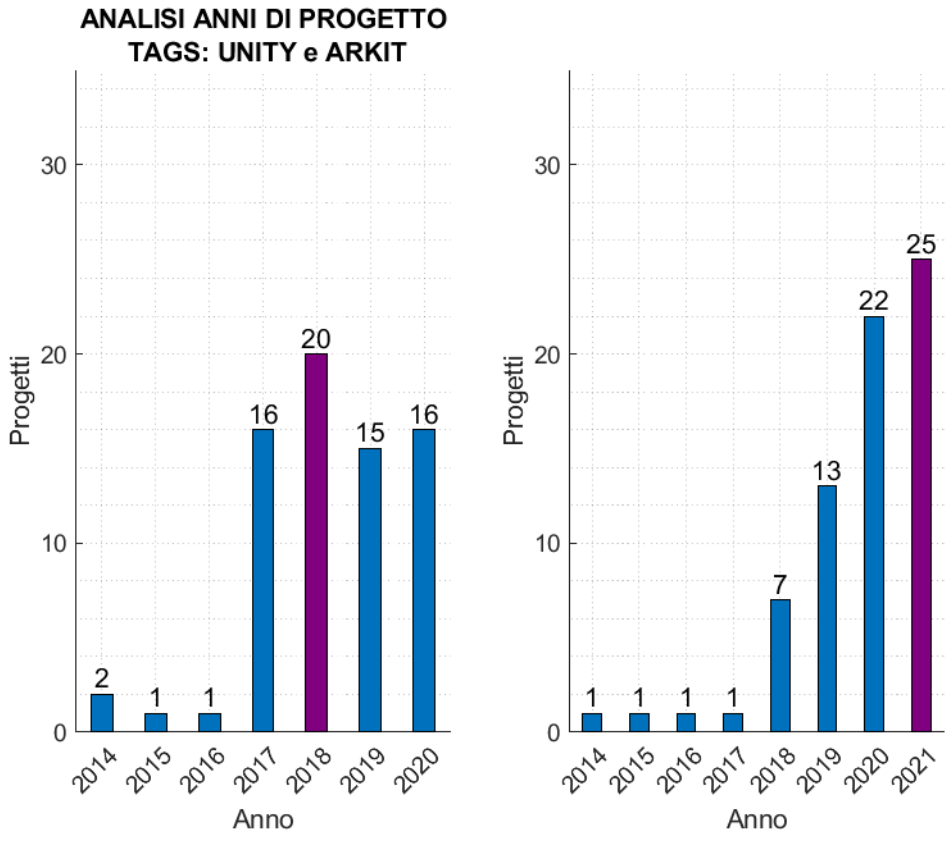

Figura 89 - Anni di progetto "Arkit"

Il boom vero e proprio di inizio di progettazione risale al 2017, in cui è possibile notare un vero e proprio picco rispetto agli anni precedenti.

Tuttavia, il numero dei progetti è destinato sempre più a crescere negli anni a venire, infatti vi è una crescita abbastanza evidente del numero di progetti iniziati negli anni dal 2017 al 2020.

Ovviamente per quanto riguarda i numeri riguardanti il 2021 sia per i progetti iniziati che per i progetti "terminati" il numero è ancora in corso di aggiornamento.

È inoltre possibile notare come Arcore (figura 87) presenti più progetti sviluppati rispetto ad Arkit, differenza che si ha probabilmente grazie al suo ampio piano di sviluppo, proprio perché offre più piattaforme a disposizione su cui poter progettare applicazioni, sia per sistemi Android che per sistemi IOS).

Presenta quindi un trend che è nettamente crescente col passare degli anni, mentre invece Arkit essendo legato a dispositivi IOS, l'andamento è rimasto costante negli anni (figura 88).

Altra informazione interessante che riguarda il tempo di sviluppo e di dedizione per un progetto, e come riepilogato nella seguente tabella 5, tendenzialmente i progetti difficilmente superano i due anni di lavoro:

ANNI	#	PERC
1	303	66,45
2	116	25,44
3 o più	37	8,11

Tabella 18 - Anni di produzione

Infatti, è possibile notare che nel 66,5% dei casi analizzati i progetti sono stati abbandonati/terminati dopo un solo anno di lavoro, mentre nel 25,5% dei casi sono stati progetti protratti in 2 anni.

Solo nell'8% dei casi invece gli anni di lavoro sono stati 3 o più.

10.5.2 Componenti dei Team di sviluppo

Altro studio che è stato condotto riguarda il numero di componenti nei team degli sviluppatori che hanno partecipato ai commit dei progetti visionati su GitHub.

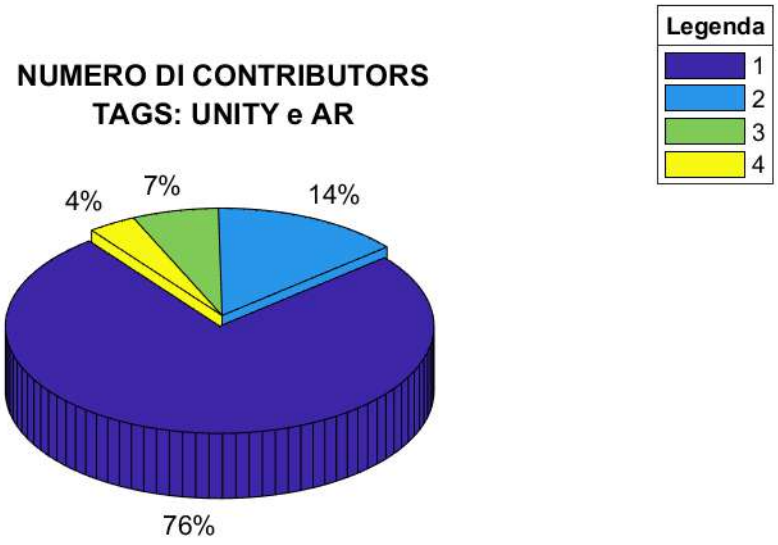

Figura 90 - Numero contributors Augmented Reality

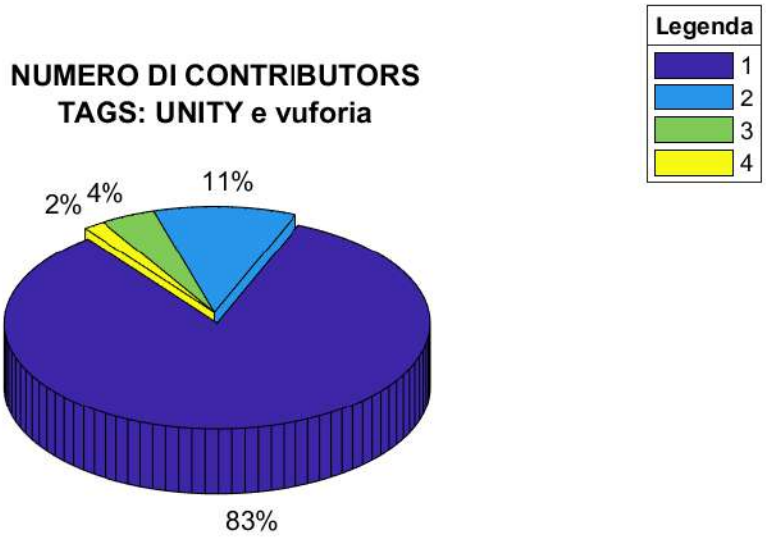

Figura 91 - Numero contributors Vuforia

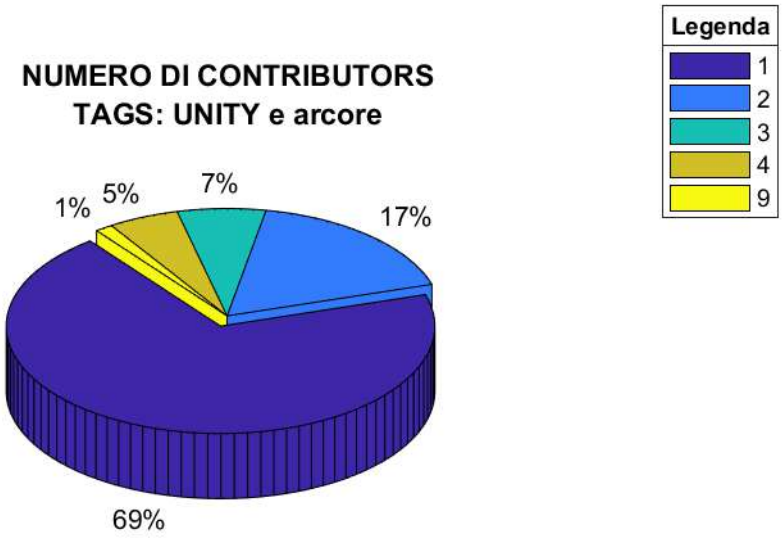

Figura 92 - Numero contributors Arcore

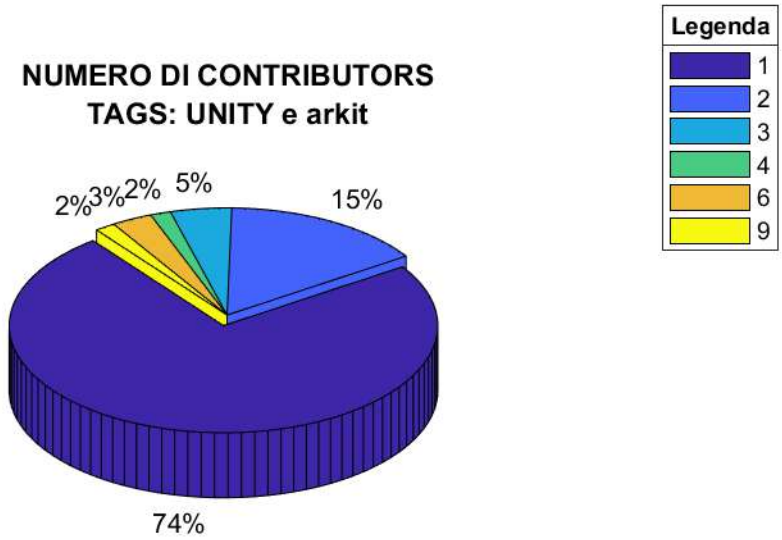

Figura 93 - Numero contributors Arkit

Nei seguenti grafici (figura 90, 91, 92 e 93) è possibile notare come in linea del tutto generale solitamente queste applicazioni vengono sviluppate da un solo componente, con una percentuale media che supera il 75% dei casi.

TEAM	*#*	*PERC*
1	353	77,58
2	72	15,82
3 o più	30	6,59

Mentre nel 16% circa si lavora in coppia, e in poco più del 6,5% in team di tre o più elementi.

10.5.3 Linguaggi di programmazioni utilizzati

In questa sezione vengono mostrati i risultati inerenti ai linguaggi di programmazione che sono stati maggiormente utilizzati all'interno dei progetti analizzati.

Nelle figure sottostanti (figura 94, 95, 96 e 97) vengono mostrati i numeri statistici di occorrenze dei linguaggi utilizzati per ogni categoria di studio analizzata, e come è possibile vedere in tutti i casi quello prevalente è sempre il C#.

Questo dato è confermato dal perché Unity come piattaforma prevede l'utilizzo di linguaggi diversi: C# e Javascript.

Chiaramente è possibile creare combinazioni complesse di script che interagiscono fra loro, però alla base di tutto c'è sempre un Game Object in scena che chiama uno (o più) script.

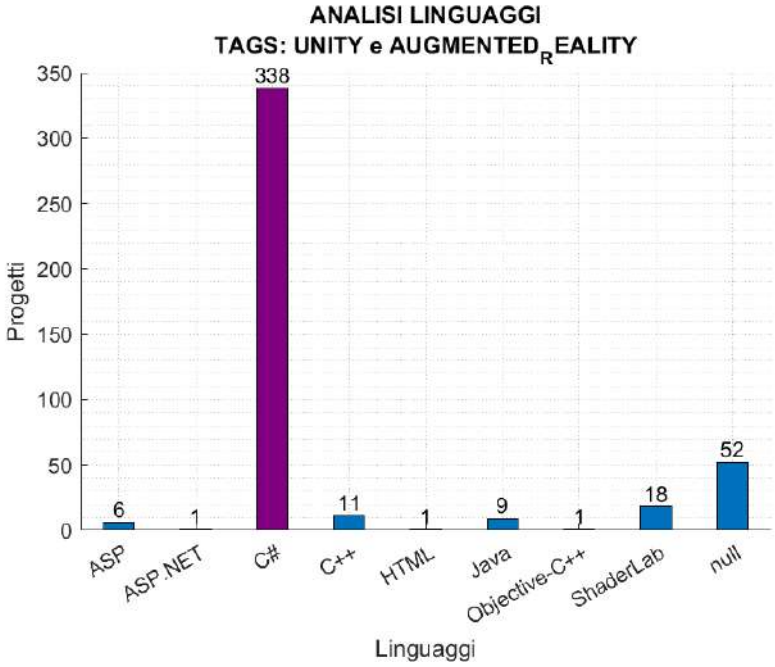

Figura 94 - Analisi linguaggi Augmented Reality

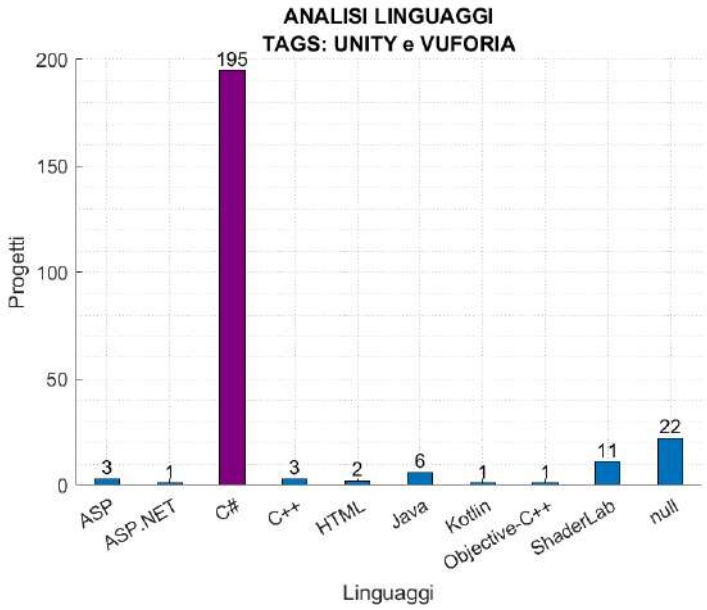

Figura 95 - Analisi linguaggi Vuforia

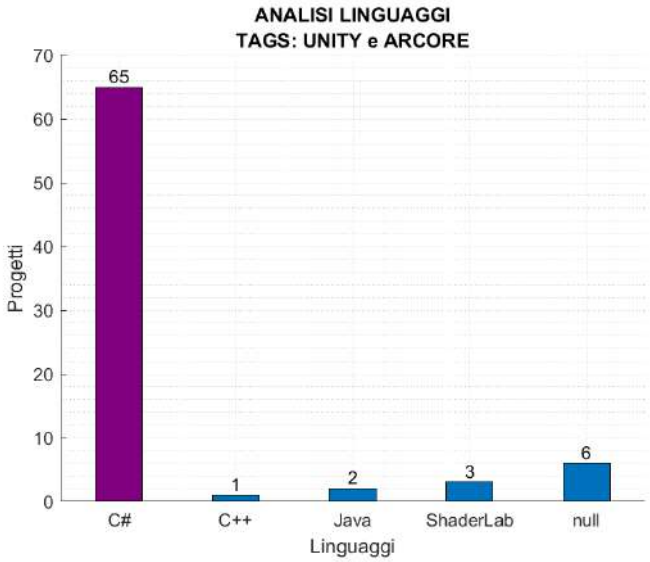

Figura 96 - Analisi linguaggi Arcore

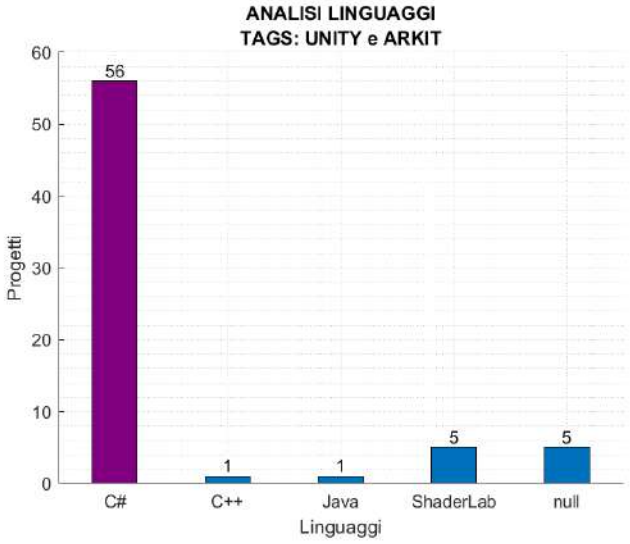

Figura 97 - Analisi linguaggi Arkit

Nella tabella sottostante sono riassunti brevemente le percentuali sui linguaggi più utilizzati dai progetti che sono stati analizzati, quello più presente è il C#, che raggiunge una percentuale di copertura delle applicazioni di quasi l'80%.

LINGUAGGIO	%
C#	79,1
ShaderLab	4,5
Java	2,2
C++	1,9
ALTRO	**12,3**

Tabella 19 - Occorrenze linguaggi

10.5.4 Analisi delle issues

In questa sezione vengono analizzate le issues dei progetti, sia quelle aperte che quelle che sono state chiuse/risolte.

Su Github le issues presentano delle etichette che ne indicano il genere di appartenenza, in modo da poter essere meglio catalogate e specificate quando devono essere trattate, e argomentate.

Nella tabella seguente (tabella 20) indica il numero di etichette per categoria nelle issues analizzate.

TAGS	#	%
ENHANCEMENT	398	29,8
BUG	280	21
QUESTION	70	5,2
SECURITY	65	4,9
FEATURE	64	4,8
DOCUMENTATION	39	2,9
ALTRO	***420***	***31,4***

Tabella 20 - Occorrenze issues label

Tra le più presenti sono quelle inerenti agli Enhancement con una percentuale di presenza del quasi 30%, e subito dopo al secondo posto delle issues più

frequenti sono quelle dei Bug, con una percentuale del 21%, queste in linea generale indicano tutte le segnalazioni dei malfunzionamenti che vengono trovati nel provare l'applicazione, dagli utenti che la provano.
Nelle successive figure (figura 98, 99, 100 e 101) vengono raffigurati per ogni tool.

Nei grafici alla sinistra vi sono le issues aperte, mentre in quelle di destra quelle chiuse, e come per i grafici precedentemente illustrati, la colonna in viola indica il valore massimo di occorrenza tra quelli presenti.

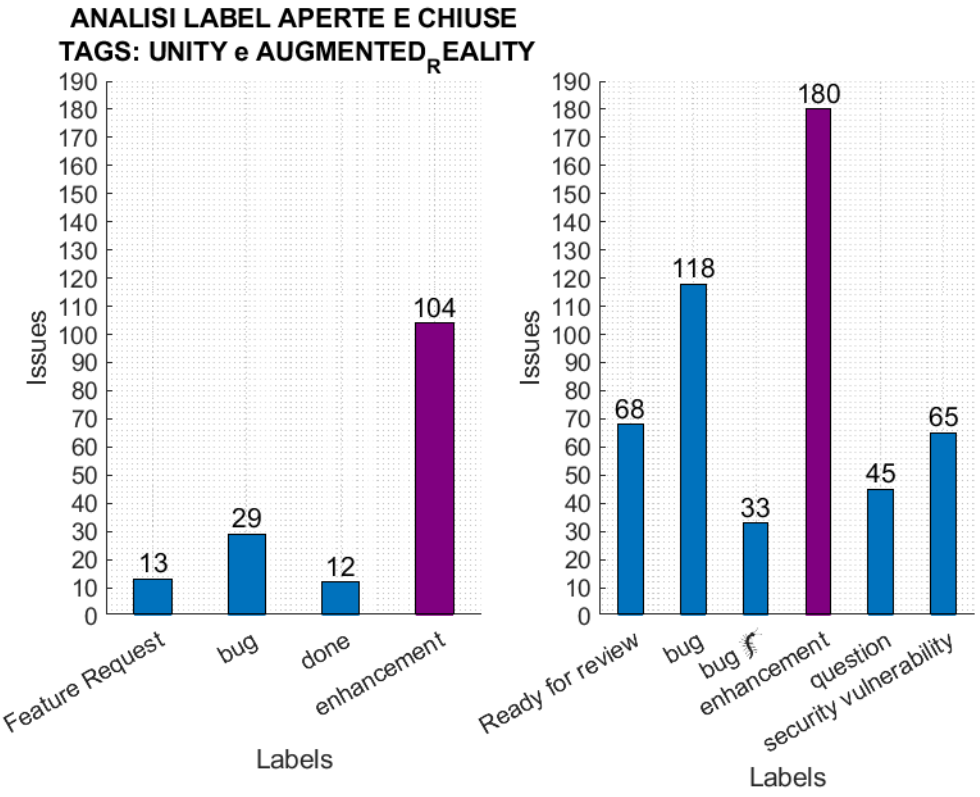

Figura 98 - Analisi issues Augmented reality

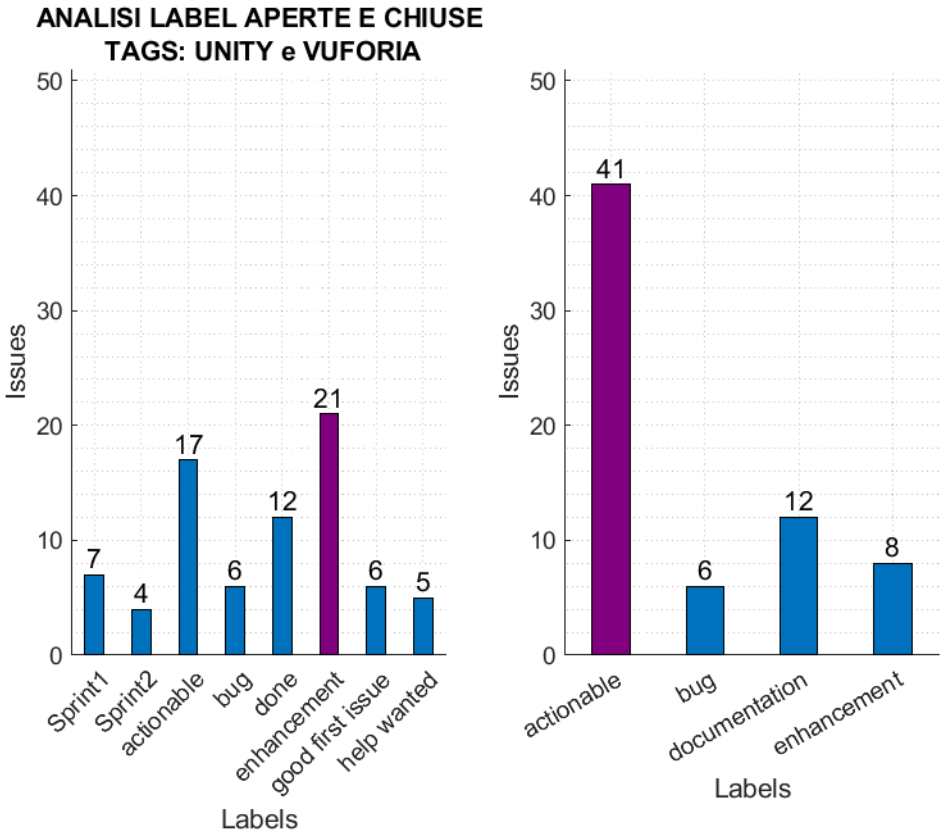

Figura 99 - Analisi issues Vuforia

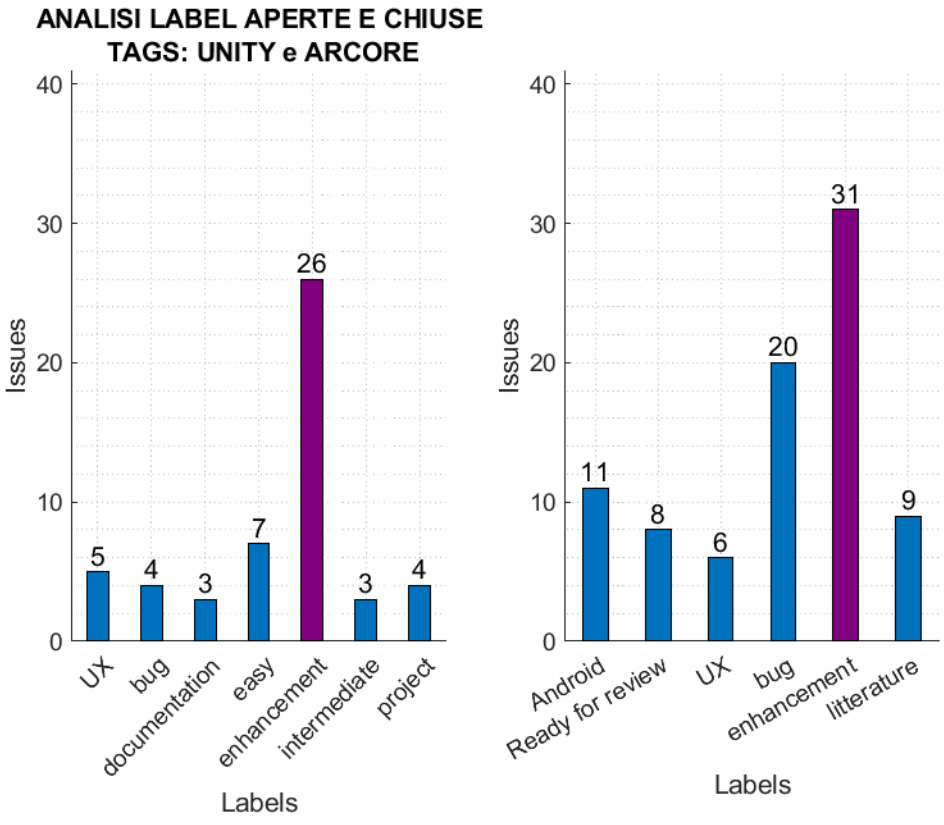

Figura 100 - Analisi issues Arcore

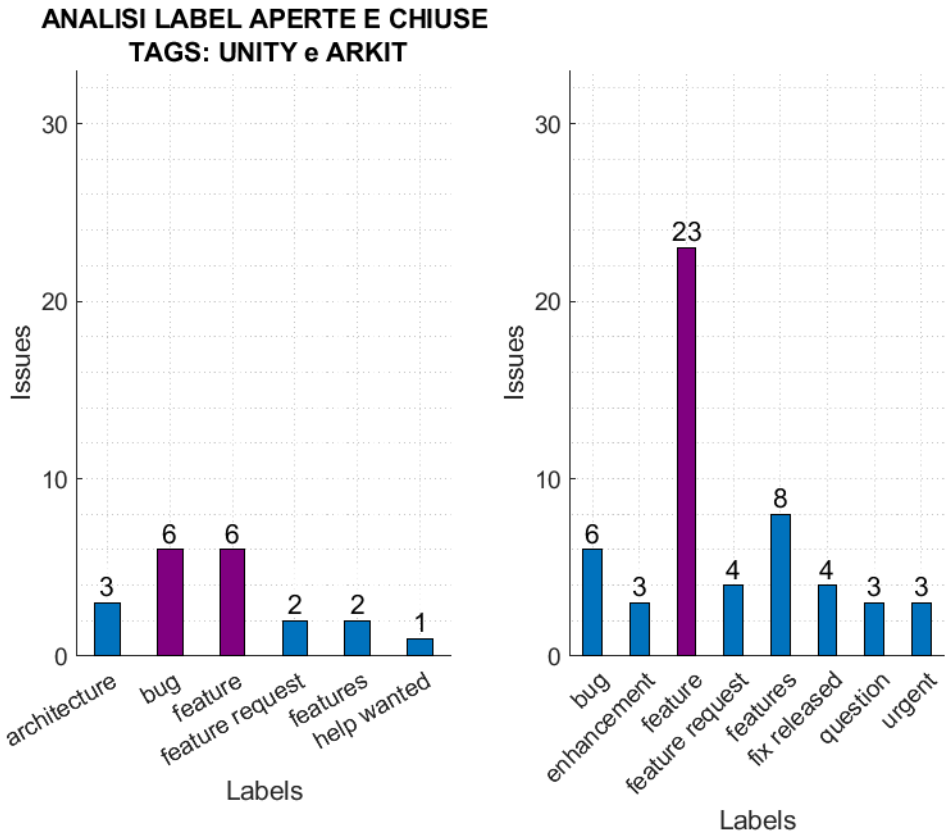

Figura 101 - Analisi issues Arkit

Capitolo 11: Ricerca Test

Arrivati a questo punto dell'analisi dei progetti open-source messi a disposizione da GitHub, il focus di questa, si concentrerà quindi sul contenuto del codice di questi repositories di realtà aumentata trovati, in modo da cercare di capire, sempre attraverso l'utilizzo di scripts e query nelle Api Github, se sono presenti ed in che quantità dei Test.

11.1 Ricerca dell'elenco dei repositories

Innanzitutto, il primo passaggio che è stato eseguito, è stato quello di avere a disposizione l'elenco completo di tutti i repositories trovati sinora, in modo da poter effettuare questa ulteriore ricerca.

Pertanto, per una questione di comodità è stato sviluppato uno script che consentisse di avere l'elenco completo dei progetti che si vuole analizzare.

Lo script utilizzato è il seguente:

```
import funzioni

# token github
ACCESS_TOKEN = '346adc523f1c40b2ea31d27f9fdffef647129821'

#TAG PER LA RICERCA DEI REPOSITORY DI GITHUB
sws = ['unity','unity3d']
types = ['augmentedreality','augmented-reality','ar']
tag = "augmented_reality"
```

Figura 102 - Script per la ricerca dell'elenco dei repositories (pt.1)

```
# ricerca dei repositories
lista_info = []
```

```
for type in types:
    info= funzioni.ricerca_info(sws[0], type, funzio-
ni.search_repositories_two_tags(sws[0], type))
    info_3d= funzioni.ricerca_info(sws[1], type, funzio-
ni.search_repositories_two_tags(sws[1], type))
    lista_info.append(funzioni.controllo_progetti_doppi(info, info_3d))
    #print(lista_info)

info_tot = funzioni.controllo_progetti_doppi(lista_info[0], lista_info[1])

#creazione e salvataggio dell'elenco degli id dei progetti, con eliminazio-
ne dei progetti doppi
lista_id = funzioni.controllo_progetti_doppi(lista_info[2], info_tot)
funzioni.salva_repositories_two_tag(lista_id, "lista_ids", "unity", tag)
```

Figura 103 - Script per la ricerca dell'elenco dei repositories (pt.2)

	ELEMENTO	***DESCRIZIONE***
NOME	RICERCA LISTA PROGETTI	Questo script consente di ottenere l'elenco dei nomi dei repositories di realtà aumentata.
TIPOLOGIA	SCRIPT	
INPUT DATI	TAG	Tag utilizzato per i salvataggi ovvero, "augmented_reality"
	ELENCO TAG SOFTWARE	Elenco dei tag inerenti ai software di realtà aumentata
	ELENCO TAG TOOLS	Elenco dei tag inerenti ai tool di realtà aumentata.
FUNZIONI USATE	RICERCA INFO	Funzione di ricerca delle informazioni inerenti ai repositories
	CONTROLLO PROGETTI DOPPI	Funzione di controllo per verificare se le informazioni doppie all'interno di elenchi di informazioni sui repositories
	SALVA REPOSITORIES TWO TAGS	Funzione di salvataggio delle informazioni ricercate, memorizzate in un apposito file json.
OUTPUT	INFORMAZIONI	Elenco di informazioni visibili tramite console inerenti ai repositories ricercati.
	FILE JSON	File Json creati contenenti le varie informazioni ricercate.
ESEMPIO D'USO	Ricerca_lista_progetti.py (da eseguire da riga di comando)	

Tabella 21 - Informazioni script "ricerca lista progetti"

Nella figura (figura 103) seguente è possibile vedere un estratto della composizione del file Json generato, ove sono presenti i primi 30 repositories trovati:


```
[
    [
        "andrewnakas/OpenBrushVR",
        "vimeo/vimeo-unity-sdk",
        "ronellsicat/DxR",
        "andrewnakas/ARKit-Cardboard-VR",
        "yasirkula/UnityIonicIntegration",
        "marlon360/covid-ar",
        "namanrajpal/FurnitureAR",
        "tomzorz/awesome-mixedreality",
        "aidawhale/AnimalRoadtrip-App",
        "aitc19x/VR-JC",
        "nheidloff/unity-watson-ar-sample",
        "dioram/Elektronik-Tools-2.0",
        "LightningArtist/latkUnity",
        "borgmon/ArKitKat",
        "ARtourism/NearSight",
        "yangfawen/RAVVAR-XunAPI",
        "Dinghow/Misaki_AR",
        "namankhurpia/Spark-AR-filters",
        "scians-team/TU7FA",
        "North-Star-Research-Institute/North-Star-Unity-Starter-Project",
        "liubq7/FoodX",
        "xuanll/Resume",
        "oliguo/Android-Unity-Vuforia",
        "JohJakob/Bauhaus-Augmented",
        "hu243285237/KnightAR_Demo",
        "arepina/leapMotionKeyboard",
        "coernel360/uwe_phd_hat",
        "danielfranze/HighscoreAR",
```


Figura 104 - File Json contenente i nomi dei repositories

Una volta ottenuto l'elenco dei repositories, si è provveduto, attraverso la progettazione di query ad hoc, alla ricerca interna di ogni repository, della parola chiave "Test".

11.2 Ricerca della parola chiave

Il passaggio descritto nel paragrafo precedente ha dato la possibilità di poter trovare tutto ciò che potesse essere in qualche modo collegato ad un Test. L'obiettivo è quello di cercare eventualmente dei Tests eseguiti all'interno

delle applicazioni di realtà aumentata, o meglio ancora vedere successivamente in modo più dettagliato.

Cosa importante è quella di capire se gli eventuali Tests erano stati scritti e pensati direttamente dal progettista in questione, oppure cercare di capire se c'è una possibilità che questi vengano creati in modo automatico da qualche software.

Per poter effettuare questa ricerca è stata quindi sviluppata una query che, dato il repository, permettesse di cercare all'interno di esso la parola chiave desiderata, ovvero "Test".

Come in altri script precedentemente descritti, anche qui per una questione di comodità è stata progettata una ricerca, che dato in input il file Json contenente l'elenco dei repositories, desse in output il frutto delle ricerche all'interno di esso.

	ELEMENTO	***DESCRIZIONE***
NOME	RICERCA NEL CODICE PER PAROLA CHIAVE	Questa funzione permette di poter effettuare delle ricerche dei file che presentano al loro interno la parola chiave usata come input, per ogni repository.
TIPOLOGIA	FUNZIONE	
INPUT	LISTA ID	Elenco dei nomi dei progetti in cui ricercare.
	PAROLA CHIAVE	Termine da usare per la ricerca nella query
OUTPUT	ELENCO URL	Lista contenente tutti i link dei file che presentano al loro interno la parola chiave passata come input nella query.
	ELENCO INFO	Lista contenente tutte le informazioni sui file trovati all'interno di ogni repository.
ESEMPIO D'USO	Funzioni.ricerca_nel_codice_per_parola_chiave(lista_id,parola_chiave)	

Tabella 22 - Informazioni funzione "ricerca nel codice per parola chiave"

Questa funzione preleva tramite le Api di Github i link di indirizzamento a tali file, in modo da avere un elenco completo di tutti i possibili test, così da rendere la ricerca ed il controllo finale molto più semplice ed efficiente.

La funzione, che è stata sviluppata ed utilizzata nell'esecuzione dello script per effettuare questa ricerca attraverso la parola chiave "Test", viene mostrata nella successiva immagine (figura 105).

```
def ricerca_nel_codice_per_parola_chiave(lista_id,parola_chiave):
    elenco_info=[]
    elenco_url=[]
    i=0
    count =0
    for id in lista_id:
        if count==30:
            i=i+1
            count=0
            print("Pausa di 60 secondi, per poter effettuare altre richie-
ste. Mancano ancora "+str(len(lista_id)-i*30)+
                  " progetti.")
            time.sleep(60)
        #query per la ricerca delle parole chiave nel codice
        test = reque-
sts.get('https://api.github.com/search/code?q='+parola_chiave+'+in%3afile+r
epo%3a' + id +
                           '&per_page=100', auth=('enzot95',
ACCESS_TOKEN)).json()

        # info del progetto che contine la parola chiave
        if len(test["items"])!=0:
            num=test["total_count"]
            elenco_info.append(num)
            if num>0:
                elen-
co_url.append(test["items"][0]["repository"]["html_url"])

        count = count + 1

    return elenco_url, elenco_info
```

Figura 105 - Funzione "ricerca_nel_codice_per_parola_chiave"

La difficoltà riscontrata inizialmente nell'esecuzione di anche di questa funzione è stato il limite temporale, di 30 richieste al minuto, da parte di Github.

Pertanto, per poter fare in modo di eseguire tutte le ricerche senza interruzione, è stato inserito uno "sleep" di 60 secondi, ogni 30 repositories analizzati, in modo da poter permettere il corretto utilizzo della query, ed ottenere l'output in modo corretto.

	ELEMENTO	***DESCRIZIONE***
NOME	RICERCA TEST	Questo script consente di ottenere l'elenco dei file che presentano al loro interno probabili test, dentro i repositories di realtà aumentata.
TIPOLOGIA	SCRIPT	
INPUT DATI	LISTA ID	Elenco dei nomi dei progetti in cui ricercare.
	PAROLA CHIAVE	Termine da usare per la ricerca nella query
FUNZIONI USATE	RICERCA NEL CODICE PER PAROLA CHIAVE	Funzione di ricerca delle informazioni inerenti ai file di test nei repositories
	SALVA REPOSITORIES TWO TAGS	Funzione di salvataggio delle informazioni ricercate, memorizzate in un apposito file json.
OUTPUT	INFORMAZIONI	Elenco di informazioni visibili tramite console inerenti ai repositories ricercati.
	FILE JSON	File Json creati contenenti le varie informazioni ricercate.
ESEMPIO D'USO	Ricerca_test.py (da eseguire da riga di comando)	

Tabella 23 - Informazioni script "ricerca test"

Lo script con la funzione, che permette di poter effettuare la ricerca sopracitata, è il seguente:

```
import json
import funzioni
import time

ACCESS_TOKEN = 'af314d573e86e8780c88bf445e33584567bf120a'

parola_chiave='test'
lista_id = json.load(open("../lista_ids_unity_augmented_reality.json"))

# Questo script serve per fare una ricerca all'interno dei repositories se-
```

```
lezionati di realtà aumentata
# per vedere se sono presenti alcune parole chiave, che potrebbero essere
ricollegate ad dei test fatti
# all'interno dei progetti.

elenco_url,elenco_info= funzio-
ni.ricerca_nel_codice_per_parola_chiave(lista_id[0], parola_chiave)
funzioni.salva_repositories_two_tag(elenco_info, "elenco_info_" + paro-
la_chiave, "unity", "augmentedreality")
funzioni.salva_repositories_two_tag(elenco_url, "elenco_url_" + paro-
la_chiave, "unity", "augmentedreality")
print("L'elenco dei progetti di realità aumentata da visionare sono
"+str(len(lista_id[0])))
print("Sono stati trovati "+str(len(elenco_info))+" che presentano la paro-
la chiave <<Test>>")
print("Pausa per fine ricerca nei repositories della parola chiave:
"+parola_chiave)
time.sleep(60)
```

Figura 106 - Script Ricerca della parola chiave

Attraverso lo script seguente, è stato quindi possibile trovare l'elenco dei repositories:

```
Pausa di 60 secondi, per poter effettuare altre richieste. Mancano ancora 476 progetti.
Pausa di 60 secondi, per poter effettuare altre richieste. Mancano ancora 446 progetti.
Pausa di 60 secondi, per poter effettuare altre richieste. Mancano ancora 416 progetti.
Pausa di 60 secondi, per poter effettuare altre richieste. Mancano ancora 386 progetti.
Pausa di 60 secondi, per poter effettuare altre richieste. Mancano ancora 356 progetti.
Pausa di 60 secondi, per poter effettuare altre richieste. Mancano ancora 326 progetti.
Pausa di 60 secondi, per poter effettuare altre richieste. Mancano ancora 296 progetti.
Pausa di 60 secondi, per poter effettuare altre richieste. Mancano ancora 266 progetti.
Pausa di 60 secondi, per poter effettuare altre richieste. Mancano ancora 236 progetti.
Pausa di 60 secondi, per poter effettuare altre richieste. Mancano ancora 206 progetti.
Pausa di 60 secondi, per poter effettuare altre richieste. Mancano ancora 176 progetti.
Pausa di 60 secondi, per poter effettuare altre richieste. Mancano ancora 146 progetti.
Pausa di 60 secondi, per poter effettuare altre richieste. Mancano ancora 116 progetti.
Pausa di 60 secondi, per poter effettuare altre richieste. Mancano ancora 86 progetti.
Pausa di 60 secondi, per poter effettuare altre richieste. Mancano ancora 56 progetti.
Pausa di 60 secondi, per poter effettuare altre richieste. Mancano ancora 26 progetti.
 Il file : 'elenco_info_test_unity_augmentedreality.json, é stato salvato correttamente!
 Il file : 'elenco_url_test_unity_augmentedreality.json, é stato salvato correttamente!
L'elenco dei progetti di realità aumentata da visionare sono 506
Sono stati trovati 221 che presentano la parola chiave <<Test>>
Pausa per fine ricerca nei repositories della parola chiave: test
```

Figura 107 - Output dell'esecuzione dello script

Come viene mostrato nel seguente output, vi sono dei messaggi che comunicano l'andamento dell'esecuzione dello script, poiché per poter effettuare questa query è stato necessario dover introdurre una pausa di 60 secondi

ogni 30 richieste, perché come spiegato prima Github ha imposto un limite di richieste per minuto, fissato proprio a 30.

Pertanto, senza l'introduzione di questa "pausa", si rischiava di non riuscire ad effettuare le richieste in modo corretto, ottenendo dei messaggi di errore, proprio perché si era superato quel limite.

Altro messaggio è quello che comunica il salvataggio e la creazione del file Json contenente l'elenco completo dei nomi dei progetti.

Ed in output è possibile vedere i seguenti risultati:

```
Pausa di 60 secondi, per poter effettuare altre richieste. Mancano ancora 476 progetti.
Pausa di 60 secondi, per poter effettuare altre richieste. Mancano ancora 446 progetti.
Pausa di 60 secondi, per poter effettuare altre richieste. Mancano ancora 416 progetti.
Pausa di 60 secondi, per poter effettuare altre richieste. Mancano ancora 386 progetti.
Pausa di 60 secondi, per poter effettuare altre richieste. Mancano ancora 356 progetti.
Pausa di 60 secondi, per poter effettuare altre richieste. Mancano ancora 326 progetti.
Pausa di 60 secondi, per poter effettuare altre richieste. Mancano ancora 296 progetti.
Pausa di 60 secondi, per poter effettuare altre richieste. Mancano ancora 266 progetti.
Pausa di 60 secondi, per poter effettuare altre richieste. Mancano ancora 236 progetti.
Pausa di 60 secondi, per poter effettuare altre richieste. Mancano ancora 206 progetti.
Pausa di 60 secondi, per poter effettuare altre richieste. Mancano ancora 176 progetti.
Pausa di 60 secondi, per poter effettuare altre richieste. Mancano ancora 146 progetti.
Pausa di 60 secondi, per poter effettuare altre richieste. Mancano ancora 116 progetti.
Pausa di 60 secondi, per poter effettuare altre richieste. Mancano ancora 86 progetti.
Pausa di 60 secondi, per poter effettuare altre richieste. Mancano ancora 56 progetti.
Pausa di 60 secondi, per poter effettuare altre richieste. Mancano ancora 26 progetti.
 Il file : 'elenco_info_test_unity_augmentedreality.json, é stato salvato correttamente!
 Il file : 'elenco_url_test_unity_augmentedreality.json, é stato salvato correttamente!
L'elenco dei progetti di realità aumentata da visionare sono 506
Sono stati trovati 221 che presentano la parola chiave <<Test>>
Pausa per fine ricerca nei repositories della parola chiave: test

Process finished with exit code 0
```

Figura 108 - Output dell'esecuzione della ricerca

Come mostrato nell'esecuzione dello script, nella figura 108, inizialmente erano presenti nella lista 506 progetti, dove solo 221 hanno presentato un riscontro della presenza al loro interno della parola chiave "Test", ovvero solo il 43.7% circa dei progetti iniziali potrebbe contenere al proprio interno dei tests.

L'esecuzione di questo script genera inoltre un file Json che contiene l'elenco di tutti i progetti in cui è stata riscontrata la presenza della parola chiave, il quale verrà utilizzato successivamente come input per lo script sull'analisi e pulizia di questo elenco.

Come mostrato nella figura successiva (figura 109), il file Json si presenta nella seguente forma:


```
[
    "https://github.com/andrewnakas/OpenBrushVR",
    "https://github.com/vimeo/vimeo-unity-sdk",
    "https://github.com/ronellsicat/DxR",
    "https://github.com/andrewnakas/ARKit-Cardboard-VR",
    "https://github.com/yasirkula/UnityIonicIntegration",
    "https://github.com/marlon360/covid-ar",
    "https://github.com/namanrajpal/FurnitureAR",
    "https://github.com/aidawhale/AnimalRoadtrip-App",
    "https://github.com/aitc19x/VR-JC",
    "https://github.com/dioram/Elektronik-Tools-2.0",
    "https://github.com/LightningArtist/latkUnity",
    "https://github.com/ARtourism/NearSight",
    "https://github.com/yangfawen/RAVVAR-XunAPI",
    "https://github.com/North-Star-Research-Institute/North-Star-Unity-Starter-Project",
    "https://github.com/liubq7/FoodX",
    "https://github.com/danielfranze/HighscoreAR",
    "https://github.com/manuelfuchs/grammAR",
    "https://github.com/Okba28/ProjectNorthStar",
    "https://github.com/carolina-ar-vr/cue",
    "https://github.com/dralois/ARchitecture",
    "https://github.com/milovanarsul/TimiPass",
    "https://github.com/hi-space/DBD-Unity-AR-Basic",
    "https://github.com/nlckfq/latkxExtras",
    "https://github.com/williamdsw/unity-vuforia-ra-demo",
    "https://github.com/nlckfq/Hello_Lens",
    "https://github.com/viviancan/Unity-Roll-a-Ball",
    "https://github.com/junyoung1992/Unity-AR-Examples",
    "https://github.com/npai96/PocketGolem_MarkerBased",
    "https://github.com/comeandsee/Hunters",
```


Figura 109 - File json contenente di output

11.3 Perfezionamento della ricerca

Lo script precedentemente descritto permette quindi la ricerca in ogni singolo file, restituendo per ogni progetto i link dei progetti in cui la ricerca ha avuto successo, ed è stato trovato un effettivo riscontro della presenza della parola chiave "Test", il che non implica però che necessariamente vi siano dei test.

Infatti, proprio per andare a fondo di questa ricerca ed effettivamente poter capire in modo più dettagliato se sono presenti o meno di file di Test, e poterli analizzare e categorizzare, è stata sviluppata una ulteriore funzione che permettesse di fare una "pulizia" ed avere a disposizione tutti i link dei file in cui vi è stato un riscontro di possibile Test trovato.

	ELEMENTO	DESCRIZIONE
NOME	RICERCA PATH	Questa funzione permette di poter effettuare delle ricerche dei link dei dei file che presentano al loro interno la parola chiave usata come input, per ogni repository.
TIPOLOGIA	FUNZIONE	
INPUT	ELENCO FINALE	Elenco dei nomi dei progetti in cui ricercare.
	PAROLA CHIAVE	Termine da usare per la ricerca nella query
OUTPUT	ELENCO INFO	Lista contenente tutte le informazioni sui file trovati all'interno di ogni repository.
ESEMPIO D'USO	Funzioni.ricerca_path(elenco_finale,parola_chiave)	

Tabella 24 - Informazioni funzione "ricerca path"

La funzione, che permette la ricerca dei path/link all'interno dei repositories, è la seguente:

```
def ricerca_path(elenco_finale,parola_chiave):
    elenco_info = []
    i = 0
    count = 0

    for id in elenco_finale:
        if count == 30:
            i = i + 1
            count = 0
            print("Pausa di 60 secondi, mancano ancora " +
str(len(elenco_finale) - i * 30) + " progetti.")
            time.sleep(60)

        # query per la ricerca delle parole chiave nel codice
        test = requests.get(
            'https://api.github.com/search/code?q=' + parola_chiave +
'+in%3afile+repo%3a' + id + '&per_page=100',
            auth=('enzot95', ACCESS_TOKEN)).json()
        num = test["total_count"]
        if num > 0:
            lista_appoggio = []
```

```
            for id in test["items"]:
                lista_appoggio.append(id["path"])
            elenco_info.append([test["items"][0]["repository"]["html_url"],
lista_appoggio])

        count = count + 1
    print("Numero dei progetti trovati che hanno al loro interno la parola
chiave "+str(len(elenco_info)))

    return elenco_info
```

Figura 110 - Funzione "ricerca_path"

In aggiunta alla query, è stato sviluppato una ulteriore funzione che consentisse inoltre di raffinare la ricerca eliminando eventuali "link doppi", ovvero nel caso in cui nello stesso file, la parola chiave dovesse esser presente più volte.

In questa funzione sono stati inoltre eliminati tutti i file che non avessero una estensione corretta, e che quindi risultassero essere file fuorvianti per l'obiettivo che ci si è prefissati.

	ELEMENTO	***DESCRIZIONE***
NOME	PULIZIA PATH TEST	Questa funzione permette di poter effettuare un filtraggio dell'elenco dei link passati come input, in modo da rimuovere tutti quelli che non sono, ne possono essere considerati come dei possibili file di test.
TIPOLOGIA	FUNZIONE	
INPUT	INFO TESTS	Elenco delle informazioni sui file di tests trovate all'interno dei progetti di realtà aumentata.
OUTPUT	INFO PULITE	Lista contenente tutte le informazioni filtrate sui file trovati all'interno di ogni repository.
ESEMPIO D'USO	Funzioni.pulizia_path_test(info_tests)	

Tabella 25 - Informazioni funzione "pulizia path tests"

Il codice della funzione viene mostrato nelle successive immagini (figura 111, 112 e 113).

```
def pulizia_path_test(info_tests):
    info_pulite = []
    paths_iniziali = []
    paths_finali = []

    for progetto in info_tests:

        lista_tests = []
        nome_progetto = progetto[0]
        stringa_http = nome_progetto + '/tree/master/'

        # Ricerca dei path che presentano possibili file di test
        for text in progetto[1]:
            count = 0
            key_words = ['/Assets/', 'Test']
            estensione_file = ['.shader', '.mat', '.prefab',
'.unity','.md','.anim','controller','.asmdef','.json','.js']

            if len(text) == 0:
                count = count + 1
            # controllo per la presenza delle key words più controllo
sull'estensione del file corretto
            y = text.find(key_words[0])
            if y != -1:
                count = count + 1
            if count != 0:
                g = text.find(key_words[1])
                if g != -1:
                    count = 0
                    for estensione in estensione_file:
                        z = text.find(estensione)
                        if z != -1:
                            count = count + 1
                    if count == 0:
                        lista_tests.append(text)
            lista_tests = sorted(lista_tests)

        paths_iniziali.append(len(progetto[1]))

        lista_finale = []
```

Figura 111 - Funzione "pulizia_path_test" (pt.1)

```
# Eliminazione di eventuali path doppi
if len(lista_tests) != 0:
    lista_finale.append(stringa_http + lista_tests[0])
    for i in range(len(lista_tests) - 1):
        if lista_tests[i] != lista_tests[i + 1]:
            lista_finale.append(stringa_http + lista_tests[i + 1])
    # print("Sono stati eliminati "+str(len(lista_tests)-
len(lista_finale))+" file doppi")

    info_pulite.append({"repos": nome_progetto, "path": lista_finale, "oc-
correnze": len(lista_finale)})

paths_finali.append(len(lista_finale))
```

Figura 112 - Funzione "pulizia_path_test" (pt.2)

```
print(" ")
print("Paths iniziali trovati " + str(sum(paths_iniziali))+" all'interno di
"+str(len(info_tests))+" repositories.")
print("Paths finali trovati " + str(sum(paths_finali))+" all'interno di
"+str(len(info_pulite))+" repositories.")
print("Sono stati eliminati il " + str(round(100 - sum(paths_finali) * 100
/ sum(paths_iniziali)))+
      " % di paths che non corrispondevano effettivamente a dei tests.")
print("Solo il " + str(round(100 - sum(paths_finali) * 100 /
sum(paths_iniziali)))+" % dei progetti su "+
      str(len(info_tests))+" presentano la potenziale presenza di tests.")

return info_pulite
```

Figura 113 - Funzione "pulizia_path_test" (pt.3)

Da annoverare che la funzione, appena mostrata, ha subito in corso d'opera una modifica.

Si è preferito infatti, apportare dei cambiamenti alla funzione di "raffinamento" della ricerca, facendo in modo di escludere tutti i file presenti in determinati path del progetto.

In particolare, questa scelta è nata dopo una prima visione dell'elenco dei link ottenuti in output, in cui sono stati riscontrati dei file che effettivamente non corrispondevano a dei test veri e propri, ma appartenevano ad esempio a specifiche librerie di Sdk, tool grafici, file unity ed altre cartelle simili.

Queste non rappresentavano dei tests veri e propri, ma erano soltanto dei file, probabilmente appartenenti anche a dei file di testo standard non inserite direttamente dal progettista, che invece appartenevano a delle cartelle create al momento della creazione del progetto in Unity.

Pertanto, grazie alla minuziosa ricerca attuata su Github, che però ovviamente ha portato ad ottenere dei file contenenti la parola chiave, ma che in pratica non erano effettivamente dei veri e propri Test.

La ricerca finale, ovvero quella attuata nella funzione mostrata precedentemente, è stata quindi effettuata attraverso lo script ad Hoc.

	ELEMENTO	DESCRIZIONE
NOME	ANALISI TEST TROVATI	Questo script consente di ottenere l'elenco dei file che presentano al loro interno probabili test, dentro i repositories di realtà aumentata.
TIPOLOGIA	SCRIPT	
INPUT DATI	LISTA TEST	Elenco dei nomi dei progetti in cui ricercare.
	PAROLA CHIAVE	Termine da usare per la ricerca nella query
FUNZIONI USATE	RICERCA PATH	Funzione di ricerca delle informazioni inerenti ai file di test nei repositories
	PULIZIA PATH TESTS	Funzione di filtraggio delle informazioni ritenute errate e non inerenti alla ricerca.
	SALVA REPOSITORIES TWO TAGS	Funzione di salvataggio delle informazioni ricercate, memorizzate in un apposito file json.
OUTPUT	INFORMAZIONI	Elenco di informazioni visibili tramite console inerenti ai repositories ricercati.
	FILE JSON	File Json creati contenenti le varie informazioni ricercate.
ESEMPIO D'USO	Analisi_test_trovati.py (da eseguire da riga di comando)	

Tabella 26 - Informazioni script "Analisi Test Trovati"

Questa ha quindi permesso di poter trovare l'elenco di tutti i link dei file presenti nei repositories, che potessero rappresentare in qualche modo dei possibili Test.

Lo script che esegue queste funzioni di ricerca, e successivamente di pulizia viene mostrato nella seguente immagine (figura 114).

```
import json
import funzioni

ACCESS_TOKEN = 'af314d573e86e8780c88bf445e33584567bf120a'

# Questo script consente di analizzare i repositories che potrebbero pre-
sentare al loro interno dei possibili tests
# ed inoltre di pulire all'interno della ricerca effettuata tutti i possi-
bili link doppi (ovvero ripetizioni), in modo
# da ottenere un elenco completo per ogni progetto dei link/paths con il
riscontro della parola chiave "Test".

parola_chiave='test'
```

```
lista_test = json.load(open("elenco_url_test_unity_augmentedreality.json"))

# Rimozione della parte del link che non serve
info=[]
for id in lista_test:
    info.append(id[19:])
elenco_finale=sorted(info)

print("Numero dei progetti totali presenti nella lista
"+str(len(elenco_finale)))

elenco_info= funzioni.ricerca_path(elenco_finale, parola_chiave)
funzioni.salva_repositories_two_tag(elenco_info, "info_test", "unity",
"augmentedreality")

#Pulizia dell'elenco delle informazioni trovate nei repositories
paths = funzioni.pulizia_path_test(elenco_info)
funzioni.salva_repositories_two_tag(paths, "paths_tests", "unity", "augmen-
tedreality")
```

Figura 114 - Script analisi e pulizia della ricerca

Nella figura sottostante è invece possibile vedere il risultato in output mostrato nella console:

```
Numero dei progetti totali presenti nella lista 221
Pausa di 60 secondi, mancano ancora 191 progetti.
Pausa di 60 secondi, mancano ancora 161 progetti.
Pausa di 60 secondi, mancano ancora 131 progetti.
Pausa di 60 secondi, mancano ancora 101 progetti.
Pausa di 60 secondi, mancano ancora 71 progetti.
Pausa di 60 secondi, mancano ancora 41 progetti.
Pausa di 60 secondi, mancano ancora 11 progetti.
Numero dei progetti trovati che hanno al loro interno la parola chiave 221
 Il file : 'info_test_unity_augmentedreality.json, é stato salvato correttamente!

Paths iniziali trovati 6380 all'interno di 221 repositories.
Paths finali trovati 232 all'interno di 27 repositories.
Sono stati eliminati il 96% di paths che non corrispondevano effettivamente a dei tests.
Solo il 96% dei progetti su 221 presentano la potenziale presenza di tests.
 Il file : 'paths_tests_unity_augmentedreality.json, é stato salvato correttamente!

Process finished with exit code 0
```

Figura 115 - Esecuzione dell'analisi

Come è possibile vedere nella precedente immagine, nei risultati vengono mostrati a video nella console anche il numero di progetti che hanno presentato la presenza di paths contenenti possibili tests.

Numericamente, quindi, solo 27 repositories su 221 iniziali, ovvero un 12,2% derivante dopo la prima scrematura, ma solo un 5,3 % sul totale dei progetti

analizzati, presenta la possibile eventualità di presenza di tests creati da parte dei progettisti.

Nella successiva figura è possibile vedere il file Json che è stato creato dopo l'esecuzione dell'ultimo script, sull'analisi/pulizia dei paths:


```
[
    [
        "https://github.com/00111000/City-of-Calgary-2018",
        [
            "AR Pedestrian/ProjectSettings/UnityConnectSettings.asset",
            "AR Pedestrian/Assets/GoogleARCore/Examples/CloudAnchor/Scripts/ARKitHelper.cs",
            "AR Pedestrian/Library/PlayerDataCache/ScriptLayoutHashes.txt",
            "AR Pedestrian/Assets/GoogleARCore/SDK/Scripts/Api/HitTestApi.cs",
            "AR Pedestrian/Assembly-CSharp-Editor.csproj",
            "AR Pedestrian/Assets/GoogleARCore/SDK/Scripts/Api/NativeSession.cs",
            "AR Pedestrian/Assembly-CSharp.csproj",
            "AR Pedestrian/Assets/GoogleARCore/SDK/Scripts/CameraMetadataTag.cs",
            "AR Pedestrian/Assets/GoogleARCore/SDK/Scripts/Frame.cs",
            "AR Pedestrian/ProjectSettings/ProjectSettings.asset",
            "AR Pedestrian/Library/ProjectSettings.asset"
        ]
    ],
    [
        "https://github.com/0x106/Nebula-Beta",
        [
            "README.md"
        ]
    ],
    [
        "https://github.com/8thwall/xr-unity",
        [
            "scripts/SunlightController.cs",
            "scripts/README.md"
        ]
    ],
```


Figura 116 - File Json grezzo


```
[
    {
        "occorrenze": 1,
        "path": [
            "https://github.com/00111000/City-of-Calgary-2018/tree/master/AR Pedestrian/Assets/GoogleARCore/SDK/Scripts/Api/HitTestApi.cs"
        ],
        "repos": "https://github.com/00111000/City-of-Calgary-2018"
    },
    {
        "occorrenze": 2,
        "path": [
            "https://github.com/AGI20-Group01/AGI20_Group01_MystARies/tree/master/AGI20/Assets/Scripts/Grid System/GroudeMaterialCreaters/Old/TestCubeMaterialCreater.cs",
            "https://github.com/AGI20-Group01/AGI20_Group01_MystARies/tree/master/AGI20/Assets/SocketIO/Scripts/Test/TestSocketIO.cs"
        ],
        "repos": "https://github.com/AGI20-Group01/AGI20_Group01_MystARies"
    },
    {
        "occorrenze": 2,
        "path": [
            "https://github.com/AhmedAlsaab/RMP-Fujitsu/tree/master/Unity/Fujitsu AR Project/Fujitsu Unity Project/Assets/Scripts/Test/ProcessGenerationTest.cs",
            "https://github.com/AhmedAlsaab/RMP-Fujitsu/tree/master/Unity/Fujitsu AR Project/Fujitsu Unity Project/Assets/Scripts/Test/ProcessPageTest.cs"
        ],
        "repos": "https://github.com/AhmedAlsaab/RMP-Fujitsu"
    },
    {
        "occorrenze": 1,
        "path": [
            "https://github.com/CJT-Jackton/Mixed-Reality-Theater/tree/master/MagicLeap/Assets/Scripts/TestWriter.cs"
        ],
        "repos": "https://github.com/CJT-Jackton/Mixed-Reality-Theater"
    },
```


Figura 117 - File Json versione raffinata

Come è possibile notare nel file json, mostrato in figura 117, sono stati inseriti degli oggetti che rappresentano in modo abbastanza dettagliato ognuno dei 27 progetti, in cui è stata riscontrata la possibilità di una presenza di Tests.

In ogni oggetto, che rappresenta quindi un repository, vi sono due attributi che indicano rispettivamente:

- il link del repository di riferimento in questione;
- il numero di occorrenze di paths che sono stati trovati all'interno di esso.

Il terzo attributo contiene la lista effettiva dei paths/link trovati all'interno di ogni progetto.

Questi script sono stati ideati, e sviluppati con lo scopo in primis di trovare eventuali tests presenti all'interno dei repositories legati alla realtà aumentata, ma anche di rendere di gran lunga più semplice l'effettiva verifica per controllare se questi link, che rappresentano l'indirizzamento a dei file ".cs" interni ai progetti, corrispondano o meno da dei tests.

11.4 Ricerca dei file di Test

Arrivati a questo punto una volta effettuata la ricerca automatizzata per ottenere la lista dei repositories, per valutarne la presenza al loro interno di file di Test, può essere svolta.

Come prima cosa sono stati controllati tutti i file presenti nella lista in modo da osservare in modo diretto se questi fossero effettivamente dei Test o meno.

Dal punto di vista numerico con l'esecuzione degli script precedenti si è arrivati ad avere una lista contenete 27 repositories, contenenti un totale di 229 file da controllare.

Quindi in questa fase si è cercato di fare una distinzione tra i vari file ottenuti dalle ricerche, dove si è evidenziato che:

- dei 229 file solo 22 non erano dei test, ma solo file fuorvianti che corrispondono ad una percentuale di quasi il 10%;
- dei restanti 207 file trattasi effettivamente di file che presentano dei test, ma che di questi solo 55 rappresentano effettivamente dei test sviluppati dai progettisti stessi, ovvero il 24%;
- quindi il restante 66%, 152 file su 229, sono tests standard prodotti da altre aziende quali ad esempio Microsoft, Google, Meta o Ibm.

Questi numeri appena citati, possono essere riassunti meglio nella seguente tabella (tabella 27):

TIPOLOGIA	#	%
NO TEST	22	9,6
TEST	55	24
TEST STANDARD	152	66,4
TOTALE	***229***	***100***

Tabella 27 - Percentuali delle tipologie di file trovate nelle ricerche

Nel dettaglio nei 27 repositories facenti parte dell'elenco della ricerca dei Test si è evidenziato che dopo il controllo 3 di questi non presentavano nessun test effettivo, mentre gli altri 24 presentavano la presenza di diverse combinazioni.

Più precisamente:

- 6 repositories avevano al loro interno dei test creati ad hoc esclusivamente dal progettista, con una percentuale del 22%;

- in 13 repositories, ovvero più del doppio, invece avevano solo test standard ovvero non creati dallo sviluppatore, che raggiungono quasi una percentuale del 50%;
- infine, solo il 19% circa delle applicazioni, quindi 5 di loro, avevano al loro interno entrambe le tipologie.

Questi numeri, appena descritti, vengono meglio riassunti nella tabella sottostante (tabella 28):

REPOSITORY	#	%
SENZA TEST	3	11,1
SOLO TEST	6	22,2
SOLO TEST STANDARD	13	48,1
ENTRAMBE	5	18,5
TOTALE	***27***	***100***

Tabella 28 - Test e non trovati nei repositories

Un ulteriore approfondimento, che è stato fatto, è di andare a controllare questi file di test, in modo da cercare di capire di che tipologia di test si trattasse.

Pertanto, dapprima sono stati controllati tutti i file di ogni progetto, valutando la presenza delle etichette che contraddistinguono la presenza o meno di un test eseguibile in Unity:

- [Test]
- [UnityTest]

Queste due etichette sono presenti ogni qual volta vi è la presenza di un Test all'interno di queste tipologie di file, la differenza tra le due è che sostanzialmente l'etichetta [UnityTest] indica che quella funzione è in grado di eseguire il test come una Coroutine.

Per Coroutine nell'ambiente Unity si intende quando viene chiamata una funzione, e viene eseguita fino al completamento prima di tornare.

Ciò significa effettivamente che qualsiasi azione che venga svolta in una funzione, essa deve avvenire all'interno di un singolo aggiornamento del frame.
Una chiamata di funzione non può essere quindi utilizzata per contenere un'animazione procedurale o una sequenza di eventi nel tempo.
Mentre per quanto riguarda la seconda l'etichetta, ovvero [Test], questa invece viene posta su quelle funzioni che eseguiranno sempre l'intera operazione in modo del tutto sincrono in un singolo frame.
Pertanto, dopo aver fatto questa primissima "categorizzazione", i test trovati sono stati ulteriormente divisi in:

- Test creati,
- Test Standard.

Nel primo caso questi solitamente vengono situati nelle cartelle principali delle applicazioni internamente a cartelle nominate "Assets", nel migliore dei casi all'interno di esse è possibile addirittura trovarli all'interno di cartelle rinominate "Tests".
Ovviamente questi repositories, contenendo delle applicazioni totalmente open-source, ed oltretutto sviluppate non necessariamente da sviluppatori professionisti, la disposizione delle cartelle non è una cosa generalizzabile.
Mentre per quanto riguarda la seconda tipologia ovvero i test definiti "Standard", in questo caso si fa riferimento a dei test non sviluppati per l'applicazione in questione, né tantomeno dallo sviluppatore stesso, ma sono dei Test che sono stati progettati dalle aziende dei tool, plugin o degli Sdk che sono stati utilizzati all'interno dell'applicazione stessa.
La seguente tabella mostra l'elenco completo del controllo dei file che è stato fatto, per ognuno dei 229 file, ottenuti dalla ricerca fatta con gli script Python:

	OCC	NO TEST	TEST	TIPO	TEST STANDARD	CASA PRODUTTRICE
1	1	0	0	-	1	GOOGLE
2	2	2	0	-	0	-
3	22	0	0	-	22	MICROSOFT
4	2	0	2	UNITA’	0	-
5	1	1	0	-	0	-
6	3	1	0	-	2	GOOGLE
7	1	0	0	-	1	GOOGLE
8	1	0	0	-	1	GOOGLE
9	3	3	0	-	0	-
10	1	0	1	CAMERA AR	0	-
11	46	2	0	-	44	LEAP MOTION
12	3	2	0	-	1	GOOGLE
13	1	1	0	-	0	-
14	1	1	0	-	0	-
15	1	0	0	-	1	GOOGLE
16	4	3	0	-	1	GOOGLE
17	4	2	2	UNITA’	0	-
18	19	1	11/6	SIST. & UNITA'	0	-
19	19	1	11/6	SIST. & UNITA'	0	-
20	19	1	0	-	18	MICROSOFT
21	4	3	0	-	1	GOOGLE
22	1	1	0	-	0	-
23	10	2	0	-	8	NAVMESH
24	25	0	0	-	25	MICROSOFT
25	21	0	0	-	21	IBM
26	3	0	3	SISTEMA	0	-
27	11	0	8	UNITA’	2	MICROSOFT
TOT	***229***	***30***	***50***		***149***	

Tabella 29 - Elenco completo dei file analizzati

Per poter comprendere meglio la precedente tabella, è meglio descrivere nel merito ogni colonna cosa rappresenta.

Partendo da sinistra verso destra, la tabella presenta sette colonne, in cui:

- La prima colonna indica il numero del repository, presente nel file Json, a cui si sta facendo riferimento;

- Nella seconda colonna, rinominata "Occ", abbreviativo della parola "Occorrenze", indica il numero di file che sono stati trovati all'interno del repository di riferimento;
- Nella terza, quarta e sesta colonna invece, sono stati suddivisi il numero di file per:
    - tipologia "primitiva", ovvero in ordine vi sono il numero di file che non presentano test,
    - il numero i file che invece presentano test,
    - infine, il numero di file che presentano test standard.
- Infine, nella quinta e settima colonna, vengono indicate rispettivamente:
    - le tipologie di test trovati, in riferimento alla colonna dei test (la quarta);
    - le case produttrici dei test standard che sono stati trovati.

I test standard in cui si fa riferimento alle loro informazioni nella sesta e settima colonna della tabella precedente, sono dei test facilmente rilevabili all'interno di un repository, perché presentano generalmente, all'interno del file di riferimento del test, delle righe commentante abbastanza caratteristiche.

In queste righe vi è spesso una breve introduzione allo scopo di quel test, oltre che il marchio del Copyright[37] "© ", attraverso il quale la casa che lo ha sviluppato ne rivendica la proprietà.

Questa tipologia di tests è stata rilevata in diversi progetti non sono altro che dei file di test tutti uguali, i quali non presentano alcuna modifica apportata dallo sviluppatore, rimasti fedelmente originali dal momento che sono stati

[37] Il copyright (termine di lingua inglese che letteralmente significa *diritto di copia*) è un termine che identifica il diritto d'autore nei paesi di common law, dal quale però differisce sotto vari aspetti. Ciononostante, il termine viene comunemente usato anche per indicare genericamente la normativa sul diritto d'autore degli ordinamenti di civil law.

importati nel progetto, tramite l'utilizzo di un particolare plugin o tool servito per lo sviluppo dell'applicazione.

11.5 Test Standard

Per chiarire e descrivere meglio a cosa ci si fa riferimento, nelle immagini successive (figura 118 e 119) vengono mostrati alcuni estratti dei test standard sviluppati da alcune case produttrici trovati nei vari repositories:

37 lines (32 sloc) | 1.44 KB

```
1  // Copyright (c) Microsoft Corporation. All rights reserved.
2  // Licensed under the MIT License. See LICENSE in the project root for license information.
3
4  using UnityEngine;
5  using System.Collections;
6
7  namespace HoloToolkit.Examples.GazeRuler
8  {
9      /// <summary>
10     /// it's a test case, we try create line correctly between two points
11     /// </summary>
12     public class LineTest : MonoBehaviour
13     {
14
15         public GameObject start;
16         public GameObject end;
17         public GameObject line;
18         public GameObject text;
19
```

Figura 118 - Esempio di test Standard Microsoft HoloLens (pt.1) [21]

```
19
20         // Update is called once per frame
21         private void Update()
22         {
23             var distance = Vector3.Distance(start.transform.position, end.transform.position);
24             var midPoint = (start.transform.position + end.transform.position) * 0.5f;
25             var direction = end.transform.position - start.transform.position;
26             line.transform.position = midPoint;
27             line.transform.localScale = new Vector3(distance, 1.0f, 1.0f);
28             line.transform.rotation = Quaternion.LookRotation(direction);
29             line.transform.Rotate(Vector3.down, 90f);
30             text.transform.position = midPoint + new Vector3(0, 0.6f, 0);
31             text.transform.rotation = Quaternion.LookRotation(direction.x + direction.y + direction.z < 0 ? direction * -1 : direction);
32             text.transform.Rotate(Vector3.up, -90f);
33
34             Debug.Log(Vector3.Angle(direction, Vector3.forward));
35         }
36     }
37 }
```

Figura 119 - Esempio di Copyright Microsoft [21]

Come è possibile notare nelle due figure precedenti, nella parte iniziale vi è una parte commentata, dove è presente il nome della casa produttrice di quel file.

In generale di questi file non si tratta di veri e propri file di test, ma sono invece dei file presenti nel momento in cui la cartella del componente da utilizzare nel progetto, viene inserito in esso.

Questi quindi non essendo dei test sviluppati direttamente per l'applicazione in questione non possono essere considerati come dei test da prendere in considerazione, e valevoli di analisi.

Pertanto, dopo aver controllato tutti i file con test standard, è possibile diramare una tabella (tabella 30) riepilogativa che presenta le seguenti informazioni:

CASA PRODUTTRICE	# TEST	% TEST	REPOS.	% REPOS.
MICROSOFT	67	47,9	4	57,1
LEAP MOTION	44	31,4	1	14,3
IBM	21	15	1	14,3
NAVMESH	8	5,7	1	14,3
TOTALE	***140***	***-***	***16***	***-***

Tabella 30 - File di test standard

Questa tabella presenta, come al solito, diverse colonne che indicano:

- la prima, il nome della casa produttrice, dei file di test standard di riferimento;
- la seconda e la terza, mostrano rispettivamente il numero di test trovati e la loro percentuale di presenza sul totale dei test standard trovati, ovvero 140;
- mentre la quarta e la quinta, mostrano a loro volta rispettivamente in quanti repositories sono stati trovati dei test standard della casa madre di riferimento, ed ovviamente la percentuale sul totale.

Attraverso questi numeri è quindi possibile notare come l'azienda più presente, in questi repository analizzati, dal punto di vista del numero di file di test standard che trovati è la Microsoft, con una percentuale di circa il 48%.

Mentre, per quanto riguarda la presenza di almeno un test standard per ogni repository, al primo posto vi è Microsoft, con una percentuale del 57%, ovvero il triplo rispetto agli altri che sono stati riscontrati.

Una spiegazione plausibile, del tale dominio dell'azienda, nasce probabilmente dal fatto che ha in produzione degli strumenti che permettono l'utilizzo e lo sviluppo di applicazioni di realtà aumentata, ovvero gli HoloLens della Microsoft.

Infatti, questi file fanno parte del plugin "HoloToolKit", il quale è un package open source disponibile sin dalla prima versione di HoloLens, che ha raggiunto i livelli attuali di qualità e popolarità grazie al contributo e l'impegno costante della community di sviluppatori Microsoft.

Una ulteriore rappresentazione grafica delle percentuali espresse in precedenza può essere visionata nel seguente grafico (Figura 120 e 121):

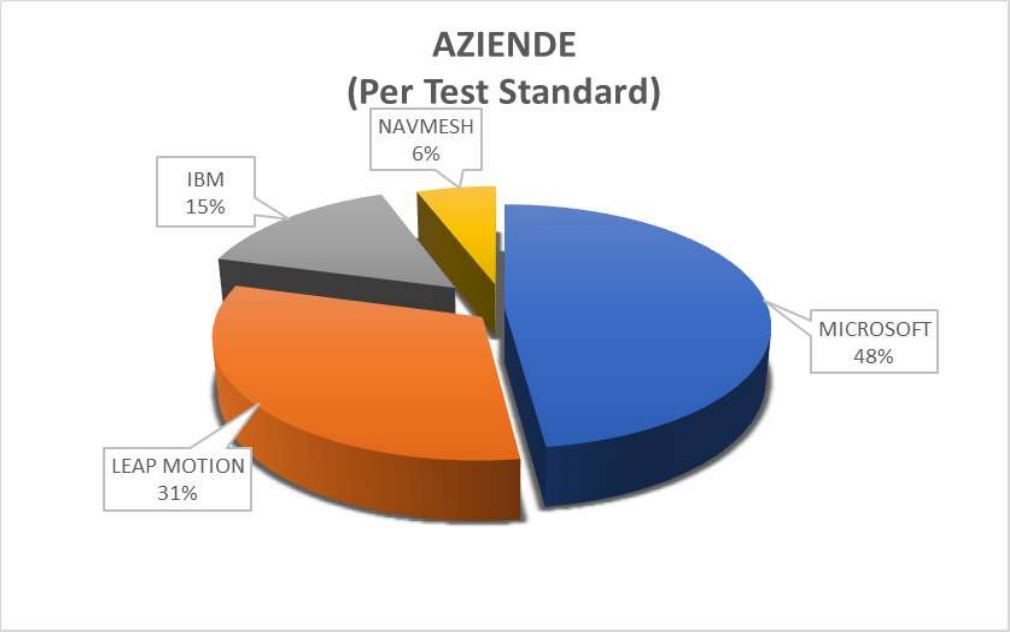

Figura 120 - Grafico della presenza dei Test Standard delle aziende

Il grafico della figura 120 indica le percentuali della presenza di test standard rilevata nei repository controllati, per casa di produzione.

Mentre il grafico successivo (figura 121) indica la percentuale di presenza delle aziende nei repository.

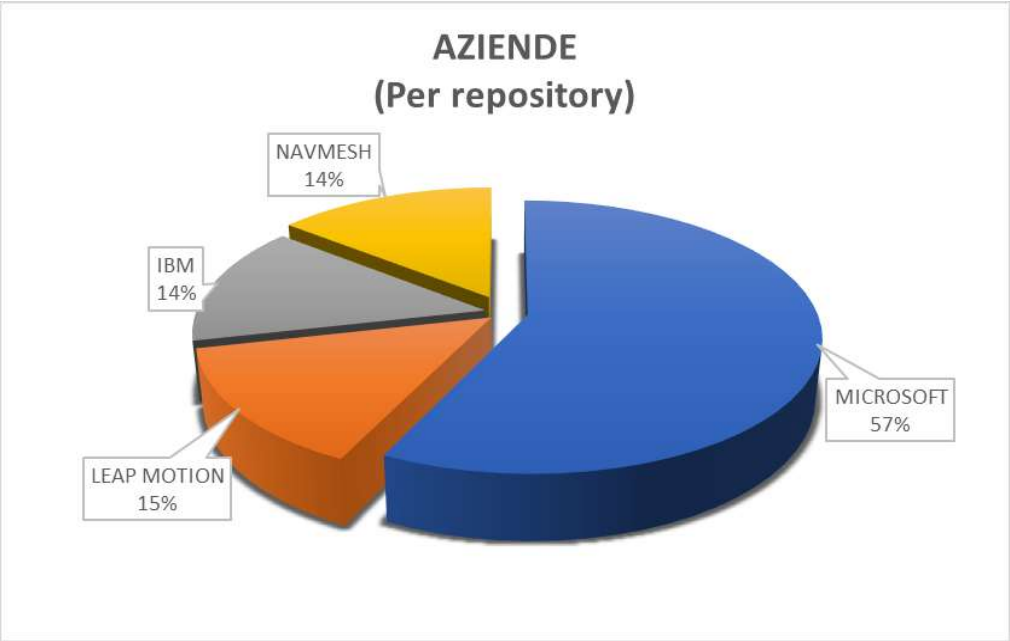

Figura 121 - Grafico della presenza delle aziende nei repository

11.6 Test creati

Spostiamoci adesso verso l'analisi dei restanti test, ovvero quelli creati dagli stessi sviluppatori.

Questi test sono stati sviluppati in modo del tutto autonomo, e sono collegati in qualche modo all'applicazione stessa di realtà aumentata a cui fanno riferimento.

Secondo le stime dei controlli effettuati nei repository, sono stati trovati 50 test, ovvero solo circa il 22% di tutti i possibili file di test che sono stati trovati dalla ricerca effettuata tramite gli script di Python, su 229 totali.

I 50 test trovati, vanno divisi per 6 repositories che li contengono, e tenendo conto che le applicazioni iniziali da cui la ricerca ha avuto inizio erano 506,

questo significa che solo circa il 1,2% dei progetti prevedono la presenza di file di test per il controllo e la verifica di questi progetti.

Questo numero è davvero molto basso, ma le spiegazioni come già riportato più volte nel corso dei capitoli, possono risiedere nel fatto che questa tecnica di sviluppo è ancora in fase nascente, quindi si conosce ancora poco sia sulla tecnica che su eventuali metodologie di testing.

Altro fattore importante, che potrebbe giustificare una così scarsa presenza di test, potrebbe essere quello della difficoltà che vi è appunto nel testare applicazioni di questo genere.

Difficoltà che nascono per via della bassissima presenza di codice, e/o di script che gestiscono il funzionamento dell'applicazione, poiché queste applicazioni presentano una tecnica di programmazione di alto livello, nel senso che gran parte del loro funzionamento è basato sul riconoscimento del marker.

Questo, molto spesso, non è un compito né tantomeno una responsabilità del programmatore, quella di verificarne il funzionamento del riconoscimento, proprio perché esistono dei software come Vuforia, che data un immagine tramite il rilevamenti di alcuni punti di ancoraggio, la trasforma in un marker, il quale viene inserito all'interno del progetto, senza che il programmatore in questione possa fare nulla per migliorarne il riconoscimento, se non quello di decidere di cambiare immagine da utilizzare come input per la generazione del marker.

Pertanto, le uniche cose che potenzialmente possono essere testate sono gli script contenenti le funzioni o le azioni di base che vengono utilizzare come contorno per il funzionamento di queste applicazioni di realtà aumentata.

Ed infatti come viene dimostrato dai numeri rilevati dalla ricerca, la grande maggioranza dei test sono di tipologie che non riguardano direttamente funzionalità inerenti alla realtà aumentata.

Infatti, è stato trovato un unico file di test, che riguarda il 2% dei file di test totali, in cui viene testata il funzionamento della camera di AR.

Il restante 98% dei file di test trovati è per la maggior parte file di test di due tipologie in particolare, ovvero:

- Test di unità
- Test di Sistema

Questi file di test fanno parte della stragrande maggioranza dei test trovati, i quali non riguardano direttamente funzionalità di AR, ma servono e vengono utilizzati solo per il controllo e la ricerca di errori nel funzionamento di alcune parti e componenti dell'applicazione.

I numeri non sono molto confortanti dal punto di vista del testing, perché questo dimostra il fatto che molti sviluppatori, soprattutto di applicazioni a livello amatoriale e non professionale, tendono a non curarsi ed a dare poca importanza a questa fase, che dovrebbe essere fondamentale per il corretto sviluppo di una applicazione di qualsiasi genere.

C'è anche da considerare la questione che molto spesso nel campo videoludico e quindi anche in questo settore della realtà aumentata, vi è la tendenza a ricorrere a metodi quali ad esempio il "Beta Testing" per testare queste applicazioni affidandosi quindi al parere dell'utilizzatore e del consumatore, piuttosto che preventivamente controllare la presenza di eventuali malfunzionamenti.

Questo ovviamente non deve essere un attenuate per giustificare l'assenza di test, ma anzi dovrebbe essere un incentivo per evitare che queste applicazioni dovessero andare sul mercato senza un minimo controllo.

Pertanto, è possibile nel dettaglio delle statistiche mediante la tabella riepilogativa (tabella 31) delle tipologie di test riscontrate nei repositories analizzati:

TIPOLOGIA	#	%	REPOS.	%
SISTEMA	25	50	3	37,5
UNITA'	24	48	4	50
CAMERA AR	1	2	1	12,5
TOTALE	*50*		*8*	

Tabella 31 - Elenco delle tipologie di test

In questa tabella è possibile vedere che il 50% dei file di test sono Test di sistema, ovvero 25 test sui 50 trovati, mentre il restante 48% sono i test di unità, ovvero 24 su 50.

Da considerare che in questi numeri c'è da considerare che tre di questi repositories presentano sia test di unità che di sistema, in due casi, ed un terzo caso in cui sono presenti test di unità.

Più nel dettaglio sul totale, vi sono due progetti che hanno 34 file di test quindi il 68% del totale, che sono oltretutto progetti sviluppati dallo stesso programmatore.

I progetti in questione sono:

- https://github.com/chandler767/Cube-Fight
- https://github.com/chandler767/Magic-Leap-Gesture-IoT-Example

che come accennato queste applicazioni al loro interno presentano sia test di unità che test di sistema.

11.7 Test Funzionali

L'altra tipologia di test presenti in queste applicazioni analizzate, sono quelli funzionali, i quali non sono altro che dei test progettati per garantire che ogni parte dell'applicazione si comporti come previsto dai casi d'uso che sono stati prestabiliti durante la progettazione.

Questi test includono quindi dei metodi e possono essere distinti in:

- Test di unità
- Test di sistema

11.7.1 Test di Unità

Nel primo caso i test di unità possono essere considerati i test "di primo livello" ovvero appunto quei test che vengono creati ed eseguiti stesso dagli sviluppatori.

Questo processo serve per garantire che ogni singolo componente dal punto di vista del codice siano funzionali appunto e funzionino come da accordi nelle intenzioni del progetto.

Essi possono essere condotti del tutto manualmente, ma è possibile ovviamente cercare di renderli automatici, il che ne migliorerebbe ulteriormente il processo di testing oltre che l'accelerazione dei cicli di esecuzione della copertura.

Il test di unità viene realizzato mediante un approccio white-box dividendosi in linea generale in "test case", ovvero casi di test, ciascuno dei quali dovrebbe essere indipendente dagli altri.

Il fatto che, esso sia un approccio definito a scatola bianca, nasce dal fatto che vi è un forte legame tra codice sorgente, casi di test e relative specifiche che devono essere verificate, per cui questi test vengono normalmente eseguiti dallo sviluppatore stesso che si è occupato dello sviluppo dell'applicazione. I casi di test, infatti, vengono abitualmente sviluppati in modo del tutto parallelo al codice applicativo e sono implementati tramite procedure automatiche, in grado di rieseguire il test ogni volta che ce ne sia la necessità.

Il motivo per cui tale tipologia di testing risulta spesso fondamentale e molto presente, anche in queste applicazioni che sono state prese in considerazione, è dovuto dal fatto che i difetti sono più facilmente identificabili tramite i malfunzionamenti avvenuti in un ambiente più piccolo e circoscritto.

Se i difetti vengono identificati e risolti in breve tempo dopo la loro introduzione, il tempo per la loro localizzazione e soluzione risulta notevolmente ridotto.

Un'altra buona ragione per eseguire i test di unità è che alcuni difetti emergono solamente in presenza di particolari configurazioni del software, che sarebbero difficilmente ricreabili in un ambiente più complesso e vasto.

Spesso vengono introdotti a tal proposito degli appositi componenti, che cercano di simulare delle unità chiamanti della componente testata, e degli altri componenti chiamati "stub", che simulano l'unità chiamata.

Inoltre, il test di unità abbassa i costi, in termini di tempo e risorse, per l'identificazione e la correzione di difetti, rispetto a quelli che si sarebbero dovuti affrontare per ottenere lo stesso risultato tramite dei test sull'intero applicativo.

In linea generale il testing non riesce a identificare tutti gli errori in un programma e lo stesso vale per i test di unità che, analizzando per definizione le singole componenti, non possono rilevare difetti di integrazione, problemi legati alla performance e altri problemi legati al sistema.

Ovviamente, quindi, Il test di unità risulta essere più efficace se utilizzato insieme con altre tecniche di testing del software.

Questa tipologia di test semplifica soprattutto la fase di debugging poiché permette di ricercare più facilmente i problemi facendo in modo che il tempo di testing sia minore.

Un esempio di test di unità può essere considerato il file di test mostrato nell'immagine sottostante (figura 122 e 123):

65 lines (50 sloc) | 1.2 KB

```
1  using UnityEngine;
2  using UnityEditor;
3  using UnityEngine.TestTools;
4  using NUnit.Framework;
5  using System.Collections;
6
7  public class TestTapNotifier {
8
9      TapNotifier notifier;
10
11         [SetUp]
12     public void SetUp()
13     {
14         GameObject go = new GameObject("TapNotifier");
15         notifier = go.AddComponent<TapNotifier>();
16     }
17
18     [Test]
19     public void TestRegisterOnInputDown()
20     {
21         notifier.RegisterListenerOnInputDown(Success);
22
23         notifier.OnInputDown(null);
24
25         Assert.Fail();
26     }
27
```

Figura 122 - Test di Unità (pt.1) [23]

```
28      [Test]
29      public void TestRegisterOnInputUp()
30      {
31          notifier.RegisterListenerOnInputUp(Success);
32
33          notifier.OnInputUp(null);
34
35          Assert.Fail();
36      }
37
38      [Test]
39      public void TestUnregisterOnInputDown()
40      {
41          notifier.RegisterListenerOnInputDown(Fail);
42          notifier.UnRegisterListenerOnInputDown(Fail);
43
44          notifier.OnInputDown(null);
45      }
46
47      [Test]
48      public void TestUnregisterOnInputUp()
49      {
50          notifier.RegisterListenerOnInputUp(Fail);
51          notifier.UnRegisterListenerOnInputUp(Fail);
52
53          notifier.OnInputUp(null);
54      }
55
56      private void Success()
57      {
58          Assert.Pass();
59      }
60
61      private void Fail()
62      {
63          Assert.Fail();
64      }
65  }
```

Figura 123 - Test di Unità (pt.2) [23]

Questo file comprende una serie di test al suo interno, i quali possono essere facilmente riconosciuti dall'etichetta [Test], come spiegato anche precedentemente.

I test in questione non sono nient'altro che delle funzioni, create ad hoc, atte a verificare il corretto funzionamento o fallimento di alcune azioni che l'applicazione deve svolge, il quale esito verrà notificato tramite gli appositi "Assert", presenti nelle righe 56 e 61 del codice.

L'unità a cui il seguente test fa riferimento è quella mostrata nelle seguenti immagini (figure 124, 125 e 126):

108 lines (97 sloc) | 2.63 KB

```
1   using HoloToolkit.Unity.InputModule;
2   using System.Collections;
3   using System.Collections.Generic;
4   using UnityEngine;
5   using System;
6   using UnityEngine.Events;
7
8   /// <summary>
9   /// Notfies listeners about input events
10  /// </summary>
11  public class TapNotifier : MonoBehaviour, IInputHandler
12  {
13
14      private UnityEvent inputDownEvent;
15      private UnityEvent inputUpEvent;
16
17      /// <summary>
18      /// calls the initialization for the events
19      /// </summary>
20      public void Start()
21      {
22          Init();
23      }
24
25      /// <summary>
26      /// initializes the events
27      /// </summary>
28      private void Init()
29      {
30          if (inputDownEvent == null)
31          {
32              inputDownEvent = new UnityEvent();
33          }
34
35          if (inputUpEvent == null)
36          {
37              inputUpEvent = new UnityEvent();
38          }
39      }
40
```

Figura 124 – Unità "TapNotifier" (pt.1) [23]

```
41      /// <summary>
42      /// Adds a new listener for the input-down-event
43      /// </summary>
44      /// <param name="callback">The listener to remove</param>
45      public void RegisterListenerOnInputDown(UnityAction callback)
46      {
47          Init();
48          if (callback != null)
49          {
50              inputDownEvent.AddListener(callback);
51          }
52      }
53
54      /// <summary>
55      /// Removes a listener from the input-down-event
56      /// </summary>
57      /// <param name="callback">The listener to remove</param>
58      public void UnRegisterListenerOnInputDown(UnityAction callback)
59      {
60          inputDownEvent.RemoveListener(callback);
61      }
62
63      /// <summary>
64      /// Removes a listener
65      /// </summary>
66      /// <param name="callback"></param>
67      public void UnRegisterListenerOnInputUp(UnityAction callback)
68      {
69          inputUpEvent.RemoveListener(callback);
70      }
71
72      /// <summary>
73      /// adds a new listener for the input-up-event
74      /// </summary>
75      /// <param name="callback"></param>
76      public void RegisterListenerOnInputUp(UnityAction callback)
77      {
78          Init();
79          if (callback != null)
80          {
81              inputUpEvent.AddListener(callback);
82          }
83      }
84
```

Figura 125 - Unità "TapNotifier" (pt.2) [23]

```
85      /// <summary>
86      /// nofiy all listeners that the input-down-event occured
87      /// </summary>
88      /// <param name="eventData"></param>
89      public void OnInputDown(InputEventData eventData)
90      {
91          if (inputDownEvent != null)
92          {
93              inputDownEvent.Invoke();
94          }
95      }
96
97      /// <summary>
98      /// nofiy all listeners that the input-up-event occured
99      /// </summary>
100     /// <param name="eventData"></param>
101     public void OnInputUp(InputEventData eventData)
102     {
103         if (inputUpEvent != null)
104         {
105             inputUpEvent.Invoke();
106         }
107     }
108 }
```

Figura 126 - Unità "TapNotifier" (pt.3) [23]

Nelle precedenti immagini viene mostrata quindi l'unità che è stata testata, nel codice di test creato ad hoc dallo sviluppatore, in cui sono presenti al suo interno tutti i singoli metodi.

11.7.2 Test di Sistema

Nel secondo caso, ovvero nel caso di test di sistema, si fa riferimento a quei test che vengono eseguiti nel momento in cui ogni singola unità è stata accuratamente testata.

Il test di sistema è considerato un metodo di test più black-box il quale viene utilizzato per valutare il sistema nel suo insieme, per garantire che soddisfi i requisiti specificati.

Le caratteristiche testate a questo livello possono essere diverse e numerose, e spesso dipendono dal tipo di applicazione che viene realizzata.

Tali test solitamente si basano sulle funzionalità espresse nelle specifiche e nei requisiti dell'applicazione stessa e possono coinvolgere diverse configurazioni software e hardware.

Anche se l'approccio utilizzato è per lo più di tipo black box, quindi orientato ai test funzionali, vengono a volte controllate anche altre caratteristiche strutturali come sicurezza, usabilità e manutenibilità.

Lo scopo di questo tipo di test è proprio quello di andare a verificare che siano soddisfatti quelli che vengono comunemente definiti requisiti prestazionali, che devono quindi essere verificati prima del rilascio dell'applicazione.

Questi test possono essere sviluppati sia dal punto di vista funzionale, che da quello strutturale, ovvero ispezionando e controllando il sistema dall'interno, considerando i tempi di esecuzione come somma dei tempi necessari per le singole operazioni.

Ciò viene effettuato con la raccolta e l'analisi di misure localizzate in diverse porzioni di codice, che facilitano la determinazione di un rallentamento dell'applicazione.

Questo tipo di test è molto utile per prevenire difetti legati alla gestione della memoria o alle dimensioni di buffer.

80 lines (69 sloc) | 3.27 KB

```
using UnityEngine;
using System.Collections;
using System.Diagnostics.Tracing;
using Base;

namespace TrilleonAutomation
{

    [AutomationClass]
    public class LoginTests : MonoBehaviour
    {
        GameObject connectBtn;
        GameObject disconnectBtn;
        GameObject keepMeConnectedToggle;
       [SetUpClass]
        public IEnumerator SetUpClass()
        {

            yield return null;

        }

        [SetUp]
        public IEnumerator SetUp()
        {
            connectBtn = Q.driver.Find(By.Name, "ConnectBtn");
            disconnectBtn = Q.driver.Find(By.Name, "DisconnectBtn");
            keepMeConnectedToggle = Q.driver.Find(By.Name, "KeepMeConnectedToggle");
            yield return null;

        }

        [Automation("Login tests")]
        [DependencyTest(1)]
        public IEnumerator UserCanSetKeepMeConnectedTest() {
            yield return StartCoroutine(Q.driver.Click(keepMeConnectedToggle, "Click keep me connected toggle."));
            yield return StartCoroutine(Q.assert.IsTrue(PlayerPrefsHelper.LoadBool("arserver_keep_connected", false), "Should be true"));
            yield return StartCoroutine(Q.driver.Click(keepMeConnectedToggle, "Click keep me connected toggle."));
            yield return StartCoroutine(Q.assert.IsTrue(!PlayerPrefsHelper.LoadBool("arserver_keep_connected", true), "Should be false"));

        }

        [DependencyTest(2)]
        [Automation("Login tests")]
        public IEnumerator UserCanConnectToServerTest()
        {
            // yield return StartCoroutine(Q.assert.IsTrue(true, "This will pass."));
            //" yield return StartCoroutine(Q.assert.IsTrue(false, "This will fail."));" yield return null;
            yield return StartCoroutine(Q.driver.Click(connectBtn, "Click connect button."));
            yield return StartCoroutine(Q.driver.WaitFor(() => GameManager.Instance.GetGameState() != GameManager.GameStateEnum.Disconnected));
            yield return StartCoroutine(Q.assert.IsTrue(MainScreen.Instance.IsActive(), "Main screen should be active after first login"));
            yield return StartCoroutine(Q.assert.IsTrue(LandingScreen.Instance.IsInactive(), "Landing screen should be inactive after first login"));
        }
```

Figura 127 - Test di Sistema (pt.1) [24]

```
54
55          [DependencyTest(3)]
56          [Automation("Login tests")]
57          public IEnumerator UserCanDisconnectFromServerTest() {
58              yield return StartCoroutine(Q.driver.Click(disconnectBtn, "Click disconnect button."));
59              yield return StartCoroutine(Q.driver.WaitFor(() => GameManager.Instance.GetGameState() == GameManager.GameStateEnum.Disconnected));
60              yield return StartCoroutine(Q.assert.IsTrue(MainScreen.Instance.IsInactive(), "Main screen should be inactive after first login"));
61              yield return StartCoroutine(Q.assert.IsTrue(LandingScreen.Instance.IsActive(), "Landing screen should be active after first login"));
62          }
63
64          [DependencyTest(4)]
65          [Automation("Login tests")]
66          public IEnumerator UserCanReconnectFromServerTest() {
67              yield return UserCanConnectToServerTest();
68          }
69
70          [TearDown]
71          public IEnumerator TearDown()
72          {
73
74              yield return null;
75
76          }
77
78          [TearDownClass]
79          public IEnumerator TearDownClass()
80          {
81
82              yield return null;
83
84          }
85
86      }
87
88  }
```

Figura 128 - Test di Sistema (pt.2) [24]

Il codice mostrato nelle immagini sovrastanti (figura 127 e 128) rappresenta un esempio di file di test di sistema trovato all'interno di un progetto.

Nello specifico questo file di test dovrebbe permettere la rilevazione di errori/bug all'interno del componente.

In questo caso vi è un file di test che permette il controllo e la verifica della funzionalità di login dell'applicazione, il quale avviene tramite la serie di metodi presenti nello script.

11.8 Test AR

Per quanto riguarda test di realtà aumentata l'unico file di test che tratta più da vicino una funzionalità dell'applicazione, e quindi intrinseca della realtà aumentata stessa, è quello che verrà mostrato nel seguente paragrafo.

Tale file di test è presente all'interno del progetto:

https://github.com/Mihir0106/AR_BasketBall

Questo script permette di testare alcune funzioni del controllo della camera utilizzata all'esecuzione di tale applicazione di realtà aumentata, in cui come descritto dallo sviluppatore stesso è un componente di test che verifica l'acquisizione dell'ultima immagine della telecamera e la sua conversione in formato RGBA.

Nel caso in questione, all'atto della verifica, nel momento in cui tale acquisizione dovesse avere successo, verrà visualizzata l'immagine sullo schermo come immagine in formato Raw[38] e verranno inoltre visualizzate anche le informazioni sull'immagine stessa.

Secondo l'autore questa è una tecnica utile per le applicazioni di visione artificiale, come queste di realtà aumentata quindi, le quali generalmente hanno bisogno di accedere ai pixel grezzi dell'immagine della telecamera sulla CPU.

Questo è diverso dal componente ARCameraBackground [25], che visualizza in modo efficiente l'immagine della telecamera sullo schermo, poiché nel

[38] La tecnica raw (in inglese "crudo", "grezzo") consiste in un particolare metodo di memorizzazione dei dati descrittori di un'immagine. Viene usata per non avere perdite di qualità della registrazione su un qualsiasi supporto di memoria, rispetto ai segnali catturati dal sensore e successivamente composti per interpolazione dal processore d'immagine della fotocamera nelle sue tre componenti fondamentali RGB (Red, Green, Blue).

momento in cui si vuole solo sfumare la trama della fotocamera sullo schermo, si utilizza ARCameraBackground o Graphics.

In questo esempio, si ottengono i dati dell'immagine della fotocamera sulla CPU, i quali vengono convertiti in un formato RGBA, visualizzati poi sullo schermo come una texture Raw per dimostrarne il funzionamento.
Una volta che abbiamo una XRCameraImage [26] valida, che rappresenta una singola immagine grezza ottenuta dalla fotocamera del dispositivo, è possibile accedere ai dati grezzi del singolo piano dell'immagine, ovvero i canali separati della stessa, ed ai metodi di conversione per la successiva conversione in formati a colori ed in scala di grigi.
Questo è considerato ovviamente come un esempio, ma non come una tecnica da utilizzare per eseguire il rendering dell'immagine della telecamera sullo schermo.
Nelle seguenti immagini (da figura 129 a 131) viene mostrato per intero il codice del file di test che è stato appena descritto:

```
30  public class TestCameraImage : MonoBehaviour
31  {
32      [SerializeField]
33      [Tooltip("The ARCameraManager which will produce frame events.")
34      ARCameraManager m_CameraManager;
35
36       /// <summary>
37      /// Get or set the <c>ARCameraManager</c>.
38      /// </summary>
39      public ARCameraManager cameraManager
40      {
41          get { return m_CameraManager; }
42          set { m_CameraManager = value; }
43      }
```

Figura 129 - Test Camera AR (pt.1) [27]

```
44
45      [SerializeField]
46      RawImage m_RawImage;
47
48      /// <summary>
49      /// The UI RawImage used to display the image on screen.
50      /// </summary>
51      public RawImage rawImage
52      {
53          get { return m_RawImage; }
54          set { m_RawImage = value; }
55      }
56
57      [SerializeField]
58      Text m_ImageInfo;
59
60      /// <summary>
61      /// The UI Text used to display information about the image on screen.
62      /// </summary>
63      public Text imageInfo
64      {
65          get { return m_ImageInfo; }
66          set { m_ImageInfo = value; }
67      }
68
69      void OnEnable()
70      {
71          if (m_CameraManager != null)
72          {
73              m_CameraManager.frameReceived += OnCameraFrameReceived;
74          }
75      }
76
77      void OnDisable()
78      {
79          if (m_CameraManager != null)
80          {
81              m_CameraManager.frameReceived -= OnCameraFrameReceived;
82          }
83      }
84
85      unsafe void OnCameraFrameReceived(ARCameraFrameEventArgs eventArgs)
86      {
87          // Attempt to get the latest camera image. If this method succeeds,
88          // it acquires a native resource that must be disposed (see below).
89          XRCameraImage image;
90          if (!cameraManager.TryGetLatestImage(out image))
91          {
92              return;
93          }
```

Figura 130 - Test Camera AR (pt.2) [27]

```
        ,
94
95          // Display some information about the camera image
96          m_ImageInfo.text = string.Format(
97              "Image info:\n\twidth: {0}\n\theight: {1}\n\tplaneCount: {2}\n\ttimestamp: {3}\n\tformat: {4}",
98              image.width, image.height, image.planeCount, image.timestamp, image.format);
99
100         // Once we have a valid XRCameraImage, we can access the individual image "planes"
101         // (the separate channels in the image). XRCameraImage.GetPlane provides
102         // low-overhead access to this data. This could then be passed to a
103         // computer vision algorithm. Here, we will convert the camera image
104         // to an RGBA texture and draw it on the screen.
105
106         // Choose an RGBA format.
107         // See XRCameraImage.FormatSupported for a complete list of supported formats.
108         var format = TextureFormat.RGBA32;
109
110         if (m_Texture == null || m_Texture.width != image.width || m_Texture.height != image.height)
111         {
112             m_Texture = new Texture2D(image.width, image.height, format, false);
113         }
114
115         // Convert the image to format, flipping the image across the Y axis.
116         // We can also get a sub rectangle, but we'll get the full image here.
117         var conversionParams = new XRCameraImageConversionParams(image, format, CameraImageTransformation.MirrorY);
118
119         // Texture2D allows us write directly to the raw texture data
120         // This allows us to do the conversion in-place without making any copies.
121         var rawTextureData = m_Texture.GetRawTextureData<byte>();
122         try
123         {
124             image.Convert(conversionParams, new IntPtr(rawTextureData.GetUnsafePtr()), rawTextureData.Length);
125         }
126         finally
127         {
128             // We must dispose of the XRCameraImage after we're finished
129             // with it to avoid leaking native resources.
130             image.Dispose();
131         }
132
133         // Apply the updated texture data to our texture
134         m_Texture.Apply();
135
136         // Set the RawImage's texture so we can visualize it.
137         m_RawImage.texture = m_Texture;
138     }
139
140     Texture2D m_Texture;
141 }
```

Figura 131 - Test Camera AR (pt.3) [27]

Capitolo 12: Pull Requests e Issue

In questo capitolo sono state affrontate le analisi delle Issues e Pulls presenti all'interno dei repositories open-source di applicazioni di realità aumentata presenti in Github.

In queste analisi sono state prese in considerazione, per lo studio statistico, sia quelle aperte che quelle chiuse al fine di valutare numericamente i loro andamenti, interessi e cercare di capire quali possano essere i principali ed eventuali problemi riscontrati dall'utenza nell'utilizzo di queste applicazioni.

Come proceduto nei capitoli precedenti sono stati sviluppati degli script, contenenti delle funzioni e delle query, appositamente per poter effettuare queste ricerche, col fine di riuscire a diramare delle statistiche su questi progetti.

Questi script sono disponibili anch'essi nel repository di Github contenente tutti i file python e Matlab, in cui i primi possono essere eseguiti tranquillamente sia da terminale che in un ambiente dedicato all'esecuzione di script python, mentre per i secondi sarà necessario avere a disposizione il software Matlab installato sul proprio dispositivo.

12.1 Analisi Pull Requests

Le Pulls presenti in Github non sono altro che dei "problemi", ma non necessariamente devono esserlo, così come non tutti devono essere considerate delle richieste di Pull.

A tal proposito, le azioni che vengono condivise per entrambe le funzionalità come, ad esempio, la manipolazione di etichette e traguardi vengono fornite direttamente dalle Issues.

Le Pull Requests permettono di poter comunicare quindi con gli altri utenti, ed avvisare o aggiornare loro delle modifiche che sono state inviate in un determinato punto del repository della propria applicazione su Github.

Pertanto, una volta che una Pull Requests viene aperta, gli utenti possono discutere ed eventualmente rivedere le potenziali modifiche con gli sviluppatori stessi, così da provvedere ad eventuali nuovi commit prima ancora che le modifiche vengano unite definitivamente al progetto di base.

Altri utenti, quindi, possono valutare le modifiche che sono state proposte, aggiungendo commenti, e contribuendo alla discussione sulla Pull Requests.

Le Pull Requests sono considerate a tutti gli effetti il cuore della collaborazione che può avvenire su GitHub, poiché quando una di queste discussioni viene aperta, si stanno proponendo delle modifiche e chiedendo inoltre che qualcuno possa partecipare alla revisione, aiutando con il proprio contributo.

12.1.1 Ricerca Pulls Requests

Come è stato fatto per le ricerche svolte precedentemente, anche in questo caso sono stati creati degli script, in Python, ad hoc, sfruttando le Api messe a disposizione da Github.

Per poter effettuare una analisi dei Pulls è stata dapprima sviluppata una funzione che permette di effettuare una ricerca all'interno dei repositories delle applicazioni di realtà aumentata.

In questa funzione è stato dato l'elenco dei repositories come input, è la stessa non farà altro che eseguire ciclicamente delle query, le quali preleveranno le informazioni necessarie, da salvare successivamente in delle apposite liste.

	ELEMENTO	DESCRIZIONE
NOME	RICERCA INFO PULLS	Questa funzione permette di poter effettuare la ricerca delle pulls e delle informazioni inerenti ad esse all'interno dei repositories di applicazioni di realtà aumentata.
TIPOLOGIA	FUNZIONE	
INPUT	INFO	Elenco dei nomi dei progetti di realtà aumentata.
OUTPUT	PULLS INFO OPEN	Lista contenente tutte le informazioni sulle pulls aperte trovate all'interno di ogni repository.
	PULLS INFO CLOSE	Lista contenente tutte le informazioni sulle pulls chiuse trovate all'interno di ogni repository.
ESEMPIO D'USO	Funzioni.ricerca_info_pulls(info)	

Tabella 32 - Informazioni funzione "ricerca info pulls"

La funzione appena descritta viene raffigurata nell'immagine seguente (Figura 132 e 133):

```
def ricerca_info_pulls(info):
    pulls_info_open = []
    pulls_info_close = []
    count = 0
    i = 0

    for id in info[0]:
        """if count == 50:
            i = i + 1
            count = 0
            print("Pulls del progetto salvate mancano " + str(len(info[0])
- i * 50) + " progetti.")"""

        pulls_open = requests.get('https://api.github.com/repos/' + id +
'/pulls?page=1&per_page=100',
                                  auth=('enzot95', ACCESS_TOKEN)).json()
```

Figura 132 - Funzione "Ricerca_info_pulls" (pt.1)

```
    if len(pull_close) != 0:
        for pull in pull_close:
            if len(pull["labels"]) != 0:
                pulls_info_close.append(
                    {"title": pull["title"], "body": pull["body"], "la-
bels": pull["labels"][0]["name"]})
            if len(pull["labels"]) == 0:
                pulls_info_close.append({"title": pull["title"], "body":
pull["body"], "labels": pull["labels"]})

    count = count + 1

return pulls_info_open, pulls_info_close
```

Figura 133 - Funzione "Ricerca_info_pulls" (pt.2)

Una volta ottenute le informazioni riguardati le Pulls sia aperte che chiuse, queste sono state date in ingresso un'altra funzione che provvederà al filtraggio di queste informazioni sistemandole in diversi file Json che conterranno le informazioni utili allo studio statistico che si ha come obiettivo.

	ELEMENTO	***DESCRIZIONE***
NOME	PULIZIA INFO PULLS	Questa funzione permette di poter effettuare la pulizia ovvero un filtraggio delle pulls e delle informazioni come le labels inerenti ad esse.
TIPOLOGIA	FUNZIONE	
INPUT	INFOS	Elenco delle informazioni sulle pulls trovate all'interno dei progetti di realtà aumentata.
	TIPO	Parola chiave che serve per distinguere le pulls aperte da quelle chiuse.
OUTPUT	BODY	Lista contenente tutte le informazioni sul testo delle pulls trovate all'interno di ogni repository.
	TITLE	Lista contenente tutte le informazioni sui titoli delle pulls trovate all'interno di ogni repository.
	LABELS	Lista contenente tutte le informazioni sulle labels delle pulls trovate all'interno di ogni repository.
ESEMPIO D'USO	Funzioni.pulizia_info_pulls(infos, tipo)	

Tabella 33 - Informazioni funzione "pulizia info pulls"

Le informazioni che verranno salvate in appositi file Json sono:

- Body, ovvero il corpo/testo della Pull di riferimento;
- Title, il titolo della Pull;
- Labels, le informazioni dell'eventuale etichetta presente nella Pull, nella quale verrà prelevato soltanto il nome.

```
def pulizia_info_pulls(infos,tipo):

    count =0
    body=[]
    title=[]
    labels=[]

    #Prelievo e pulizia delle informazioni delle pulls
    for info in infos:
        count = count + 1
        if count!= 277|count!=279:
            if len(info["body"])!=0:
                body.append(info["body"])
            if len(info["title"]) != 0:
                title.append(info["title"])
            if len(info["labels"]) != 0:
                if info["labels"][0:3]=='bug':
                    labels.append('Bug')
                elif info["labels"][0:3] == 'Bug':
                    labels.append('Bug')
                elif info["labels"][0:3]=='enh':
                    labels.append('Enhancement')
                elif info["labels"][0:3]=='Enh':
                    labels.append('Enhancement')
                else:
                    labels.append(info["labels"])
```

Figura 134 - Funzione "pulizia_info_pulls" (pt.1)

```
print("Le pulls "+tipo+" che presentano delle etichette sono " +
str(len(labels)) + " su " + str(len(infos)) +
      " con una percentuale del " + str(round(len(labels) * 100 /
len(infos))) + " % sul totale.")
print("Le pulls "+tipo+ " che presentano delle titoli sono " +
str(len(title)) + " su " + str(len(infos)) +
      " con una percentuale del " + str(round(len(title) * 100 /
len(infos))) + " % sul totale.")
print("Le pulls "+tipo+" che presentano dei body sono " + str(len(body)) +
" su " + str(len(infos)) +
      " con una percentuale del " + str(round(len(body) * 100 /
len(infos))) + " % sul totale.")

return body,title,labels
```

Figura 135 - Funzione "pulizia_info_pulls" (pt.2)

Nelle immagini precedenti (figura 134 e 135) viene mostrato il codice della funzione che permette di effettuare il filtraggio delle informazioni delle Pull Requests.

A questo punto le due funzioni appena descritte verranno eseguite in un apposito script che consentirà di effettuare quindi la ricerca ed il filtraggio delle Pull Requests sia aperte che chiuse.

	ELEMENTO	***DESCRIZIONE***
NOME	INFO PULLS	Questo script consente di ottenere l'elenco filtrato delle informazioni inerenti alle pulls aperte e chiuse, divisi per file che riguardano separatamente l'elenco dei titoli, testi e labels.
TIPOLOGIA	SCRIPT	
INPUT DATI	INFO	Elenco dei nomi dei progetti in cui ricercare.
	KEYS	Parole chiavi che servono per distinguere le pulls aperte da quelle chisue.
FUNZIONI USATE	RICERCA INFO PULLS	Funzione di ricerca delle informazioni inerenti alle pulls nei repositories
	PULIZIA INFO PULLS	Funzione di filtraggio delle informazioni delle pulls sia aperte che chiuse.
	SALVA REPOSITORIES TWO TAGS	Funzione di salvataggio delle informazioni ricercate, memorizzate in un apposito file json.
OUTPUT	INFORMAZIONI	Elenco di informazioni visibili tramite console inerenti ai repositories ricercati.
	FILE JSON	File Json creati contenenti le varie informazioni ricercate.
ESEMPIO D'USO	Info_pulls.py (da eseguire da riga di comando)	

Tabella 34 - Informazioni script "info pulls"

Lo script che esegue queste funzioni è raffigurato nella seguente immagine (figura 136).

```
import json
import funzioni

info = json.load(open("../lista_ids_unity_augmented_reality.json"))

#Ricerca e salvataggio delle pulls all'interno dell'elenco dei repositories
pulls_info_open,pulls_info_close= funzioni.ricerca_info_pulls(info)
funzioni.salva_repositories_two_tag(pulls_info_open, "pulls_info_open",
"unity", "augmented_reality")
funzioni.salva_repositories_two_tag(pulls_info_close, "pulls_info_close",
"unity", "augmented_reality")
print(" ")

keys =['open','close']
for key in keys:

    pulls =
json.load(open("pulls_info_"+key+"_unity_augmented_reality.json"))
    body,title,labels= funzioni.pulizia_info_pulls(pulls, key)

    #Salvataggio dei file puliti
    funzioni.salva_repositories_two_tag(body, "pulls_testo_" + key, "uni-
ty", "augmented_reality")
    funzioni.salva_repositories_two_tag(title, "pulls_titolo_" + key, "uni-
ty", "augmented_reality")
    funzioni.salva_repositories_two_tag(labels, "pulls_labels_" + key,
"unity", "augmented_reality")
    print(" ")
```

Figura 136 - Script per la ricerca delle Pulls

Una volta eseguito lo script verranno creati e salvati i file Json che conterrà tutte le informazioni riguardanti le Pull Requests trovate nei repositories, ed anche i singoli file Json contenenti tutti i singoli attributi che serviranno per l'analisi dei dati da fare con Matlab, sia per le Pulls aperte che per quelle chiuse.

Nella successiva immagine (figura 137) sarà possibile vedere l'output ottenuto dall'esecuzione dello script:

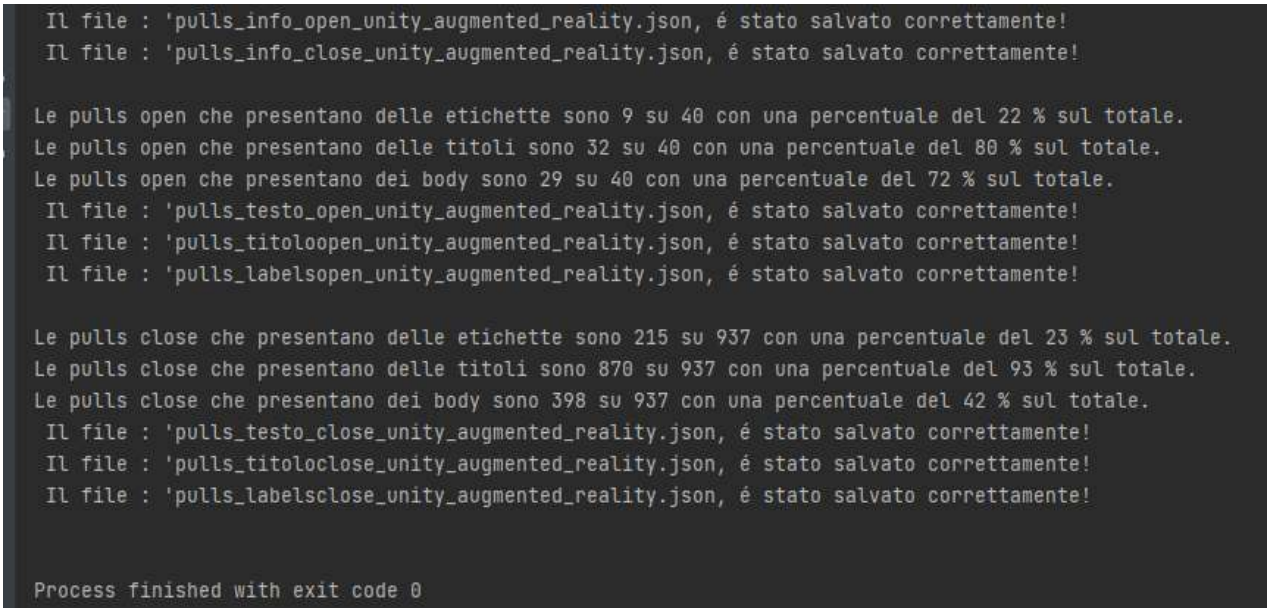

Figura 137 - Output esecuzione script ricerca delle Pull Requests

Come è possibile notare nella precedente figura, sono stati riportati e calcolati anche alcuni dati numerici che riguardano le Pull Requests trovate.

Sono state rinvenute un totale di 977 Pull Requests, in cui soltanto 40 sono quelle ancora aperte, mentre le restanti 937 riguardano quelle già chiuse, raggiungendo una percentuale di 96% circa.

È inoltre possibile notare come solo 224 sul totale delle Pulls presentano l'etichetta che le contraddistingue, oltre a catalogarle, spingendo ad una percentuale di quasi 23%, poco meno dell'un quarto.

Altro dato interessante riguarda il numero di testi, ovvero il corpo della discussione, che sono stati rinvenuti all'intero di questa ricerca, in cui si ha una percentuale del 43,7%, precisamente solo 427 Pull Requests hanno un "body".

Nella successiva immagine (figura 138) viene mostrato un estratto del file json contenete tutte le informazioni generali delle Pulls Requests che viene generato dopo l'esecuzione del relativo script della loro ricerca:

```
[
    {
        "body": "- [x] Fix `CanvasPlayer` so the `AudioSource` is not attached to a `RectTransform`
        object (fixes #85)\r\n- [x] Get rid of `PS2` as a build target (fixes #82)\r\n\r\nTested
        against:\r\n- [x]
        `2019.1.0f1`\r\n- [x] `2018.2.14f1`\r\n- [x] `2017.1.5f1`",
        "labels": "bug \ud83d\udc1b",
        "title": "Unity 2019 Fixes"
    },
    {
        "body": "This feature enables developer-level partners to implement play logging of videos\
        r\n\r\nTested against (w & w/o plugins):\r\n- [x] 2018.3.6f1\r\n- [x] 2018.2.14f1\r\n-
        [x] 2017.1.4f1",
        "labels": "enhancement \ud83d\ude80",
        "title": "Play logging"
    },
    {
        "body": "Using this PR (https://github.com/vimeo/vimeo-unity-sdk/pull/49) the current PR formats our
        code-base using the Omnisharp config file",
        "labels": "enhancement \ud83d\ude80",
        "title": "Formatting the code-base based on the Omnisharp config file"
    },
    {
        "body": "Solves #79 by adding an `AudioListener` to the scene\r\n\r\nTested against:\r\n-
        2018.2.1.18f1\r\n- 2017.4.16f1\r\n- 5.6.4f1",
        "labels": "bug \ud83d\udc1b",
        "title": "Adding an audio listener to the VimeoPlayer integration tests scene"
    },
    {
        "body": "**What?**\r\n- Adding an `Unfurl` coroutine to the Vimeo Player so that when it get's
        adaptive streams from the API it unfurls them to the right manifest destination. Fixes #56\r\n-
        Adding basic tests to check that `Unfurl` is unfurling when it should/should not\r\n\r\nTested
        against:\r\n- Unity 2018.2.18f1\r\n- Unity 2017.4.16f1",
        "labels": "bug \ud83d\udc1b",
        "title": "Unfurl adaptive video file links"
    },

        "body": "I was missing a handler for cases where the Vimeo upload would fail (I remember I had a
        \"failed to transmit data\" error)\r\n\r\n**What:**\r\n- Adding error event bubbling for upload
        errors to `VimeoRecorder`\r\n- Fixing problem with `VideoChunk` using `IsNetworkError` that doesn't
        support Unity 2017.1 and lower\r\n\r\n**Tested against:**\r\n- 2018.2.1.18f1\r\n- 2017.4.16f1\r\n-
        5.6.4f1",
        "labels": "enhancement \ud83d\ude80",
        "title": "Feature/upload errors"
    },
    {
        "body": "A suggested fix for issue #51 , to remove ambiguity conflict with a JSON library like Matt
        Shoen's: don't use the overloaded JSON symbol, and instead use directly the JSONNode class:\r\n\r\nJSONNode
        node = JSONNode.Parse(jsonString);\r\nfeels better than:\r\nJSONNode node = JSON.Parse(jsonString);\r\n",
        "labels": "enhancement \ud83d\ude80",
        "title": "Fix JSON namespace/class ambiguity"
    },
    {
        "body": "\u2026 Matt Shoen,s: don't use the overloaded JSON symbol, and use a more explicit JSONNode class
        reference\r\n\r\nJSONNode node = JSONNode.Parse(jsonString);\r\nmakes more sense than:\r\nJSONNode node =
        JSON.Parse(jsonString);",
        "labels": [],
        "title": "A suggested fix to remove ambiguity conflict with a JSON library like\u2026"
    },
```


Figura 138 - File json contenente le informazioni delle Pulls

12.1.2 Analisi Pulls con Matlab

Una volta ottenute e filtrate le informazioni riguardanti le Pull Requests, sono stati creati degli script in Matlab che permettessero di poter analizzare dal punto di vista statistico tali attributi.

Pertanto, sono stati creata una funzione che realizzasse il Word Cloud per la rappresentazione sia grafica che concettuale per descrivere al meglio la frequenza delle parole presenti all'interno sia dei testi che dei titoli delle Pulls.

Il Word Cloud, detta anche nuvola di parole, o di etichette, non è nient'altro che una rappresentazione visiva delle parole che sono presenti all'interno di un testo.

L'obiettivo è quello di eliminare la punteggiatura e tutti i caratteri speciali al suo interno in modo da avere soltanto le parole presenti nel testo, creando quindi una lista di parole.

Generalmente questa lista viene mostrata in ordine alfabetico, dall'alto verso il basso, mostrando all'interno della figura le parole di maggiore importanza di una dimensione maggiore o con un colore diverso.

Nel nostro caso oltre al filtraggio si è puntato a dare maggiore importanza alle parole più frequenti, in modo da avere come impatto visivo subito a nostra disposizione le parole più presenti all'interno dei testi delle Pulls Requests che stiamo analizzando.

Pertanto, al fine di poter realizzare questo tipo di rappresentazione, è stata creata una funzione Matlab, che innanzitutto filtra i testi delle Pulls Requests rimuovendone tutta la punteggiatura, e calcolandone l'occorrenza delle parole più frequentemente utilizzate all'interno dei testi.

Ovviamente, oltre al filtraggio della punteggiatura, si è optato di filtrare e quindi di non considerare tutte le parole di piccole dimensioni, escludendole dal conteggio, poiché queste generalmente corrispondevano a verbi, preposizioni, articoli, etc., ovvero quindi parole del tutto fuorvianti per lo scopo.

Quindi il codice, che permette la creazione del Word Cloud per le Pulls, è mostrato nella seguente immagine (figura 139):

```
function wordcloud_ar(testo,titolo)

%rimozione della punteggiatura
punctuationCharacters =
  ["." "?" "!" "," ";" ":" "#" "'" "_" "[" "]" "<" ">" "@" "/" "-" "=" "(" ")"];
testo = replace(testo,punctuationCharacters," ");
words = split(join(testo));
words(strlength(words)<6) = [];
words = lower(words);
words(1:10);
C = categorical(words);

%creazione della figura di word cloud
wordcloud(C);
title(titolo)

end
```

Figura 139 - Funzione Word Cloud

Questa funzione è stata quindi poi utilizzata nello script mostrato nella successiva immagine (figura 140), in cui è stato permesso di poter realizzare il Word Cloud sia per i titoli che per i testi di tutte le Pulls, separandole in aperte e chiuse.

In questo script sono state inoltre calcolate e raffigurate tramite la generazione di grafici a barre, l'occorrenza delle etichette che sono state trovate all'interno delle Pulls.

```
clc; close all;clear all;

key=["open","close"];
word=["labels","titolo","testo"];

for i=1:2,
    for j=1:3,

 file(j)='pulls_'+word(j)+'_'+key(i)+'_unity_augmented_reality.json';

        % Lettura dei files contenenti le varie info dei pulls
        str = char(fread(fopen(file(j)),inf)');
        fclose(fopen(file(j)));
        val = string(sort(jsondecode(str)));

        if strcmp(word(j),"labels")==1
            vettore=conta_occorrenze(length(val),val);
            grafico(vettore,"AR","Labels "+key(i)+"
 Pulls","Label","Occorrenze");
        elseif strcmp(word(j),"titolo")==1
            figure;
            wordcloud_ar(string(val),'Word Cloud Titoli Pulls
 ('+key(i)+')');
            saveas(gcf,'Word_Cloud_Testo_Pulls_'+key(i)+'.png');
        else
            figure;
            wordcloud_ar(string(val),'Word Cloud Testo Pulls
 ('+key(i)+')');
            saveas(gcf,'Word_Cloud_Testo_Pulls_'+key(i)+'.png');
        end
    end
end
```

Figura 140 – Script analisi Pulls

Una volta eseguito il precedente script sono stati generati ben due grafici a barre e quattro Word Cloud, i quali vengono mostrati nelle sei immagini che seguono (dalla figura 141 alla figura146).

Come è possibile vedere nei primi due grafici, che mostrano il numero di occorrenze delle etichette, le etichette più presenti in linea generale in tutte le Pulls considerando sia quelle aperte che quelle chiuse sono in ordine gli “Enhancement”, “Ready for review” ed i “Bug”.

Solo queste tre tipologie di etichette rappresentano il quasi 84% di tutte le Pulls etichettate.

Questo a dimostrazione del fatto che fondamentalmente le Pulls Requests vengono utilizzate principalmente per la discussione di modifiche o aggiornamenti delle applicazioni.

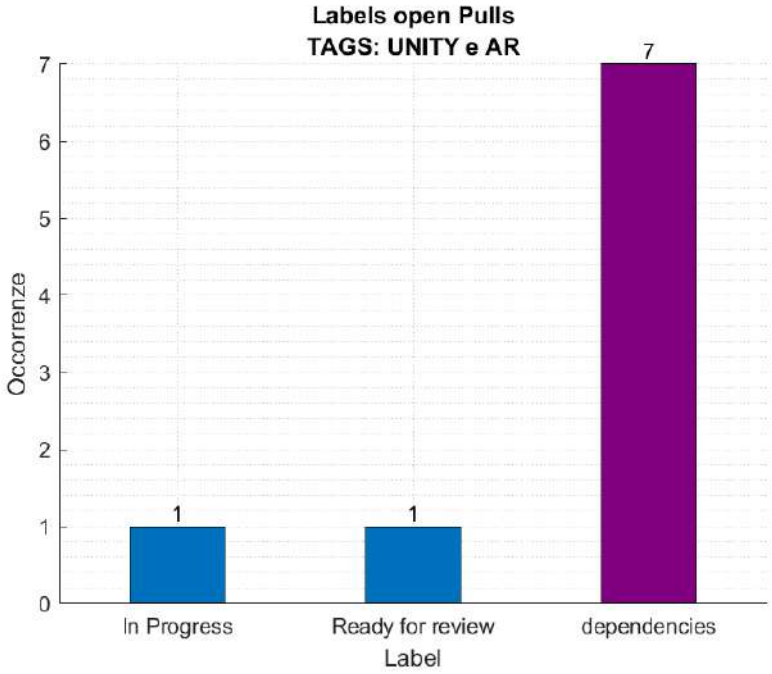

Figura 141 - Labels Open Pulls

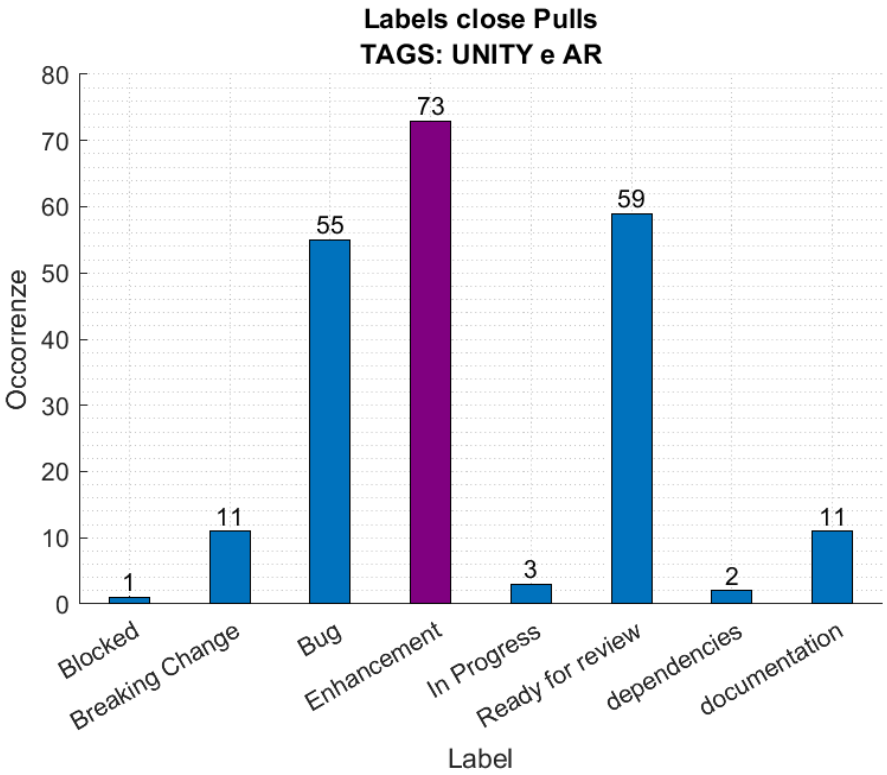

Figura 142 - Labels Close Pulls

In queste successive immagini (figure da 136 a 140) vengono mostrati i quattro Word Cloud generati dallo script Matlab precedentemente illustrato:

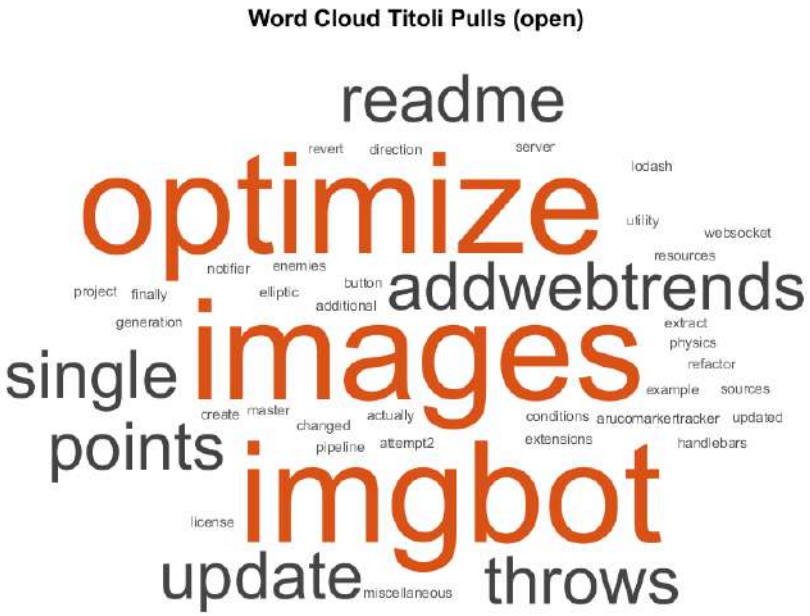

Figura 143 - Word Cloud Titoli Open Pulls

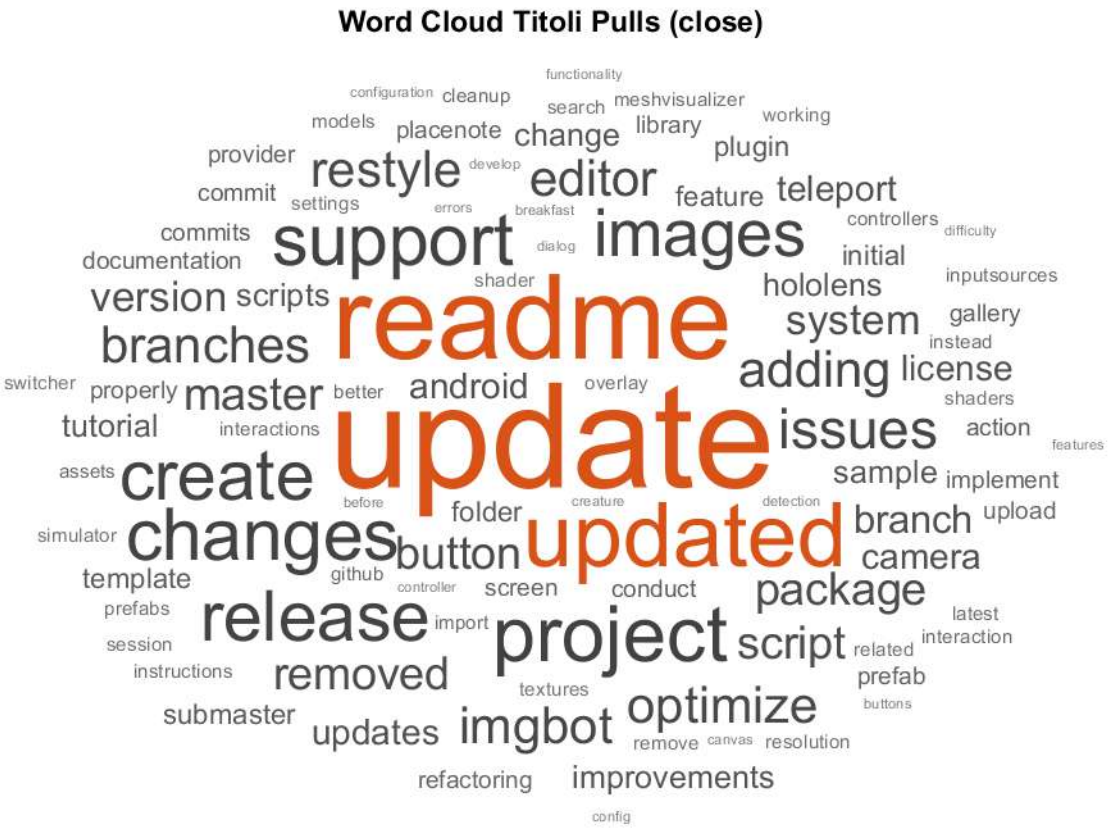

Figura 144 - Word Cloud Titoli Close Pulls

In questi Word Cloud riguardanti i titoli delle Pulls è possibile vedere delle parole più comuni, le quali vengono brevemente riassunte nella tabella qui sotto (tabella 35):

TOP 10 PAROLE PULLS (TITOLO)			
OPEN	***#***	***CLOSE***	***#***
optimize	4	update	80
images	***4***	readme	50
imgbot	4	updated	31
update	3	project	27
single	2	support	24
points	2	create	23
throws	2	changes	22
readme	2	release	22
utility	1	***images***	***19***
button	***1***	master	17

Tabella 35 - Top 10 Parole Titolo Pulls

Da notare che nella tabella vengono messe in grassetto le parole che possono essere associate ad argomenti riguardanti la realtà aumentata.

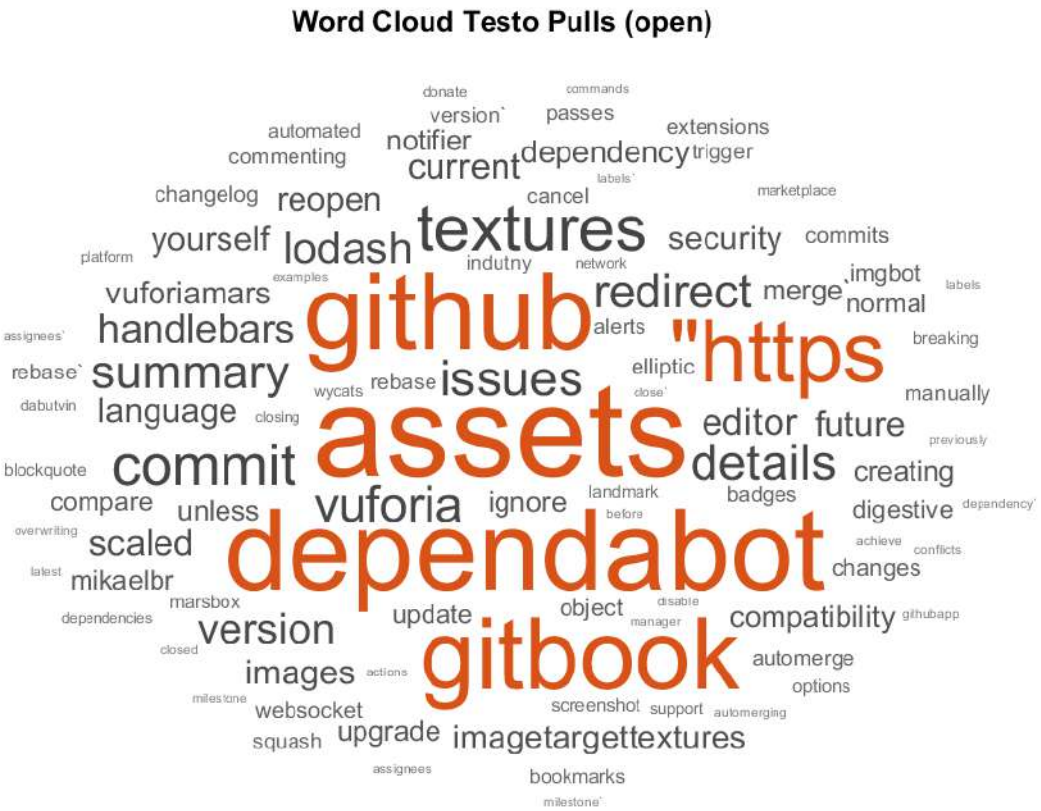

Figura 145 - Word Cloud Testo Open Pulls

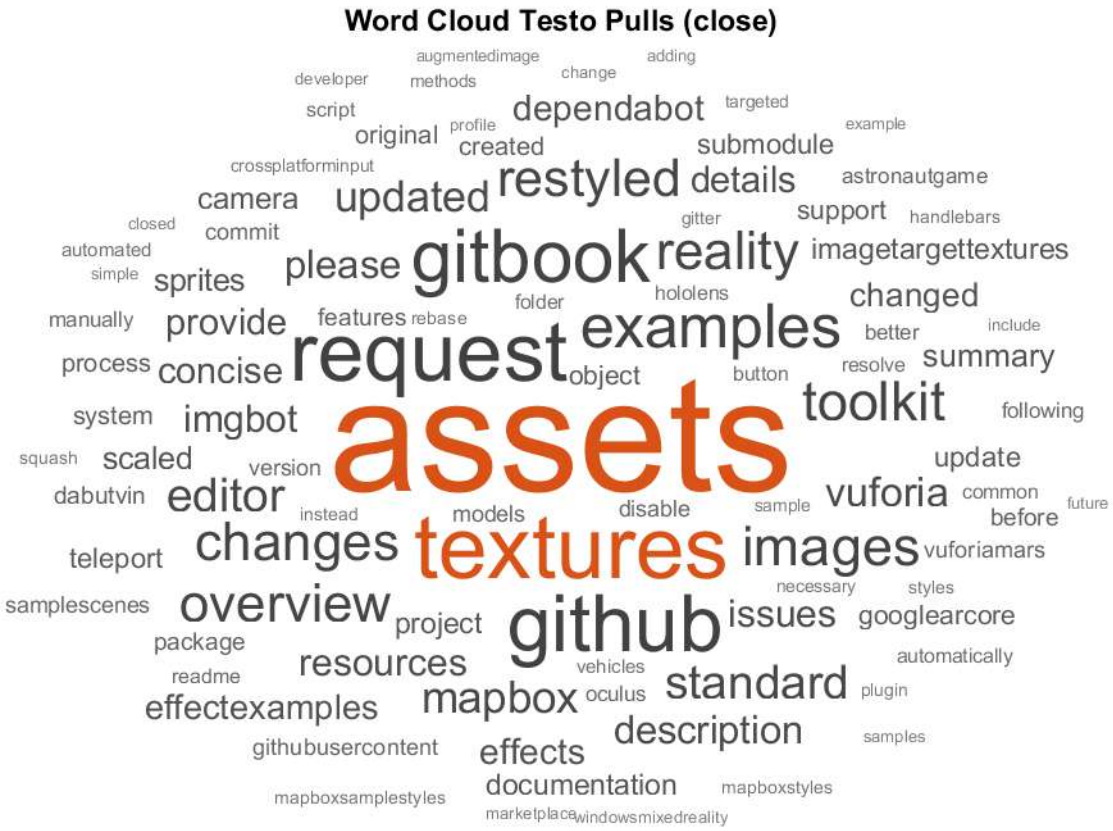

Figura 146 - Word Cloud Testo Close Pulls

In queste altre due word Cloud vengono invece mostrate le parole più frequenti all'interno dei testi delle Pulls, le quali parole vengono ulteriormente sintetizzate nella tabella seguente (tabella 36).

In essa sono presenti sia le Pulls aperte che chiuse, rispettivamente le prime nelle colonne di sinistra, mentre le seconde nelle colonne a destra.

TOP 10 PAROLE PULLS (TESTO)			
OPEN	***#***	***CLOSE***	***#***
assets	305	assets	715
dependabot	235	***texture***	***286***
gitbook	182	Requests	224
texture	***116***	github	217
commit	83	gitbook	169
vuforia	***80***	examples	164
issues	55	***reality***	***125***
summary	50	***toolkit***	***101***
security	***28***	changes	95
language	***28***	overview	94

Tabella 36 - Top 10 Parole Testo Pulls

12.2 Analisi Issues

Una ulteriore analisi, che è stata pensata, è quella di cercare di analizzare i testi delle issues, in modo da cercare di carpirne il contenuto e valutare se in esse vi sono dei particolari temi trattati ricorrenti, oppure valutare gli errori più frequenti su cui nascono delle discussioni.

In questo paragrafo vengono mostrati quindi i procedimenti che hanno portato all'analisi delle Issues presenti all'interno dei repositories trovati di realtà aumentata.

Questa analisi è stata condotta sulla falsa riga di quella fatta per le Pulls Requests descritta nel paragrafo precedente (12.1).

Il focus di questa analisi saranno, come accennato prima, le Issues che sono presenti su Github, dove anche queste vengono suddivise in aperte e chiuse, in cui le prime rappresentano sono quelle discussioni / topic che sono ancora attivi, quindi che sono ad esempio in caso di errori eventualmente ancora in fase di visione.

Nel secondo caso invece ci sono quelle che hanno trovato una soluzione al loro problema, dell'argomento della issues.

Le Issues, in italiano "problemi", sono un ottimo modo per tenere traccia di attività, miglioramenti e bug per i tuoi progetti.

La maggior parte dei progetti software ha un bug tracker di qualche tipo, in questo caso, ovvero nel caso di GitHub, il suo tracker sono proprio le Issues, le quali come accennato prima hanno una propria sezione in ogni repository.

Nel dettaglio queste issues hanno la possibilità di essere etichettate, secondo una categorizzazione scelta dagli utenti, che aprono quella issues, il che permette di poter identificare meglio il senso e l'appartenenza della issue mossa.

Per poter provare a fare, quindi, una sorta di analisi e cercare di capire, come è stato fatto anche nel caso delle Pulls, se sono presenti delle parole frequenti a cui possono essere ricollegati degli argomenti trattati in esse.

12.2.1 Ricerca delle Issues

Anche in questo caso è stato necessario dover sviluppare uno script che consentisse la ricerca delle informazioni delle issues sia aperte che chiuse presenti nei repositories di realtà aumentata.

Pertanto, partendo dall'avere come input iniziale l'elenco completo di tutti i progetti di realtà aumentata a cui si vogliono analizzare le issues, lo script di ricerca sfrutta due ricerche parallele, per ogni progetto, dove vengono create due liste di informazioni, una per le issues aperte ed una per quelle chiuse.

Le informazioni, che sono state considerati importanti da voler provare ad analizzare, sono il titolo ed il body delle issues.

	ELEMENTO	***DESCRIZIONE***
NOME	RICERCA INFO ISSUES	Questa funzione permette di poter effettuare la ricerca delle issues e delle informazioni inerenti ad esse all'interno dei repositories di applicazioni di realtà aumentata.
TIPOLOGIA	FUNZIONE	
INPUT	LISTA ID	Elenco dei nomi dei progetti di realtà aumentata.
OUTPUT	INFO ISSUES OPEN	Lista contenente tutte le informazioni sulle issues aperte trovate all'interno di ogni repository.
	INFO ISSUES CLOSE	Lista contenente tutte le informazioni sulle issues chiuse trovate all'interno di ogni repository.
ESEMPIO D'USO	Funzioni.ricerca_info_issues(lista_id)	

Tabella 37 - Informazioni funzione " ricerca info issues"

La funzione che consente la ricerca di queste informazioni è la seguente (figura 147 e 148):

```
def ricerca_info_issues(lista_id):
    # Ricerca dei titoli e del contenuto delle issues dei repositories tro-
vati
    info_issues_open = []
    info_issues_close = []
    for id in lista_id[0]:

        issues_open = requests.get('https://api.github.com/repos/' + id +
'/issues?page=1&per_page=100',
                                   auth=('enzot95', ACCESS_TOKEN)).json()

        if len(issues_open) != 0:
            for issue in issues_open:
                if len(issue["labels"]) != 0:
                    info_issues_open.append({"repos": id, "title": is-
sue["title"], "body": issue["body"], "labels":
                        issue["labels"][0]["name"], "comments": is-
sue["comments"]})
                else:
                    info_issues_open.append({"repos": id, "title": is-
sue["title"], "body": issue["body"], "labels":
                        issue["labels"], "comments": issue["comments"]})

        issues_close = requests.get('https://api.github.com/repos/' + id +
'/issues?state=closed&page=1&per_page=100',
                                    auth=('enzot95', ACCESS_TOKEN)).json()
```

Figura 147 - Funzione "ricerca_info_issues"(pt.1)

```
    if len(issues_close) != 0:
        for issue in issues_close:
            if len(issue["labels"]) != 0:
                info_issues_close.append({"repos": id, "title": is-
sue["title"], "body": issue["body"], "labels":
                    issue["labels"][0]["name"], "comments": is-
sue["comments"]})
            else:
                info_issues_close.append({"repos": id, "title": is-
sue["title"], "body": issue["body"], "labels":
                    issue["labels"], "comments": issue["comments"]})

print('La ricerca delle informazioni sulle issues dei repositories è stata
effettuata con successo!')

print("Sono state trovate " + str(len(info_issues_open)) + " issues aper-
te.")
print("Sono state trovate " + str(len(info_issues_close)) + " issues chiu-
se.")

return info_issues_open, info_issues_close
```

Figura 148 - Funzione "ricerca_info_issues"(pt.2)

Nello script che effettuerà la ricerca delle issues viene utilizzata inoltre una particolare funzione che permette di fare un filtraggio, ovvero come visto anche per le Pulls, viene effettuata una pulizia dei testi delle issues.

L'obiettivo della funzione è quello di fare in modo da eliminare all'interno di essi elementi che non sono utili alla comprensione del testo, come:

- i link di informazioni e/o immagini,
- i paths dei file a cui ci si fa riferimento,
- altri particolari caratteri che non risultano necessari.

	ELEMENTO	***DESCRIZIONE***
NOME	PULIZIA INFO ISSUES	Questa funzione permette di poter effettuare la pulizia ovvero un filtraggio delle issues e delle informazioni come le labels inerenti ad esse.
TIPOLOGIA	FUNZIONE	
INPUT	INFOS	Elenco delle informazioni sulle issues trovate all'interno dei progetti di realtà aumentata.
	TIPO	Parola chiave che serve per distinguere le issues aperte da quelle chiuse.
OUTPUT	BODY	Lista contenente tutte le informazioni sul testo delle issues trovate all'interno di ogni repository.
	TITLE	Lista contenente tutte le informazioni sui titoli delle issues trovate all'interno di ogni repository.
	LABELS	Lista contenente tutte le informazioni sulle labels delle issues trovate all'interno di ogni repository.
ESEMPIO D'USO	Funzioni.pulizia_info_issues(infos, tipo)	

Tabella 38 - Informazioni funzione "pulizia info issues"

La funzione appena descritta è la seguente (figura 149 e 150):

```
def pulizia_info_issues(lista_issues, tipo):

    elenco_labels=[]
    title_issues=[]
    body_issues=[]

    for issue in lista_issues:
        if len(issue["labels"])!=0:
            labels=issue["labels"]
            if labels[:3]=='bug'or labels[:3]=='Bug':
                elenco_labels.append('bug')
            elif labels[:3]=='enh' or labels[:3]=='Enh':
                elenco_labels.append('enhancement')
            elif labels[:3]=='doc':
                elenco_labels.append('documentation')
            elif labels[:3]=='Spr':
                elenco_labels.append('sprint')
            else:
                elenco_labels.append(labels)
```

Figura 149 - Funzione "pulizia_info_issues" (pt.1)

```
for issue in lista_issues:
    if len(issue["title"]) != 0:
        title_issues.append(issue["title"])
    if issue["body"] != None:
        if len(issue["body"]) != 0:
            body_issues.append(issue["body"])

print("Le issues "+tipo+" che presentano delle etichette sono " +
str(len(elenco_labels)) + " su " + str(len(lista_issues)) +
      " con una percentuale del " + str(round(len(elenco_labels) * 100 /
len(lista_issues))) + " % sul totale.")
print("Le issues "+tipo+ " che presentano delle titoli sono " +
str(len(title_issues)) + " su " + str(len(lista_issues)) +
      " con una percentuale del " + str(round(len(title_issues) * 100 /
len(lista_issues))) + " % sul totale.")
print("Le issues "+tipo+" che presentano dei body sono " +
str(len(body_issues)) + " su " + str(len(lista_issues)) +
      " con una percentuale del " + str(round(len(body_issues) * 100 /
len(lista_issues))) + " % sul totale.")

return elenco_labels,title_issues, body_issues
```

Figura 150 - Funzione "pulizia_info_issues" (pt.2)

A questo punto una volta descritte le funzioni che servono per l'esecuzione della ricerca delle issues, nelle immagini seguenti (figura 145 e 146) viene mostrato lo script che permette di fare ciò.

	ELEMENTO	DESCRIZIONE
NOME	INFO ISSUES	Questo script consente di ottenere l'elenco filtrato delle informazioni inerenti alle issues aperte e chiuse, divisi per file che riguardano separatamente l'elenco dei titoli, testi e labels.
TIPOLOGIA	SCRIPT	
INPUT DATI	INFO	Elenco dei nomi dei progetti in cui ricercare.
	KEYS	Parole chiavi che servono per distinguere le issues aperte da quelle chisue.
FUNZIONI USATE	RICERCA INFO ISSUES	Funzione di ricerca delle informazioni inerenti alle issues nei repositories
	PULIZIA INFO ISSUES	Funzione di filtraggio delle informazioni delle issues sia aperte che chiuse.
	SALVA REPOSITORIES TWO TAGS	Funzione di salvataggio delle informazioni ricercate, memorizzate in un apposito file json.
OUTPUT	INFORMAZIONI	Elenco di informazioni visibili tramite console inerenti ai repositories ricercati.
	FILE JSON	File Json creati contenenti le varie informazioni ricercate.
ESEMPIO D'USO	Info_issues.py (da eseguire da riga di comando)	

Tabella 39 - Informazioni script "info issues"

Questo non fa altro che eseguire gli script per la ricerca ed il filtraggio delle informazioni delle issues, per poi salvare tutte queste in dei relativi file json dedicati.

```
import funzioni
import json

# In questo script sarà possibile fare una ricerca di tutte le issues sia
aperte che chiuse, dato in input l'elenco
# dei repositories in cui si vuole ottenere tali informazioni.
# Sarà possibile inoltre effettuare una pulizia e divisione delle informa-
zione nella seconda parte dello script.

tag = "augmented_reality"
lista_id=json.load(open("../lista_ids_unity_augmented_reality.json"))
```

Figura 151 – Script ricerca delle issues (pt.1)

```
info_issues_open,info_issues_close= funzioni.ricerca_info_issues(lista_id)
funzioni.salva_repositories_two_tag(info_issues_open, "informazio-
ni_issues_open", "unity", tag)
funzioni.salva_repositories_two_tag(info_issues_close, "informazio-
ni_issues_close", "unity", tag)

types =['open','close']

for type in types:
    li-
sta_info=json.load(open("informazioni_issues_"+type+"_unity_augmented_reali
ty.json"))

    #Rimozione e pulizia delle informazioni delle issues trovate
    elenco_issues,title_issues, body_issues= funzio-
ni.pulizia_info_issues(lista_info, type)

    # Pulizia del testo delle issues in modo da ottenere dei testi compren-
sibili
    body_issues = funzioni.pulizia_issues(body_issues)

    #Salvataggio dei file Json delle informazioni ottenute e pulite
    funzioni.salva_repositories_two_tag(elenco_issues, "issues_labels_" +
type, "unity", tag)
    funzioni.salva_repositories_two_tag(title_issues, "issues_titolo_" +
type, "unity", tag)
    funzioni.salva_repositories_two_tag(body_issues, "issues_testo_" + ty-
pe, "unity", tag)

    print(" ")
```

Figura 152 - Script ricerca delle issues (pt.2)

All'atto dell'esecuzione dello script nella console compaiono i seguenti risultati presentati nell'immagine sottostante (figura 153 e 154):

```
La ricerca delle informazioni sulle issues dei repositories è stata effettuata con successo!
Sono state trovate 516 issues aperte.
Sono state trovate 1821 issues chiuse.
 Il file : 'informazioni_issues_open_unity_augmented_reality.json, é stato salvato correttamente!
 Il file : 'informazioni_issues_close_unity_augmented_reality.json, é stato salvato correttamente!
Le issues aperte che presentano delle etichette sono 243 su 516 con una percentuale del 47 % sul totale.
Le issues open che presentano delle titoli sono 516 su 516 con una percentuale del 100 % sul totale.
Le issues open che presentano dei body sono 443 su 516 con una percentuale del 86 % sul totale.
Dopo la pulizia delle issue aperte, sono rimaste 436 issues su 443 iniziali con una riduzione del 2%.
 Il file : 'issues_labels_open_unity_augmented_reality.json, é stato salvato correttamente!
 Il file : 'issues_titolo_open_unity_augmented_reality.json, é stato salvato correttamente!
 Il file : 'issues_testo_open_unity_augmented_reality.json, é stato salvato correttamente!
```

Figura 153 - Output esecuzione ricerca delle issues (pt.1)

```
Le issues aperte che presentano delle etichette sono 715 su 1821 con una percentuale del 39 % sul totale.
Le issues close che presentano delle titoli sono 1821 su 1821 con una percentuale del 100 % sul totale.
Le issues close che presentano dei body sono 1201 su 1821 con una percentuale del 66 % sul totale.
Dopo la pulizia delle issue aperte, sono rimaste 1180 issues su 1201 iniziali con una riduzione del 2%.
 Il file : 'issues_labels_close_unity_augmented_reality.json, é stato salvato correttamente!
 Il file : 'issues_titolo_close_unity_augmented_reality.json, é stato salvato correttamente!
 Il file : 'issues_testo_close_unity_augmented_reality.json, é stato salvato correttamente!


Process finished with exit code 0
```

Figura 154 - Output esecuzione ricerca delle issues (pt.2)

In queste figure è possibile notare innanzitutto delle informazioni numeriche che mostrano la massiccia presenza di issues trovate all'interno dei repositories, per un totale di 2337 Issues divise in 516 aperte e 1821 chiuse.

Altri numeri interessanti riguardano il numero di issues che presentano le etichette che raggiungono una percentuale di circa il 41%, ed il testo nelle issues è presente in 1644 casi su 2337, ovvero nel 70% dei casi.

Sono presenti, inoltre, dei messaggi che comunicano la corretta creazione dei file Json ed anche il loro corretto salvataggio.

Nella seguente figura è presente un estratto del file Json:


```
[
    {
        "body": "",
        "comments": 1,
        "labels": [],
        "repos": "andrewnakas/OpenBrushVR",
        "title": "Create LICENSE"
    },
    {
        "body": "Hi!\r\n\r\nTrying to get a straight build from Unity with the scene ARKIT_IOSpORTRAIT.
        \r\nUnfortunately compiling to device I am getting the following errors from xCode...\r\n\r\n
        (Unity 2017.2.0f3 || Xcode v9.2 )\r\n\r\n... any idea? :(\r\n\r\nThank you by advance!\r\n\r\n<img
        width=\"863\" alt=\"screen shot 2017-12-22 at 17 10 57\" src=\
        "https://user-images.githubusercontent.com/6184992/34290513-1ef2b9e2-e73b-11e7-8e32-b1503988c3d0.png\"
        >\r\n",
        "comments": 1,
        "labels": [],
        "repos": "andrewnakas/OpenBrushVR",
        "title": "iOS sample drop Xcode errors on build?..."
    },
    {
        "body": "[This Gamasutra article](http://www.gamasutra.com/blogs/TimPettersen/20161206/286981/The_
        complete_guide_to_Unity__Git.php)outlines how to set up the Unity project to be more Git friendly.
        You'll probably want to make this transition in a unique commit with no other changes. I'd do it
        myself but I didn't want to stomp on any changes you'd potentially have made.",
        "comments": 6,
        "labels": [],
        "repos": "andrewnakas/OpenBrushVR",
        "title": "Unity project is not Git friendly"
    },
```


Figura 155 - File Json contenete le informazioni delle issues (pt.1)

```
    {
        "body": "I'm trying to add another platform and the best resource I have at the moment is the git
        commit history. It would be very helpful to have a high level overview of files to modify to add
        controller functionality.",
        "comments": 6,
        "labels": [],
        "repos": "andrewnakas/OpenBrushVR",
        "title": "New Platform Readme"
    },
    {
        "body": "I'm interested in using Vimeo in some upcoming projects for 360\u00b0 video streaming with
        Unity, so I like to know if this SDK is still maintained.",
        "comments": 5,
        "labels": [],
        "repos": "vimeo/vimeo-unity-sdk",
        "title": "Is the Unity SDK still maintained? "
    },
    {
        "body": "### Describe the issue\r\n\r\nWhen trying to reproduce a live video from vimeo having a
        premium account we get the following error:\r\n[AVProVideo] Error: Loading failed. File not found,
        codec not supported, video resolution too high or insufficient system resources.\r\nUnityEngine.
        Debug:LogError(Object)\r\n\r\nAnyone using live videos with this SDK?",
        "comments": 3,
        "labels": [],
        "repos": "vimeo/vimeo-unity-sdk",
        "title": "Live not working with AvPRO"
    },
    {
        "body": "### Describe the issue\r\nI have created a mobile application where I use vimeo platform
        in order to live stream my content from my studio. I am using the unity plugin as my development is
        with unity. Problem that I face is that I can't find a way through the api to be able to sync the
        video with the current stream time in case that video stays behind, which is very often the case.
        \r\n\r\n### Steps to reproduce the problem\r\n### Unity version\r\n\r\n### Operating system\r\nIos
        and Android\r\n\r\nThanks!",
        "comments": 0,
        "labels": [],
        "repos": "vimeo/vimeo-unity-sdk",
        "title": "Not able to sync video with current stream time"
    },
```


Figura 156 - File Json contenete le informazioni delle issues (pt.2)

Nel file json, mostrato nella figura 155 e 156, sono presenti le issues trovate nei repositories, dove in ognuno di essi sono presenti degli attributi, che sono:

- body, testo delle issues,
- comments, numero dei commenti in risposta alla issue,
- labels, tipologia dell'etichetta descrittiva della issue,
- repos, nome del progetto di riferimento,
- title, titolo della issue.

12.2.2 Analisi con Matlab

Per poter procedere con l'analisi statistica dei numeri delle issues presenti nei repositories di realtà aumentata, anche in questo caso si è fatto uso di Matlab per la creazione di uno script.

Lo script in questione fa uso della stessa funzione, usata per le Pulls Requests, per la creazione del Word Cloud, adattato semplicemente ad i dati che si hanno a disposizione per le Issues.

In esso è stato automatizzata anche la creazione dei grafici per l'analisi statistico delle labels, oltre alla creazione stessa delle Word Cloud per le issues sia aperte che chiuse.

Tutto questo che è stato appena descritto fa parte dello script mostrato nelle seguenti immagini (figura 157 E158):

```
clc; close all;clear all;

key=["open","close"];
word=["labels","titolo","testo"];

for i=1:2,
    for j=1:3,

 file(j)='issues_'+word(j)+'_'+key(i)+'_unity_augmented_reality.json';

        % Lettura dei files contenenti le varie info dei issues
        str = char(fread(fopen(file(j)),inf)');
        fclose(fopen(file(j)));
        val = string(sort(jsondecode(str)));

        if strcmp(word(j),"labels")==1
            vettore=conta_occorrenze(length(val),val);
            dim=max(double(vettore(:,2)))*5/100;
            count=0;
            for k=1:length(vettore)
                if double(vettore(k,2))>dim
                    count=count+1;
                    vettore_final(count,1)=vettore(k,1);
                    vettore_final(count,2)=vettore(k,2);
                end
            end
```

Figura 157 - Script analisi issues (pt.1)

```
            grafico(vettore_final,"AR","Labels "+key(i)+"
 Issues","Label","Occorrenze");
        elseif strcmp(word(j),"titolo")==1
            figure;
            wordcloud_ar(string(val),'Word Cloud Titoli Issues');
            saveas(gcf,'Word_Cloud_Titoli_Issues_'+key(i)+'.png');
        else
            figure;
            wordcloud_ar(string(val),'Word Cloud Testo Issues
 ('+key(i)+')');
            saveas(gcf,'Word_Cloud_Testo_Issues_'+key(i)+'.png');
        end
    end
end
```

Figura 158 - Script analisi issues (pt.2)

Una volta eseguito tale script vengono creati dei grafici e delle immagini che mostrano i Word Cloud costruiti per l'analisi delle parole presenti all'interno delle issues aperte e chiuse trovate dentro i repositories.

Nelle immagini che seguono, figure 159 e 160, sono raffigurati i due grafici a barre che indicano le occorrenze della presenza delle etichette usate per le issues.

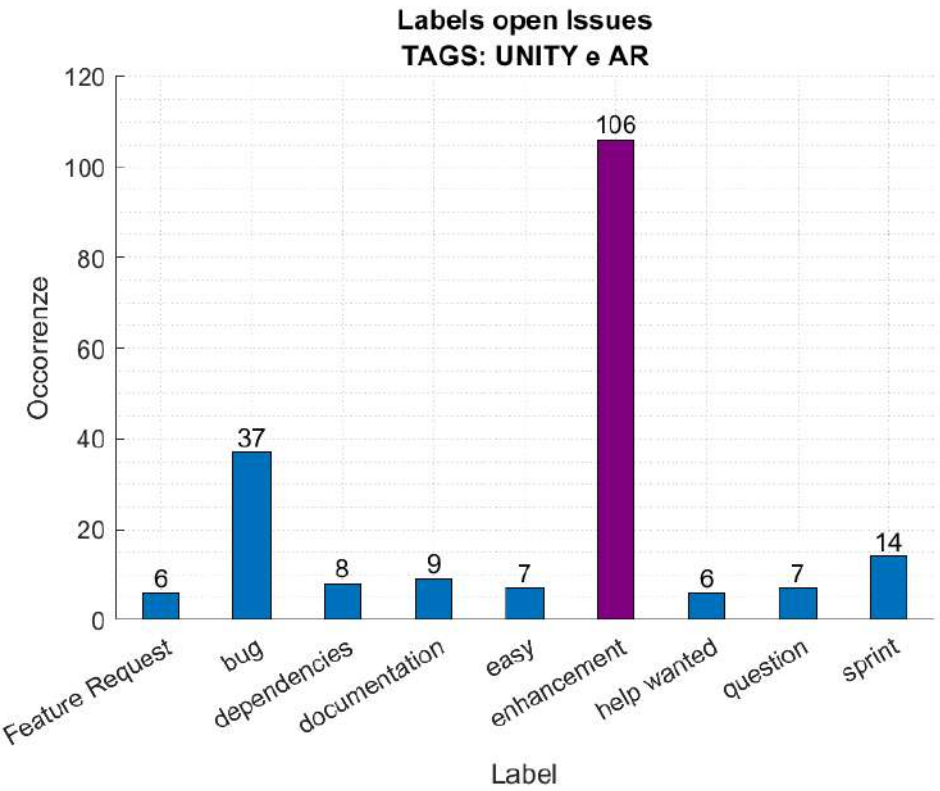

Figura 159 - Labels open issues

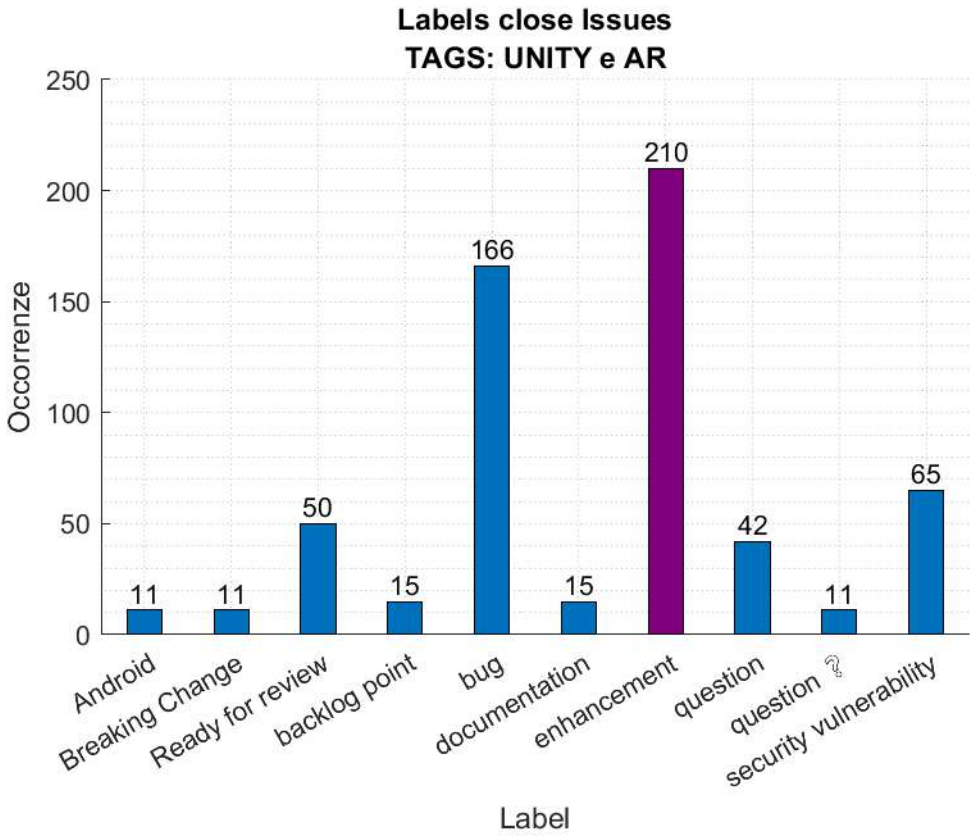

Figura 160 - Labels close issues

Come è possibile notare le etichette più presenti sono anche in questo caso "enhancement" e "bug", le quali sono davvero molto numerose rispetto alle altre tipologie.

Numeri alla mano è possibile notare che gli "ehnancement" sono 326, mentre i "bug" 204 che entrambe hanno un totale complessivo di 530 occorrenze su 958 etichette presenti nelle 2337 issues trovate.

Questo comporta il fatto che le etichette più frequentemente utilizzate, rappresentano il 55% delle etichette totali, ed il quasi 22,7% delle discussioni delle issues.

Seguono a questa statistica le etichette "security vulnerability", dove le 65 occorrenze rappresentano il 2,8% delle issues totali, "question" con il 2,5 % e "ready for review" con il 2,1%, dove queste ultime due presentano un'occorrenza rispettivamente di 59 e 50 casi.

RIEPILOGO ETICHETTE ISSUES					
ETICHETTA	***# OPEN***	***# CLOSE***	***# TOT***	***% su TAG***	***% su TOT***
ENHANCEMENT	*106*	*210*	*316*	*33*	*13,5*
BUG	*37*	*166*	*203*	*21,2*	*8,7*
SECURITY VULNERABILITY	*0*	*65*	*65*	*6,8*	*2,8*
QUESTION	*7*	*53*	*60*	*6,3*	*2,6*
READY FOR REVIEW	1	50	51	5,3	2,2
DOCUMANTATION	9	15	24	2,5	1
BACKLOG POINT	2	15	17	1,8	0,7
BREAKING CHANGE	2	11	13	1,4	0,6
SPRINT	14	0	14	1,5	0,6
ANDROID	1	11	12	1,3	0,5
DEPENDENCIES	8	2	10	1	0,4
EASY	7	2	9	0,9	0,4
HELP WANTED	6	4	10	1	0,4
FEATURE REQUESTS	6	1	7	0,7	0,3
ALTRO	**37**	**110**	**147**	**15,3**	**6,3**
NO TAG	**273**	**1106**	**1379**	**143,9**	**59**
TOTALE	***516***	***1821***	***2337***	***243,9***	***100***

Tabella 40 - Riepilogo Etichette Issues

Nella tabella sovrastante vengono riepilogate tutte le etichette in modo totale, trovate sia all'interno delle issues aperte che chiuse.

Successivamente, nelle immagini qui di seguito (dalla figura 161 alla figura 164) vengono mostrati i quattro Word Cloud che rappresentano ogni combinazione di analisi delle parole nelle issues, ovvero sia per i titoli che per i testi, così come per quelle aperte che quelle chiuse.

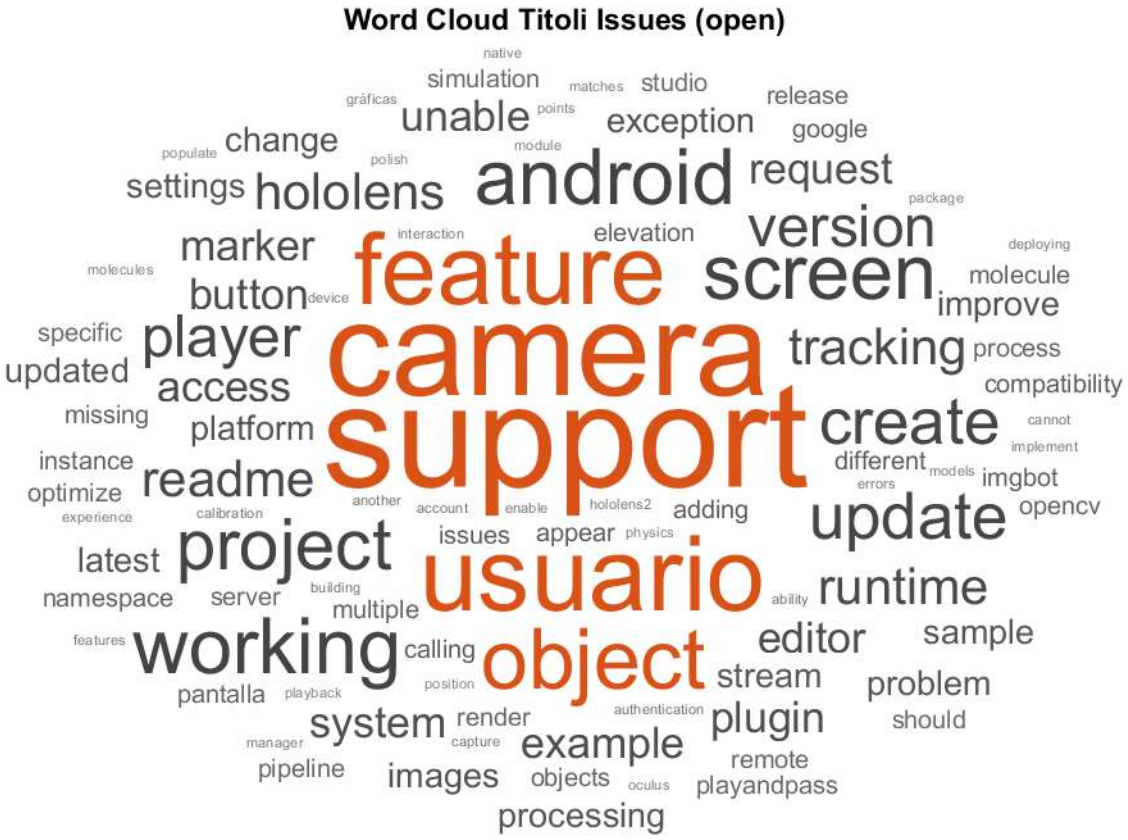

Figura 161 - Word Cloud Titoli Issues Open

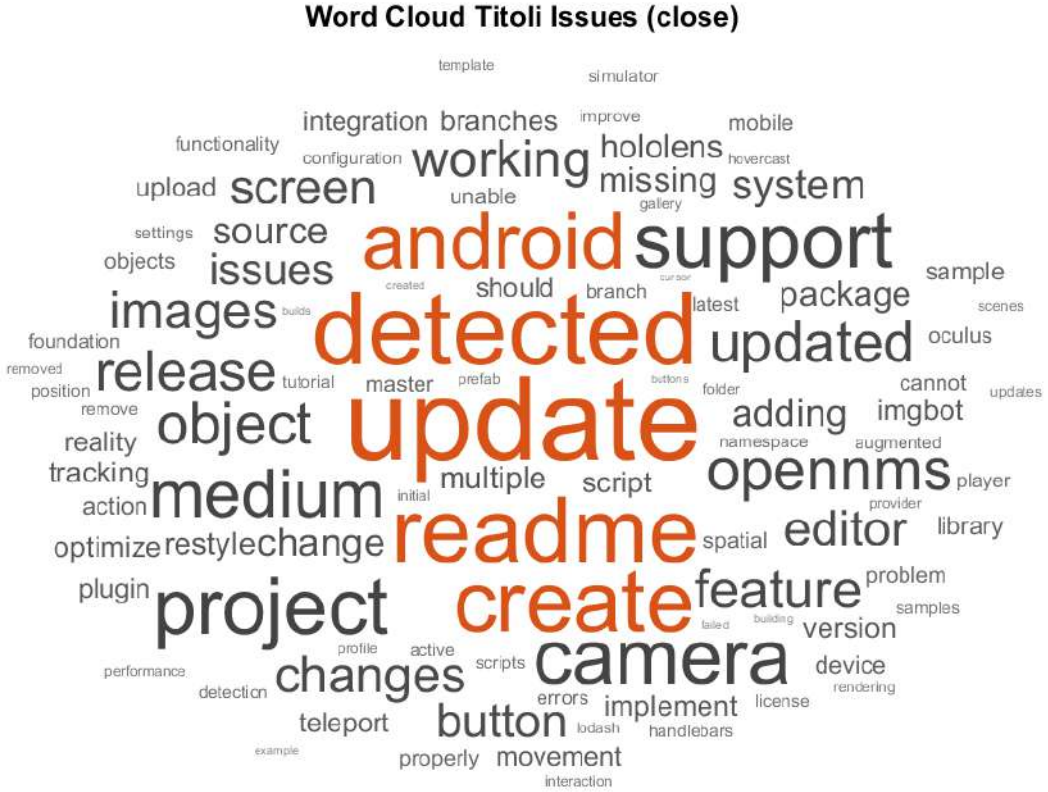

Figura 162 - Word Cloud Titoli Issues Close

Word Cloud Testo Issues (open)

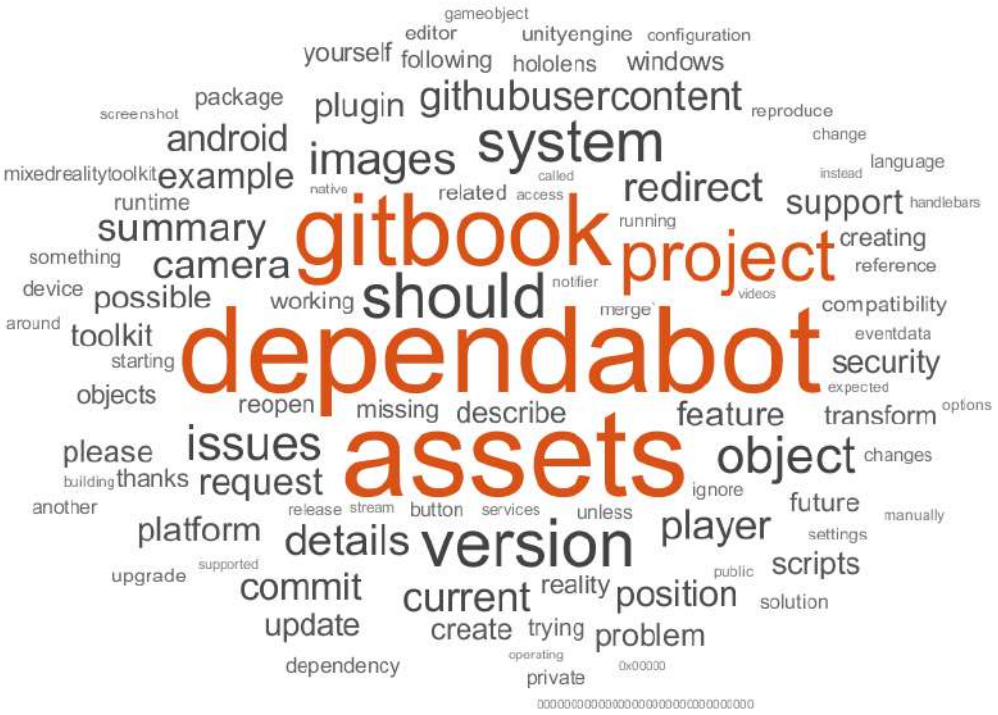

Figura 163 - Word Cloud Testo Issues Open

Word Cloud Testo Issues (close)

Figura 164 - Word Cloud Testo Issues Close

Anche in questo caso, come avvenuto nel paragrafo precedente per le Pulls, sono state riportate delle tabelle (tabella 41 e 42) che riassumono le parole più frequentemente presenti all'interno dei titoli e dei testi delle issues.

TOP 10 PAROLE IUSSUES (TITOLO)			
OPEN	***#***	***CLOSE***	***#***
camera	***28***	update	126
support	27	detected	68
object	21	create	64
feature	19	readme	59
update	17	project	54
hololens	***12***	***camera***	***53***
marker	***12***	support	51
android	***11***	***android***	***48***
button	***8***	working	28
tracking	***7***	***hololens***	***17***

Tabella 41 - Top 10 parole Issues Titolo

TOP 10 PAROLE IUSSUES (TESTO)			
OPEN	***#***	***CLOSE***	***#***
assets	262	details	897
dependabot	254	***framework***	***802***
gitbook	182	assets	696
object	155	***reality***	***685***
project	148	***library***	***636***
version	143	summary	628
system	129	***augmented***	***443***
camera	***103***	preview	440
should	95	bootstrap	409
images	***61***	***android***	***233***

Tabella 42 - Top 10 parole Issues Testo

Come è possibile notare anche che le parole in grassetto rappresentano quei termini che potenzialmente indicano dei temi che fanno parte della realtà aumentata.

Capitolo 13: Categorizzazione

In questo paragrafo si è cercato di effettuare una categorizzazione dei testi sia delle pull requests che delle issues, in modo da cercare di capire in che modo, queste vengono utilizzate per la discussione di problematiche inerenti a questa tecnologia, più nel dettaglio cercare di fare una categorizzazione di quelle che trattano la realtà aumentata.

13.1 Traduzione

Come prima cosa sono stati presi i testi precedentemente ottenuti, e tradotti in modo da poter condurre una analisi quanto più automatizzata e generalizzata possibile, e per questo si è scelta come unica lingua dei testi l'inglese.

Per fare ciò è stata sviluppata una funzione che prende in input i testi e tramite le Api di Google Traduttore[39] provvede a tradurli.

Nella seguente tabella vengono mostrate le informazioni inerenti alla funzione che serve per poter effettuare la traduzione dei testi.

[39] Google Traduttore è un servizio di traduzione automatica multilingue sviluppato da Google LLC. Offre un'interfaccia web, app mobili per Android e iOS e un'API che aiuta gli sviluppatori a creare estensioni del browser e applicazioni software. Google Traduttore supporta oltre 100 lingue a vari livelli e, a partire da maggio 2017, serve oltre 500 milioni di persone al giorno.

	ELEMENTO	DESCRIZIONE
NOME	TRADUZIONE FILE	Questa funzione permette di tradurre in inglese i file di testo, sfruttando le Api di Github.
TIPOLOGIA	FUNZIONE	
INPUT	FILE	Elenco dei testi che devono essere tradotti dall'esecuzione della funzione
	TIPO	Testo che viene utilizzato all'interno della funzione per la distinzione delle funzioni print di stampa a video nella console.
OUTPUT	TRADUZIONE	Lista dei testi tradotti
ESEMPIO D'USO	funzioni.traduzione_file(file,tipo)	

Tabella 43 - Informazioni funzione "traduzione_file"

Il codice che effettua la traduzione tramite le Api di Google è il seguente:

```
def traduzione_file(file, tipo):
    print("Ci sono " + str(len(file)) +" "+ tipo+ " da tradurre")
    print("Inizio traduzione "+tipo+"...")
    traduzioni = []
    count = 0
    count2 = 0
    i = 0
    start = time.time()
    for testo in file:
        l25 = len(file) / 100 * 25
        count = count + 1
        count2 = count2 + 1
        if count2!=361:
            if count >= l25:
                count = 0
                i = i + 1
                end = time.time()
                print("Sono state tradotte il " + str(i * 25) + "% delle
"+tipo+" in " + str(
                    round((end - start))) + " sec.")
            result = translator.translate(testo, dest='en', src='auto')
            traduzioni.append(result.text)
    end = time.time()
    print("Sono state tradotte il 100% delle "+tipo+" in " + str(round(end
- start)) + " sec.")
    return traduzioni
```

Figura 165 - Funzione "traduzione_file"

La funzione precedentemente mostrata viene utilizzata all'interno di due script uno per le pull requests ed una per le issues, i quali provvedono a tra-

durre i testi in inglese che verranno poi utilizzati per effettuare la categorizzazione.

Nella seguente tabella vengono mostrate le informazioni inerenti allo script per la traduzione delle pull requests.

	ELEMENTO	DESCRIZIONE
NOME	TRADUZIONE PULL REQUESTS	Questo script consente di ottenere i testi delle pull requests aperte e chiuse, tradotte in inglese.
TIPOLOGIA	SCRIPT	
INPUT DATI	INFO	Elenco dei testi delle pull requests sia aperte che chiuse
FUNZIONI USATE	TRADUZIONE FILE	Funzione per la traduzione dei testi.
	SALVA REPOSITORIES TWO TAGS	Funzione di salvataggio delle informazioni ricercate, memorizzate in un apposito file json.
OUTPUT	INFORMAZIONI	Elenco di informazioni visibili tramite console inerenti ai repositories ricercati.
	FILE JSON	File Json creati contenenti le varie informazioni tradotte.
ESEMPIO D'USO	traduzione_pull_requests.py (da eseguire da riga di comando)	

Tabella 44 - Informazioni script "traduzione_pull_requests"

Il codice dello script che effettua la traduzione delle pulls:

```
import json
import funzioni

# Questo script serve per tradutte i file che contengono i testi delle
pulls requests
# sia aperte che chiuse

pulls_open=json.load(open('pulls_testo_open_unity_augmented_reality.json'))
pulls_close=json.load(open('pulls_testo_close_unity_augmented_reality.json'
))

traduzioni_open=funzioni.traduzione_file(pulls_open,"pulls open")
funzioni.salva_repositories_two_tag(traduzioni_open, "testo_inglese",
"pulls", "open")

print("")
traduzioni_close=funzioni.traduzione_file(pulls_close,"pulls close")
funzioni.salva_repositories_two_tag(traduzioni_close, "testo_inglese",
"pulls", "close")
```

Figura 166 - Script "traduzione pull requests"

Una volta eseguito lo script precedentemente mostrato è possibile vedere i messaggi che avvisano l'evoluzione della traduzione dei testi, ed il completamento della traduzione con il salvataggio di questi su un file Json.

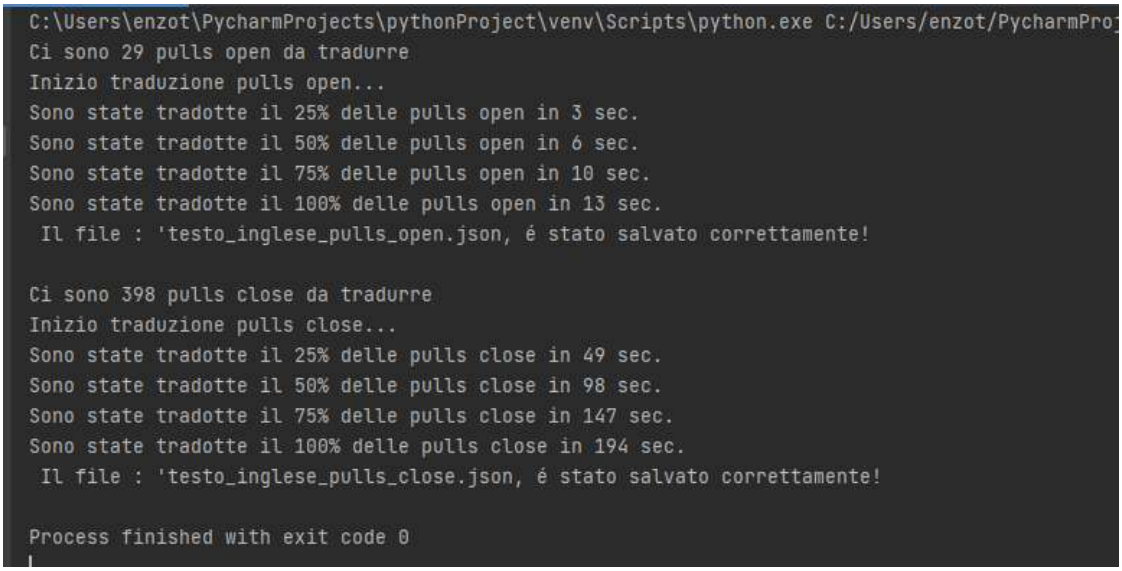

Figura 167 - Esecuzione "traduzione pull requests"

Nella seguente tabelle vengono mostrate le informazioni inerenti alla versione dello script utilizzato per la traduzione dei testi delle issues.

	ELEMENTO	***DESCRIZIONE***
NOME	TRADUZIONE ISSUES	Questo script consente di ottenere i testi delle issues aperte e chiuse, tradotte in inglese.
TIPOLOGIA	SCRIPT	
INPUT DATI	INFO	Elenco dei testi delle issues sia aperte che chiuse
FUNZIONI USATE	TRADUZIONE FILE	Funzione per la traduzione dei testi.
	SALVA REPOSITORIES TWO TAGS	Funzione di salvataggio delle informazioni ricercate, memorizzate in un apposito file json.
OUTPUT	INFORMAZIONI	Elenco di informazioni visibili tramite console inerenti ai repositories ricercati.
	FILE JSON	File Json creati contenenti le varie informazioni tradotte.
ESEMPIO D'USO	traduzione_issues.py (da eseguire da riga di comando)	

Tabella 45 - Informazioni script "traduzione_issues"

Il codice che effettua la traduzione dei testi delle issues è il seguente:

```
import json
import funzioni

# Questo script serve per tradutte i file che contengono i testi delle is-
sues sia aperte
# che chiuse

is-
sues_open=json.load(open('issues_testo_open_unity_augmented_reality.json'))
is-
sues_close=json.load(open('issues_testo_close_unity_augmented_reality.json'
))

traduzioni_open=funzioni.traduzione_file(issues_open,"issues open")
funzioni.salva_repositories_two_tag(traduzioni_open, "testo_inglese", "is-
sue", "open")

print("")
traduzioni_close=funzioni.traduzione_file(issues_close,"issues close")
funzioni.salva_repositories_two_tag(traduzioni_close, "testo_inglese", "is-
sue", "close")
```

Figura 168 - Script "traduzione issues"

Anche in questo caso all'atto dell'esecuzione dello script per la traduzione delle issues, vengono mostrati nella console l'evoluzione dell'esecuzione, ed infine il messaggio di avvenuto salvataggio dei testi tradotti.

```
C:\Users\enzot\PycharmProjects\pythonProject\venv\Scripts\python.exe C:\Users\enzot\PycharmProjects\pythonProject\t
Ci sono 436 issues open da tradurre
Inizio traduzione issues open...
Sono state tradotte il 25% delle issues open in 54 sec.
pausa di 60 secondi
Sono state tradotte il 50% delle issues open in 170 sec.
pausa di 60 secondi
Sono state tradotte il 75% delle issues open in 284 sec.
pausa di 60 secondi
Sono state tradotte il 100% delle issues open in 395 sec.
 Il file : 'testo_inglese_issue_open.json, é stato salvato correttamente!

Ci sono 1180 issues close da tradurre
Inizio traduzione issues close...
Sono state tradotte il 25% delle issues close in 145 sec.
pausa di 60 secondi
Sono state tradotte il 50% delle issues close in 347 sec.
pausa di 60 secondi
Sono state tradotte il 75% delle issues close in 546 sec.
pausa di 60 secondi
Sono state tradotte il 100% delle issues close in 750 sec.
pausa di 60 secondi
Sono state tradotte il 100% delle issues close in 810 sec.
 Il file : 'testo_inglese_issue_close.json, é stato salvato correttamente!

Process finished with exit code 0
```

Figura 169 - Esecuzione "traduzione issues"

13.2 Raggruppamento

A questo punto è possibile poter effettuare la categorizzazione dei testi, dove per fare ciò sono state effettuate delle operazioni particolari al testo in modo da ottenere il testo pulito e facilmente analizzabile.

La prima funzione che è stata sviluppata è quella che permette la rimozione all'interno del testo, di tutti gli elementi di punteggiatura ed i numeri, in modo da avere soltanto le parole all'interno del testo.

Le informazioni che sintetizzano questa funzione sono presenti nella seguente tabella:

	ELEMENTO	***DESCRIZIONE***
NOME	PAROLE SPLIT	Questa funzione permette di splittare il testo e rimuovere la punteggiatura nel testo.
TIPOLOGIA	FUNZIONE	
INPUT	TEXT	Elenco dei testi che devono essere splittati e filtrati.
OUTPUT	LISTA	Lista dei testi splittati e tradotti.
ESEMPIO D'USO	funzioni.parole_split(text)	

Tabella 46 - Informazioni funzione "parole_split"

Il codice che effettua questa "pulizia" ed il conseguente split delle parole è il seguente:

```
def parole_split(text):
    text=text.translate(str.maketrans(' ', ' ', string.punctuation))
    lista=[]

    number = ['1', '2', '3', '4', '5', '6', '7', '8', '9', '0']
    for n in number:
        text = re.sub(n, '', text)

    words = text.split(" ")
    for word in words:
        if len(word)<15 and len(word)>1:
            lista.append(word.lower())

    return lista
```

Figura 170 - Funzione "parole_split"

Altra funzione che è stata progettata è la funzione che permette di rimuovere le "stop words" e di individuare allo stesso momento le parole chiave che servono per la nostra analisi del testo automatizzata.

Per stop words di intende un elenco di parole ad esempio composte da poche lettere, oppure delle congiunzioni che grossomodo non tolgono il "significato" al testo, ma permettono un più rapido riscontro delle parole chiave.

Le informazioni inerenti questa funzione sono sintetizzate nella seguente tabella:

	ELEMENTO	DESCRIZIONE
NOME	RIMOZIONE STOP WORD	Questa funzione permette di rimuovere all'interno del testo stop words e di trovare all'interno di esso le parole chiave per la categorizzazione.
TIPOLOGIA	FUNZIONE	
INPUT	LISTA	Elenco dei testi che devono essere splittati e filtrati.
OUTPUT	LISTA FINALE	Lista dei testi in cui sono state rimosse le stop words.
	TOT AR	Lista dei testi in cui sono state trovate parole chiave inerenti alla realtà aumentata.
	TOT BUG	Lista dei testi in cui sono state trovate parole chiave inerenti alle problematiche ed agli errori.
	ELENCO ETICHETTE	Lista delle parole chiave trovate all'interno dei testi.
	CATEGORIE	Lista delle categorie trovate all'interno dei testi.
ESEMPIO D'USO	funzioni.rimozione_stop_word(lista)	

Tabella 47 - Informazioni funzione "rimozione stop word"

Il codice della funzione che effettua queste operazioni è il seguente, e viene mostrato nelle seguenti immagini (figure da 171 a 173).

```
def rimozione_stop_word(lista):
    stop_words = ['Hi', 'I','ourselves', 'hers', 'between', 'yourself',
'but', 'again', 'there', 'about', 'once', 'during',
                  'out', 'very', 'having', 'with', 'they', 'own', 'an',
'be', 'some', 'for', 'do', 'its', 'yours',
                  'such', 'into', 'of', 'most', 'itself', 'other', 'off',
'is', 's', 'am', 'or', 'who', 'as', 'from',
                  'him', 'each', 'the', 'themselves', 'until', 'below',
'are', 'we', 'these', 'your', 'his', 'through',
```

```
                    'don', 'nor', 'me', 'were', 'her', 'more', 'himself',
'this', 'down', 'should', 'our', 'their',
                    'while', 'above', 'both', 'up', 'to', 'ours', 'had',
'she', 'all', 'no', 'when', 'at', 'any',
                    'before', 'them', 'same', 'and', 'been', 'have', 'in',
'will', 'on', 'does', 'yourselves', 'then',
                    'that', 'because', 'what', 'over', 'why', 'so', 'can',
'did', 'not', 'now', 'under', 'he', 'you',
                    'herself','has', 'just', 'where', 'too', 'only', 'my-
self', 'which', 'those', 'i', 'after', 'few',
                    'whom', 't','being', 'if', 'theirs', 'my', 'against',
'a', 'by', 'doing', 'it', 'how', 'further',
                    'was', 'here', 'than','','by','at']
    key_word = ['ar', 'augmented', 'reali-
ty','vuforia','arcore','arkit','hololens', 'marker','tracking', 'texture',
                'image','camera','button','library','toolkit']
    word_bug =
['bug','error','security','broken','bust','bad','hurt','harm','wrong','malf
unction','dysfunction',
                    'disor-
der','trouble','critic','critical','crucial','fault','mistake','fallacy','f
ailure','lack',
                    'de-
fault','defect','lacuna','failing','absence','weak','tight','scant','defici
t','insufficient',
                    'dan-
ger','hazard','risk','peril','distress','trap','deception','pitfall','probl
ematic','cause','break',
                    'sma-
sh','crash','violate','breach','collapse','crumble','crack','decay','slump'
,'fall','flop','miss',
                    'abort','backfire','miscarry','fiasco']

    lista_finale=[]
    punteggio_ar=[]
    punteggio_bug=[]
    etichette=[]
```

Figura 171 - Funzione "rimozione_stop_words" (pt.1)

```
for word in lista:
    try:
        stop_words.index(word)
    except:
        lista_finale.append(word)
    try:
        parola=key_word.index(word)
        punteggio_ar.append(10)
        etichette.append(parola)
    except:
        punteggio_ar.append(0)
    try:
        word_bug.index(word)
        punteggio_bug.append(10)
    except:
        punteggio_bug.append(0)

tot_ar = sum(punteggio_ar)
tot_bug = sum(punteggio_bug)
elenco_etichette=[]
```

Figura 172 - Funzione "rimozione_stop_words" (pt.2)

```
for i in etichette:
    elenco_etichette.append(key_word[i])

categoria=[]
i=sorted(etichette)
if len(i)>0:
    i=i[0]
    if i>=0 and i<6:
        categoria.append("AR TOOL")
    if i==6:
        categoria.append("HOLOLENS")
    if i >=7 and i < 11:
        categoria.append("MARKER")
    if i>=11 and i<13:
        categoria.append("CAMERA")
    if i>=13:
        categoria.append("LIBRERIA")

return lista_finale,tot_ar,tot_bug,elenco_etichette,categoria
```

Figura 173 - Funzione "rimozione_stop_words" (pt.3)

Altra funzione utilizzata è stata quella che ha permesso di contare all'interno della lista dei testi quante occorrenze di parole chiave sono state trovate al suo interno.

Le informazioni riguardanti questa funzione sono presenti nella seguente tabella:

	ELEMENTO	***DESCRIZIONE***
NOME	CONTA PUNTEGGIO	Questa funzione permette di calcolare le occorrenze delle parole chiave trovate.
TIPOLOGIA	FUNZIONE	
INPUT	PUNTEGGIO	Elenco delle parole in cui devono essere calcolate le occorrenze.
OUTPUT	COUNT TOT	Conteggio dell'occorrenza calcolata.
ESEMPIO D'USO	funzioni.conta_punteggio(punteggio)	

Tabella 48 - Informazioni funzione "conta_punteggio"

Il codice che effettua tale operazione viene mostrato nella seguente immagine:

```
def conta_punteggio(punteggio):
    count_0=0
    count_10=0
    count_20=0
    count_30=0
    count_40=0
    count_50=0

    for x in punteggio:
        if x==0 :
            count_0=count_0+1
        elif x==10:
            count_10 =count_10+1
        elif x==20:
            count_20 = count_20 + 1
        elif x==30:
            count_30 = count_30 + 1
        elif x==40:
            count_40 = count_40 + 1
        elif x >= 50:
            count_50 = count_50 + 1

    count_tot=count_10+count_20+count_30+count_40+count_50
    return count_tot
```

Figura 174 - Funzione "conta_punteggio"

Per poter effettuare la categorizzazione del testo in modo automatizzato è stata implementata una funzione che a seconda delle parole chiavi che sono state trovate, associa una “etichetta” che permette di contraddistinguere in modo univoco quel determinato testo a cui si fa riferimento, in modo da carpirne il contenuto.

Le etichette, che sono state applicate per categorizzare i testi sia delle issues che delle pull requests, sono le seguenti:

- Ar Tool, la quale prevede di inserire in questa tipologia tutti testi che presentano al suo interno parole e termini riguardanti la realtà aumentata dal punto di vista dei tool come ad esempio Vuforia, Arcore e Arkit;

- Hololens, dove con questa etichetta si vuole raggruppare tutti quei testi che presentano tematiche che riguardano la categoria del visore stesso;
- Marker, questa etichetta raggruppa tutti i termini relativi alle immagini ed al tracking di esse;
- Camera, con questa etichetta invece si raggruppano tutti i termini relativi alle tematiche vicine al settore della Camera;
- Libreria, la seguente etichetta indica quelle tematiche inerenti le librerie dei tools utilizzati nell'applicazione di riferimento.

Per effettuare in modo ottimale la categorizzazione dei testi sono state fatte delle analisi incrociate, in modo da considerare soltanto quelle che presentano all'interno di esse parole che riguardano problematiche o errori inerenti all'applicazione di riferimento.

Le informazioni, che sintetizzano questa funzione, sono state trascritte nella seguente tabella:

	ELEMENTO	***DESCRIZIONE***
NOME	CATEGORIZZAZIONE	Questa funzione permette di calcolare la categorizzazione dei testi.
TIPOLOGIA	FUNZIONE	
INPUT	ELENCO CATEGORIE	Elenco delle parole in cui devono essere effettuata la categorizzazione.
	TITOLO	Testo che serve le print da console.
OUTPUT	INFO	Conteggio dell'occorrenza calcolata.
ESEMPIO D'USO	funzioni.categorizzazione(elenco_categorie, titolo)	

Tabella 49 - Informazioni funzione "categorizzazione"

La funzione che permette di categorizzare il testo è la seguente:

```
def categorizzazione(elenco_categorie, titolo):

    count_1=0
    count_2=0
    count_3=0
    count_4=0
    count_5=0

    for tipo in elenco_categorie:
        if tipo == "AR TOOL":
            count_1 = count_1 + 1
        if tipo == "HOLOLENS":
            count_2 = count_2 + 1
        if tipo == "MARKER":
            count_3 = count_3 + 1
        if tipo == "CAMERA":
            count_4 = count_4 + 1
        if tipo == "LIBRERIA":
            count_5 = count_5 + 1

    count_tot = count_1 + count_2 + count_3 + count_4 + count_5
```

Figura 175 - Funzione "categorizzazione" (pt.1)

```
print("Tipologie di problemi inerenti all'AR presenti nelle " + titolo + "
sono: ")
print(" 1- AR TOOL  con una percentuale del " + str(round(count_1 * 100 /
count_tot))+" %")
print(" 2- HOLOLENS con una percentuale del " + str(round(count_2 * 100 /
count_tot))+" %")
print(" 3- MARKER   con una percentuale del " + str(round(count_3 * 100 /
count_tot))+" %")
print(" 4- CAMERA   con una percentuale del " + str(round(count_4 * 100 /
count_tot))+" %")
print(" 5- LIBRERIA con una percentuale del " + str(round(count_5 * 100 /
count_tot))+" %")
print()
```

Figura 176 - Funzione "categorizzazione" (pt.2)

L'ultima funzione, ma non meno importante, che è stata sviluppata è quella di analisi che permette di gestire l'analisi testuale di ogni singolo testo, permettendo di effettuate la categorizzazione dello stesso, e di calcolare le statistiche dei testi che sono stati trovati e di stamparli a video nella console.

Questa funzione viene descritta nella seguente tabella:

	ELEMENTO	DESCRIZIONE
NOME	ANALISI	Questa funzione permette di analizzare i testi che verranno poi categorizzati
TIPOLOGIA	FUNZIONE	
INPUT	ELENCO INFO	Elenco delle parole in cui devono essere effettuata l'analisi e la categorizzazione.
	TIPO	Testo che caratterizza la tipologia di testo che si sta analizzando
	TESTO	Testo che serve le print da console.
OUTPUT	INFO	Conteggio dell'occorrenza calcolata.
ESEMPIO D'USO	funzioni.categorizzazione(elenco_categorie, titolo)	

Tabella 50 - Informazioni funzione "analisi"

Il codice che esegue queste operazioni viene mostrato nelle seguenti immagini (figure 177 e 179):

```
def analisi(elenco_info, tipo,testo):
    #applicazione delle funzioni per l'eliminazione delle stop word e della
punteggiatura al'interno del testo
    lista_issue=[]
    punteggio_ar=[]
    punteggio_bug=[]
    punteggio=[]
    info_ar_bug=[]
    elenco_etichette=[]
    elenco_categorie=[]

    for info in elenco_info:
        lista=parole_split(info)
        li-
sta_finale,tot_ar,tot_bug,etichette,categoria=rimozione_stop_word(lista)
        lista_issue.append(lista_finale)
        punteggio_ar.append(tot_ar)
        punteggio_bug.append(tot_bug)
        if tot_ar>0:
            if tot_bug>0:
                punteggio.append(tot_ar+tot_bug)
                info_ar_bug.append(info)
        if len(etichette)>0:
            elenco_etichette.append(etichette)
            elenco_categorie.append(categoria[0])

    count_ar=conta_punteggio(punteggio_ar)
    count_bug=conta_punteggio(punteggio_bug)
    count_ar_bug=conta_punteggio(punteggio)
```

Figura 177 - Funzione "analisi" (pt.1)

```
print("Nelle "+tipo+" analizzate sono state trovate:")
print("- "+str(len(elenco_info)-count_ar-count_bug)+" "+tipo+" senza rife-
rimenti all' AR, con una percentuale del "+str(
    round((len(elenco_info)-count_ar-count_bug)*100/(len(elenco_info))))+"
%")
print("- "+str(count_ar)+" "+tipo+" con riferimenti all'AR, con una percen-
tuale del "+str(round(count_ar*100/
      (len(elenco_info))))+" %")
print("- "+str(count_bug)+" "+tipo+" con riferimenti a degli errori, con
una percentuale del "+str(round(count_bug*100/
      (len(elenco_info))))+" %")
print("- "+ str(count_ar_bug) + " "+tipo+" con riferimenti ad errori di AR,
con una percentuale del " +
      str(round(count_ar_bug * 100 / (len(elenco_info)))) + " %")
print()

categorizzazione(elenco_categorie,testo)

return info_ar_bug
```

Figura 178 - Funzione "analisi" (pt.2)

Le funzioni appena descritte sono poi state utilizzate nei seguenti script, che eseguono la caratterizzazione del testo delle pull requests:

```
import json
import funzioni

# Analisi con categorizzazione delle pull requests inerenti alla realtà au-
mentata

pulls_open=json.load(open('testo_inglese_pulls_open.json'))
pulls_close=json.load(open('testo_inglese_pulls_close.json'))

print()
print("### PULLS OPEN ###")
pulls_ar_bug_open=funzioni.analisi(pulls_open,"pulls","pulls open")
funzio-
ni.salva_repositories_two_tag(pulls_ar_bug_open,"analisi","pulls","open")

print()
print("### PULLS CLOSE ###")
pulls_ar_bug_close=funzioni.analisi(pulls_close,"pulls","pulls close")
funzio-
ni.salva_repositories_two_tag(pulls_ar_bug_close,"analisi","pulls","close")
```

Figura 179 - Script "sentiment pull requests"

L'esecuzione dello script precedente mostra a video nella console le seguenti informazioni riguardanti il numero di pull requests inerenti la realtà aumentata che sono state trovate all'interno dei testi, con le relative percentuali di presenza.

Inoltre, nella parte inferiore nelle immagini (figura 181 e 182) è possibile vedere l'elenco delle categorizzazioni automatizzate che sono state fatte per i testi.

```
### PULLS OPEN ###
Nelle pulls analizzate sono state trovate:
- 11 pulls senza riferimenti all' AR, con una percentuale del 38 %
- 7 pulls con riferimenti all'AR, con una percentuale del 24 %
- 11 pulls con riferimenti a degli errori, con una percentuale del 38 %
- 1 pulls con riferimenti ad errori di AR, con una percentuale del 3 %

Tipologie di problemi inerenti all'AR presenti nelle pulls open sono:
 1- AR TOOL  con una percentuale del 29 %
 2- HOLOLENS con una percentuale del 0 %
 3- MARKER   con una percentuale del 71 %
 4- CAMERA   con una percentuale del 0 %
 5- LIBRERIA con una percentuale del 0 %

 Il file : 'analisi_pulls_open.json, é stato salvato correttamente!
```

Figura 180 - Esecuzione script "sentiment pull requests" (pt.1)

```
### PULLS CLOSE ###
Nelle pulls analizzate sono state trovate:
- 198 pulls senza riferimenti all' AR, con una percentuale del 50 %
- 152 pulls con riferimenti all'AR, con una percentuale del 38 %
- 47 pulls con riferimenti a degli errori, con una percentuale del 12 %
- 23 pulls con riferimenti ad errori di AR, con una percentuale del 6 %

Tipologie di problemi inerenti all'AR presenti nelle pulls close sono:
 1- AR TOOL  con una percentuale del 76 %
 2- HOLOLENS con una percentuale del 2 %
 3- MARKER   con una percentuale del 12 %
 4- CAMERA   con una percentuale del 7 %
 5- LIBRERIA con una percentuale del 4 %

 Il file : 'analisi_pulls_close.json, é stato salvato correttamente!

Process finished with exit code 0
```

Figura 181 - Esecuzione script "sentiment pull requests" (pt.2)

Come è possibile vedere nella tabella sottostante, sono state calcolate le percentuali totali per ogni categoria riscontrata nelle pull requests:

PULL REQUESTS	OPEN		CLOSE		TOTALE	
	%	#	%	#	%	#
AR TOOL	29	8	76	301	**72**	**309**
HOLOLENS	0	0	2	7	2	7
MARKER	71	21	12	47	16	68
CAMERA	0	0	7	27	6	27
LIBRERIA	0	0	4	15	4	15

Tabella 51 - Riepilogo categorie Pull Requests

Le funzioni appena descritte sono poi state utilizzate nei seguenti script, che eseguono la caratterizzazione del testo delle issues:

```
import json
import funzioni

# Analisi con categorizzazione delle issues inerenti alla realtà aumentata

issues_open=json.load(open('testo_inglese_issue_open.json'))
issues_close=json.load(open('testo_inglese_issue_close.json'))

print()
print("### ISSUES OPEN ###")
issues_ar_bug_open=funzioni.analisi(issues_open,"issues","issues open")
funzio-
ni.salva_repositories_two_tag(issues_ar_bug_open,"analisi","issues","open")

print()
print("### ISSUES CLOSE ###")
issues_ar_bug_close=funzioni.analisi(issues_close,"issues","issues close")
funzio-
ni.salva_repositories_two_tag(issues_ar_bug_close,"analisi","issues","close
")
```

Figura 182 - Script "sentiment_issues"

L'esecuzione dello script precedente mostra a video nella console le seguenti informazioni riguardanti il numero di issues che riportano tematiche inerenti alla realtà aumentata che sono state trovate all'interno dei relativi testi, con le percentuali di presenza.

```
### ISSUES OPEN ###
Nelle issues analizzate sono state trovate:
- 215 issues senza riferimenti all' AR, con una percentuale del 49 %
- 129 issues con riferimenti all'AR, con una percentuale del 30 %
- 91 issues con riferimenti a degli errori, con una percentuale del 21 %
- 33 issues con riferimenti ad errori di AR, con una percentuale del 8 %

Tipologie di problemi inerenti all'AR presenti nelle issues open sono:
 1- AR TOOL  con una percentuale del 45 %
 2- HOLOLENS con una percentuale del 10 %
 3- MARKER   con una percentuale del 22 %
 4- CAMERA   con una percentuale del 16 %
 5- LIBRERIA con una percentuale del 7 %

 Il file : 'analisi_issues_open.json, é stato salvato correttamente!
```

Figura 183 - Esecuzione script "Sentiment issues" (pt.1)

```
### ISSUES CLOSE ###
Nelle issues analizzate sono state trovate:
- 536 issues senza riferimenti all' AR, con una percentuale del 45 %
- 409 issues con riferimenti all'AR, con una percentuale del 35 %
- 234 issues con riferimenti a degli errori, con una percentuale del 20 %
- 112 issues con riferimenti ad errori di AR, con una percentuale del 9 %

Tipologie di problemi inerenti all'AR presenti nelle issues close sono:
 1- AR TOOL  con una percentuale del 45 %
 2- HOLOLENS con una percentuale del 7 %
 3- MARKER   con una percentuale del 11 %
 4- CAMERA   con una percentuale del 17 %
 5- LIBRERIA con una percentuale del 20 %

 Il file : 'analisi_issues_close.json, é stato salvato correttamente!

Process finished with exit code 0
```

Figura 184 - Esecuzione script "sentiment issues" (pt.2)

Inoltre, come fatto nel caso delle pull requests, nella parte inferiore nelle immagini (figura 184 e 185) è possibile vedere l'elenco delle categorizzazioni automatizzate che sono state fatte per i testi.

Come è possibile vedere nella tabella sottostante, sono state calcolate le percentuali totali per ogni categoria riscontrata nelle issues:

ISSUES	OPEN		CLOSE		TOTALE	
	%	#	%	#	%	#
AR TOOL	45	196	45	531	**45**	**727**
HOLOLENS	10	44	7	83	8	127
MARKER	22	96	11	130	14	226
CAMERA	16	70	17	200	16	270
LIBRERIA	7	30	20	236	17	266

Tabella 52 - Riepilogo categorie issues

13.3 Tipologie

Nel dettaglio è possibile evidenziare alcune argomentazioni più frequenti in merito di questa categoria, ovvero AR TOOL, oppure semplicemente trattate all'interno di questi testi:

- le difficoltà nell'apertura dei progetti in caso di disuniformità delle versioni di utilizzo dei progetti stessi.
- Errori nell'impacchettamento delle cartelle in cui sono situati parti del progetto.
- Problemi di visualizzazione della telecamera dall'apertura dell'applicazione tramite versione android o ios.
- Visualizzazione dello schermo tutto nero senza avvio diretto dell'applicazione.
- Problemi nella rilevazione del marker nei vari ambienti di sviluppo, nel momento in cui si vuole replicare un determinato progetto.
- Impossibilità di trovare oggetti per tipo e nome all'interno di percorsi prefissati nelle cartelle del progetto.
- Malfunzionamenti nell'utilizzo di plugin integrativi nell'utilizzo degli strumenti di sviluppo.

- Difficoltà di adattabilità del progetto esistente con aggiunta di ulteriori elementi.
- Malfunzionamento dell'utilizzo dell'sdk di arcore e di arkit.
- Collisione di oggetti all'interno della scena non previsti dal funzionamento dell'applicazione.
- Collisione degli oggetti con il mondo reale

Quindi come si è potuto evincere dal precedente elenco le argomentazioni/problemi principali, che sono stati trovati in questa categoria, rappresentano dei malfunzionamenti nell'utilizzo delle piattaforme per lo sviluppo delle applicazioni stesse, oppure nel settaggio delle varie piattaforme con conseguente adattamento alle varie configurazioni.

Sempre dal punto di vista analitico/statistico si evince che sui 309 testi per le pulls e 727 per issues, il quasi 3% per le pulls, mentre il 26% nel secondo caso delle issues, esse sono ancora aperte, il che implica che queste argomentazioni/segnalazioni non hanno ancora ricevuto una risposta/risoluzione che esse ponevano.
Le restanti categorie anche se meno presenti dal punto di vista percentuale, comunque nell'insieme riguardano una buona fetta delle argomentazioni che vengono trattate all'interno sia delle Issues che delle Pull Requests.
Anche in questo caso per ogni categoria che è stata utilizzata per l'analisi, verranno riportati gli elenchi che sintetizzano grossomodo le tematiche più affrontate, così come le problematiche più frequenti trovate dall'utenza.

Per quanto riguarda la categoria "HOLOLENS" in questi casi erano presenti tutti i testi che trattavano dubbi o miglioramenti da proporre per consentire una esperienza più performante sotto alcuni punti di vista.

Un breve elenco dei temi che sono stati trattati è il seguente:

- Integrazione dei tools con le simulazioni previste per il funzionamento con hololens.
- Miglioramenti dell'esperienza utente tramite hololens.
- Suggerimenti vari per il miglioramento della visualizzazione dei controller manuali.
- Incompatibilità dei pacchetti con il dispositivo.
- Errori nel collegamento dell'applicazione con il dispositivo e conseguente accesso alla piattaforma.
- Problemi di lentezza nell'emulazione della applicazione.
- Errata visualizzazione del marker.
- Errori nel funzionamento dell'applicazione dato il click del bottone virtuale per l'esecuzione di una determinata funzionalità prevista.
- Suggerimenti per l'implementazione dell'applicazione con l'utilizzo del visore.
-

Altre due categorie che sono state trattate sono quelle che riguardano le etichette di:

- Marker
- Camera

Nel primo caso si trattava per la maggior parte di problematiche e di errori nella visualizzazione di alcuni elementi dell'applicazione, quali principalmente i marker per la creazione di oggetti atti allo svolgimento delle funzionalità.

Mentre nel secondo caso consistevano in argomentazioni inerenti ai problemi di visualizzazione di condivisione o settaggio dei parametri delle fotocamere nei vari dispositivi che sono stati utilizzati per l'esecuzione oppure la replica di tali progetti.

Un riepilogo delle principali tematiche trattate, nei testi contrassegnati con l'etichetta Marker, sono:

- Problemi di acquisizione del marker con la fotocamera.
- Problemi di acquisizione della texture di una immagine personalizzata.
- Problemi di scaling delle figure e degli oggetti all'interno dell'ambiente di sviluppo.
- Mancata visualizzazione degli oggetti nel mondo reale.
- Problemi di occlusione degli oggetti.
- Interruzione dell'acquisizione del video.

Mentre per quanto riguarda la categoria Camera sono:

- Problemi di setting dei parametri della fotocamera.
- Problemi nella visualizzazione delle anteprime video e matriciali della fotocamera.
- Problemi di condivisione della camera all'interno delle applicazioni.
- Problemi di visualizzazione dello sfondo della fotocamera

In ultima analisi vi è la categoria denominata "Libreria", in cui sono stati inseriti al proprio interno tutte quelle Issues e Pull Requests che consistono nel capire come funzionano e come poter reperire determinate informazione

all'atto del download del progetto, e quindi comprenderne al meglio la configurazione.

Tra le principali richieste di informazioni vi sono:

- Malfunzionamento delle librerie dedicate per una determinata funzione dell'applicazione.
- Miglioramento del funzionamento degli editor su diverse piattaforme.
- Impossibilità di trovare una configurazione corrispondente del progetto.

Come si è potuto evincere da queste informazioni che sono state raccolte e riportate, la maggior parte si trattano quindi di:

- Richieste di miglioramenti estetici e progettuali.
- Segnalazione di errori e malfunzionamenti.
- Proposte di soluzioni ad alcuni malfunzionamenti.
- Chiarimenti di utilizzo dei software per l'utilizzo dei progetti.

Capitolo 14: Conclusioni

Come già accennato nei paragrafi precedenti, dal punto di vista dell'analisi dei testi che sono stati trovati nelle Issues e nelle Pull request, si è potuto evincere che la maggior parte di esse possono essere racchiuse all'interno delle 5 macrocategorie che sono state utilizzate per la circoscrizione delle informazioni.

Per poter effettuare al meglio una categorizzazione dei testi delle issues e delle pull request è stata fatta una lettura preventiva in modo da capire meglio come poter permettere di fare una analisi quanto più oggettiva e corretta possibile.

Questa lettura ha permesso di poter quindi sviluppare al meglio la creazione di questa funzione per la categorizzazione.

La categorizzazione, che si è voluta fare, prevede quindi, la suddivisione per macro-argomenti dei testi trovati nei repositories di github, dove ad ogni accorpamento è stato dato un nome, ovvero una etichetta univoca e generalista.

Queste etichette che sono state date sono 5 ed ognuna di essa ingloba quindi una serie di argomenti simili tra loro in cui è possibile carpirne una sorta di analogia tra di esse.

Come è possibile notare dai dati che sono stati trovati, la categoria più presente è quella che mostra le informazioni inerenti alla realtà aumentata, ed ai

tool annessi alla creazione di tali applicazioni, ovvero quella denominata con l'etichetta AR TOOL.

Questa ha raggiunto una percentuale di presenza del 72% per quanto riguarda i testi delle Pull requests, ed il 45% per le issues, il che ovviamente era facilmente prevedibile dato che trattandosi di applicazioni di realtà aumentata, la maggior parte delle discussioni parlano di curiosità o problematiche trovate con l'utilizzo dell'applicazioni dall'utenza.

Questi numeri non implicano necessariamente che le informazioni che sono state trovate inneggino a dei provvedimenti o delle segnalazioni solo per la risoluzione di problemi di queste applicazioni o dei tool con cui sono state progettate, ma anche molto spesso si tratta di domande poste dall'utenza per capire come replicare il progetto, apportarne modifiche oppure capire come adattare al meglio i software che vengono utilizzati per i vari progetti.

Come descritto anche in precedenza per fare la seguente categorizzazione sono state effettuate delle ricerche mirate a trovare dei termini precisi che ricordassero in qualche modo delle problematiche, o quanto meno dei malfunzionamenti o dei bug.

Nella maggior parte dei casi quindi queste discussioni sono servite per il miglioramento quindi di queste applicazioni sotto tutti i punti di vista sia progettuali che ai fini realizzativi degli stessi.

Queste procedure di condivisione dei repositories hanno aiutato senza ombra di dubbio gli sviluppatori alla soluzione dei problemi riscontrati in esse dagli utilizzatori, ma anche ad ampliarne le funzionalità stesse di queste applicazioni.

Indice delle figure

Indice delle tabelle

Bibliografia e Sitografia

[1] Treccani, Realtà Aumentata, Treccani (Ultima visita 20/01/2021)

[2] Tom Paquin, analista di ricerca, Aberdeen Market Società di intelligence

[3] Andrea Carobene, Realtà Virtuale, Treccani (Ultima visita 20/01/2021)

[4] Ivan Sutherland, Information Processing Techniques, “The ultimate display”, Proceedings of the IFIP Congress, Gennaio 1965

[5] Tobias Hollerer e Dieter Schmalstieg, “Augmented Reality: Principles and Practice, A Brief History of Augmented Reality”, Giugno 2016

[6] Thomas P. Caudell e David W. Mizell, “Augmented reality: An application of heads-up display technology to manual manufacturing processes”, System Sciences, 1992. Proceedings of the Twenty-Fifth Hawaii International Conference, Gennaio 1992

[7] Daniel Wagner e Dieter Schmalstieg, “First Steps Towards Handheld Augmented Reality”, Wearable Computers, 2003. Proceedings. Seventh IEEE International Symposium, Novembre 2005

[8] Ventana Research, “Augmented reality is changing marketing and sales”, Gennaio 2018

[9] quattroruote.it, https://www.quattroruote.it/news/nuove_tecnologie/2017/03/30/jeep_compass_visualiser_negli_showroom_con_la_realta_aumentata.html, (Ultima visita 25/01/2021)

[10] Google Glass, wikipedia.com (Ultima visita 26/01/2021)

[11] Anymotion.com, “What are augmented reality markers?” (Ultima visita 26/01/2021)

[12] Sneha Pawar, Sayali Thorat, Harshanjali Chandekar e Sneha Bhosale, “3D Based Augmented Reality for Structural Diagrams”, International Journal of Engineering Research & Technology (IJERT), Marzo 2014

[13] Bernd Brugge, Asa MacWilliams, Thomas Reicher, “Study on Software Architectures for Augmented Reality Systems”, Chair for Applied Software Engineering Institut f¨ur Informatik Technische Universit¨at Munchen, Ottobre 2002

[14] Blender, “https://www.blender.org/download/releases/2-82/” (Ultima visita 1/02/2021)

[15] Cinema 4D, “https://www.studiodaily.com/2018/09/maxon-publishes-cinema-4d-release-20/” (Ultima visita 1/02/2021)

[16] Lumion, “https://lumion.com/product.html” (Ultima visita 1/02/2021)

[17] Jim Scheibmeir, Yashwant K. Malaiya, “Quality Model for Testing Augmented Reality Applications”, 2019 IEEE 10th Annual Ubiquitous Computing, Electronics & Mobile Communication Conference (UEMCON), Ottobre 2019

[18] macitynet.it, “https://www.macitynet.it/salto-apple-nella-realta-aumenta-la-piattaforma-arkit/” (Ultima visita 4/02/2021)

[19] bitbar.com, “https://bitbar.com/blog/how-bitbar-helped-google-to-automate-their-augmented-reality-testing/ (Ultima visita 5/02/2021)

[20] altom.gitlab.io, “https://altom.gitlab.io/altunity/altunitytester/” (Ultima visita 10/02/2021)

[21] Progetto Github, “https://github.com/kliments/ProteinViz” (Ultima visita 10/03/2021)

[22] Progetto Github, “https://github.com/CallumHoughton18/ChemViewAR” (Ultima visita 14/03/2021)

[23] Progetto Github, “https://github.com/rwth-acis/GaMR” (Ultima visita 14/03/2021)

[24] Progetto Github, “https://github.com/robofit/arcor2_areditor” (Ultima visita 14/03/2021)

[25] Documentazione Unity 3D, ARCameraBackground https://docs.unity3d.com/Packages/com.unity.xr.arfoundation@4.1/api/UnityEngine.XR.ARFoundation.ARCameraBackground.html (Ultima visita 3/04/2021)

[26] Documentazione Unity 3D, XRCameraImage https://docs.unity3d.com/Packages/com.unity.xr.arsubsystems@2.1/api/UnityEngine.XR.ARSubsystems.XRCameraImage.html (Ultima visita 3/04/2021)

[27] Progetto Github, “https://github.com/Mihir0106/AR_BasketBall” (Ultima visita 3/04/2021)