UNIVERSITÀ DEGLI STUDI DI NAPOLI FEDERICO II
Scuola Politecnica e delle Scienze di Base
Corso di Laurea Magistrale in Ingegneria Informatica

Tesi di laurea magistrale

Strumenti e tecniche di automation testing per applicazioni di realtà aumentata

2019/2020

Relatore
Ch.mo Prof.ssa Anna Rita Fasolino

Correlatore
Ch.mo Prof. Porfirio Tramontana

Candidato
Sabato Danilo Bevilacqua
Matr. M63000836

Overview del lavoro di tesi

Contesto:

- applicazioni di realtà aumentata per dispositivi mobili Android

Problema affrontato:

- automazione del testing di sistema di tali applicazioni che, ancora oggi, è effettuato da tester umani

Contributi:

- Studio preliminare dei tools aventi le potenzialità per effettuare il testing di sistema di applicazioni di realtà aumentata;
- proposta di una metodologia di testing model based per tali applicazioni.

Introduzione alla realtà aumenta

Definizione e principali differenze rispetto alla realtà virtuale e alla realtà mista

- *Realtà aumentata (AR)*: tecnologia che combina, in uno schermo, oggetti virtuali con la scena reale che si sta osservando.
  - Combinazione di oggetti virtuali nel mondo reale;
  - Interazione real-time;
  - Gestione di oggetti 3D;

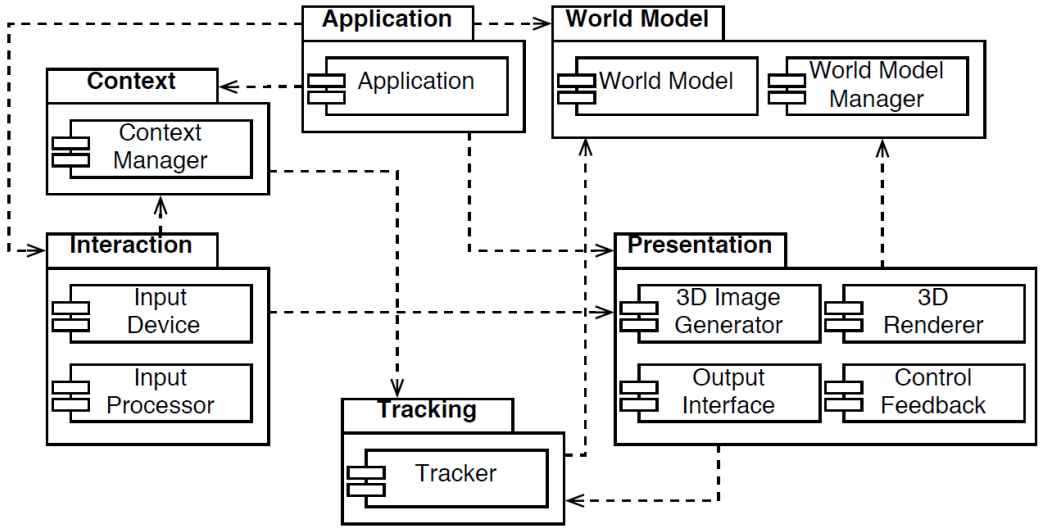
- *Realtà virtuale (VR)*: tecnologia che catapulta il suo utilizzatore in un mondo completamente virtuale escludendo quello reale.

- *Realtà Mista (MR)*: tecnologia che combina caratteristiche dell'AR e della VR.

Architettura generale di un sistema AR

Package Diagram dell'architettura di riferimento

- ***Application***: componente che ingloba tutto il codice necessario per una specifica tipologia di applicazione;
- ***Interaction***: recepisce ed elabora gli input provenienti dall'utente;
- ***Presentation***: responsabile dell'output all'utente, gestisce gli oggetti virtuali;
- **World Model**: raccoglie le informazioni legate al mondo reale in cui si muove l'utente;



- ***Tracking***: responsabile della registrazione della posizione di oggetti reali e virtuali. E‘ importante sottolineare che dovrà essere eseguito sempre e parallelamente ad altre attività del sistema;
- ***Context***: raccoglie i diversi tipi di dati di contesto tra cui le preferenze e le attività dell'utente.

Tools per la realizzazione di applicazioni AR

Panoramica degli SDK e degli Engine per la realizzazione di AR app

SDK

- ARKit
- ARCore
- wikitude
- ARTOOLKIT
- vuforia™

Engine

- unity
- UNREAL ENGINE

Tools per il testing di sistema di AR app

Confronto tra AltUnity Tester e AirTestProject

AltUnity Tester

- Scaricabile gratuitamente dallo Unity Asset Store;
- Linguaggi supportati: C#, Python e Java;
- Moduli di cui si compone: AltUnity Server, AltUnity Client, AltUnity Tester Editor Window.

Airtest Project

- Scaricabile gratuitamente dal sito ufficiale;
- Linguaggi supportati: Python;
- Moduli di cui si compone: AirTest, Poco-SDK, AirTestIDE, AirLab.

Entrambi i tools sono in grado di generare eventi GUI ed eventi sensore per l'effettuazione dei test. Il tester ha l'onere di implementare la test suite che supporterà l'automazione dei casi di test.

Per entrambi i tools è stata provata l'interazione con diverse categorie di oggetti Unity ed è emerso che AltUnity Tester presenta delle limitazioni maggiori rispetto a quelle di AirTest Project

Metodologia proposta

Di seguito sono mostrati gli step di cui si compone la metodologia di testing proposta:

- Modellazione dell'applicazione da testare;
- Utilizzo dei criteri di copertura, sul modello creato, per la generazione della relativa test suite;
- Implementazione ed esecuzione della test suite;
- Valutazione della test suite con metriche di copertura e qualità.

Modellazione e criteri di copertura

Model Based Testing (MBT) e State Based Testing (SBT)

Caratteristiche dell'MBT

- comportamento del SUT descritto dal modello;
- derivazione dei test a partire dal modello ottenuto (con possibilità dell'automazione nella codifica dei test);
- Facilitazione nella manutenzione dei test.

Lo SBT è una particolare categoria di MBT con le seguenti caratteristiche:

- Modello utilizzato: FSM (Finite State Machine)
  - Stato: rappresenta una particolare condizione del SUT che deve essere verificata;
  - Transizione: permette il passaggio da uno stato all'altro e rappresenta l'interazione effettuata dall'utente;
- Criteri di copertura della FSM ottenuta, in grado di fornire la test suite relativa:
  - All States Coverage: prevede la generazione di uno o più paths in grado di coprire tutti gli stati della FSM;
  - All Transitions Coverage: prevede la generazione di uno o più paths in grado di coprire tutte le transizioni della FSM;
  - All One Loop Paths Coverage: prevede la generazione di uno o più paths contenti loop, ossia path con la ripetizione (almeno e al più due volte) di uno stesso stato.

Valutazione test suite

Metriche di copertura e qualità

Il metodo delle sonde permette di verificare la copertura del codice relativa a una specifica test suite:

- Creazione dello Script «SondaManager» da inserire nel progetto Unity dell'applicazione per la creazione del file di log;
- Inserimento delle sonde negli script del progetto Unity (instrumentazione) secondo il seguente template:

  NomeScript.NomeFunzione.CostruttoECondizione.CostruttoInnestato
- Impiego del programma «Analisi Log» utile all'import dal dispositivo del file di log con la conseguente analisi. Tale analisi mostra la percentuale di copertura di ogni script e quali rami non sono stati coperti.

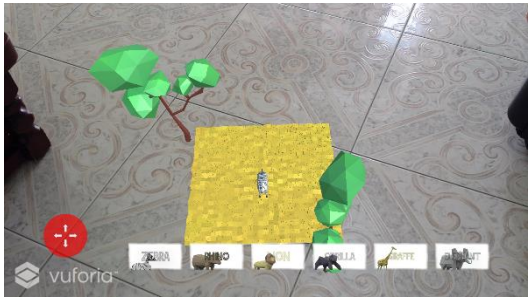
L'analisi mutazionale permette di valutare la qualità della test suite relativa a uno specifico criterio di copertura

- Generazione mutante mediante l'iniezione dell'errore all'interno del SUT (supposto corretto);
- Esecuzione della test suite sul mutante;
- Confronto del report della test suite, eseguita sul mutante, con l'oracolo;
- Calcolo del TER (Test Effectiveness Ratio) secondo la seguente formula:

$$TER = \frac{km}{tm}$$

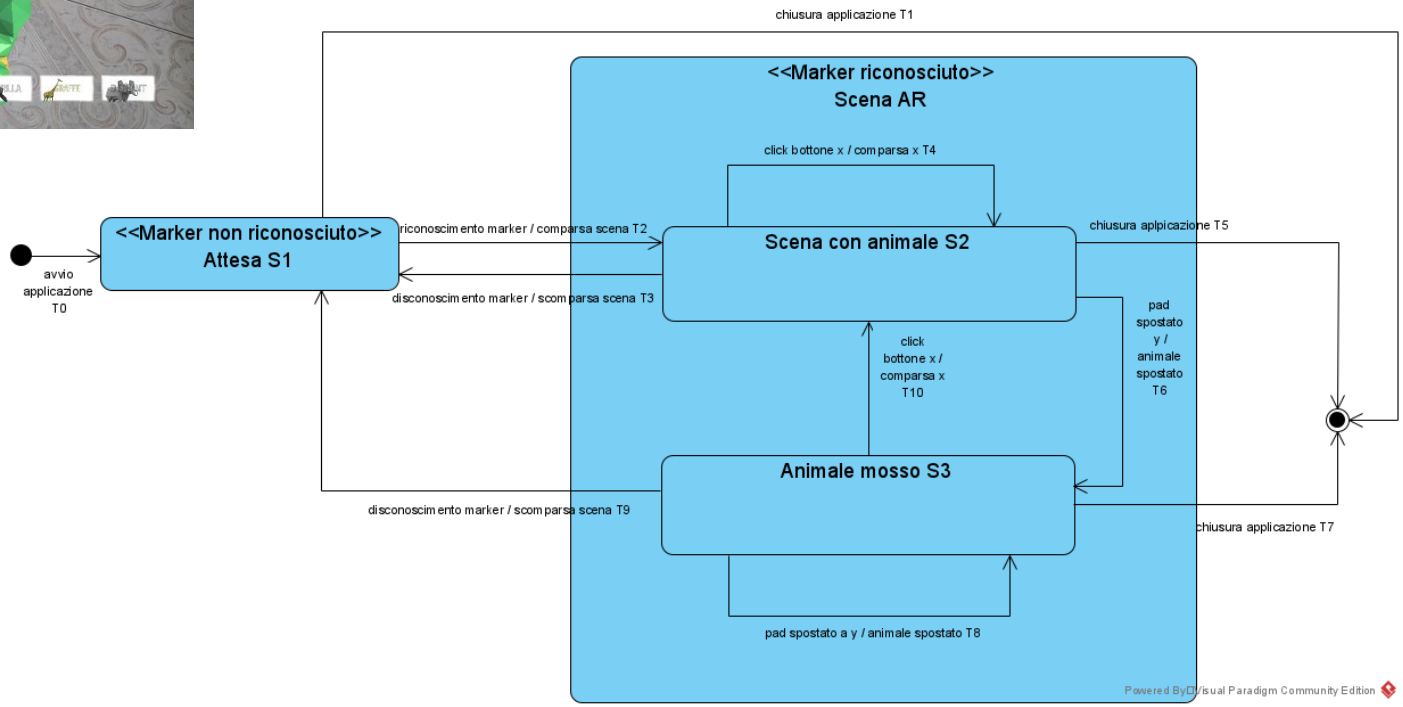
Caso di studio 1: Safari Animal AR

Modellazione



X = {zebra, rinoceronte, leone, gorilla, giraffa, elefante}

Y = {su, giù, destra, sinistra}



Caso di studio 1: Safari Animal AR

Analisi di copertura e qualità test suite

						Branch code coverage			Rilevazione Malfunzionamento		
		N° Test	Percentuale Stati coperti	Percentuale Transizioni coperte	Tempo di esecuzione	AutoFocus	AnimalSpawn	AnimalsController	Mutante 1	Mutante 2	TER
Test suite	All States Coverage	2	100%	8,7%	12s	75%	40%	75%	Si	No	0,5
	All Transitions Coverage	23	100%	100%	1m 13s	75%	80%	100%	Si	Si	1
	All One Loop Paths Coverage	218	100%	100%	46m 37s	75%	80%	100%	Si	Si	1

- Mutante 1: malfunzionamento che impatta sul movimento del pad invertendo le direzioni di spostamento (destra e sinistra)
- Mutante 2: malfunzionamento che viola il vincolo di esclusività di presenza di un animale sulla scena

Caso di studio 1: Safari Animal AR

Test suite migliore

					Branch code coverage			Rilevazione Malfunzionamento		
	N° Test	Percentuale Stati coperti	Percentuale Transizioni coperte	Tempo di esecuzione	AutoFocus	AnimalSpawn	AnimalsController	Mutante 1	Mutante 2	TER
All Transitions Coverage	23	100%	100%	1m 13s	75%	80%	100%	Si	Si	1

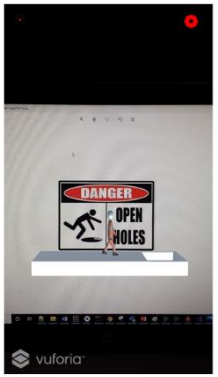
Miglioramento test suite  dopo analisi di copertura

					Branch code coverage			Rilevazione Malfunzionamento		
	N° Test	Percentuale Stati coperti	Percentuale Transizioni coperte	Tempo di esecuzione	AutoFocus	AnimalSpawn	AnimalsController	Mutante 1	Mutante 2	TER
All Transitions Coverage estesa	24	100%	100%	1m 15s	100%	80%	100%	Si	Si	1

Caso di studio 2: Point AR

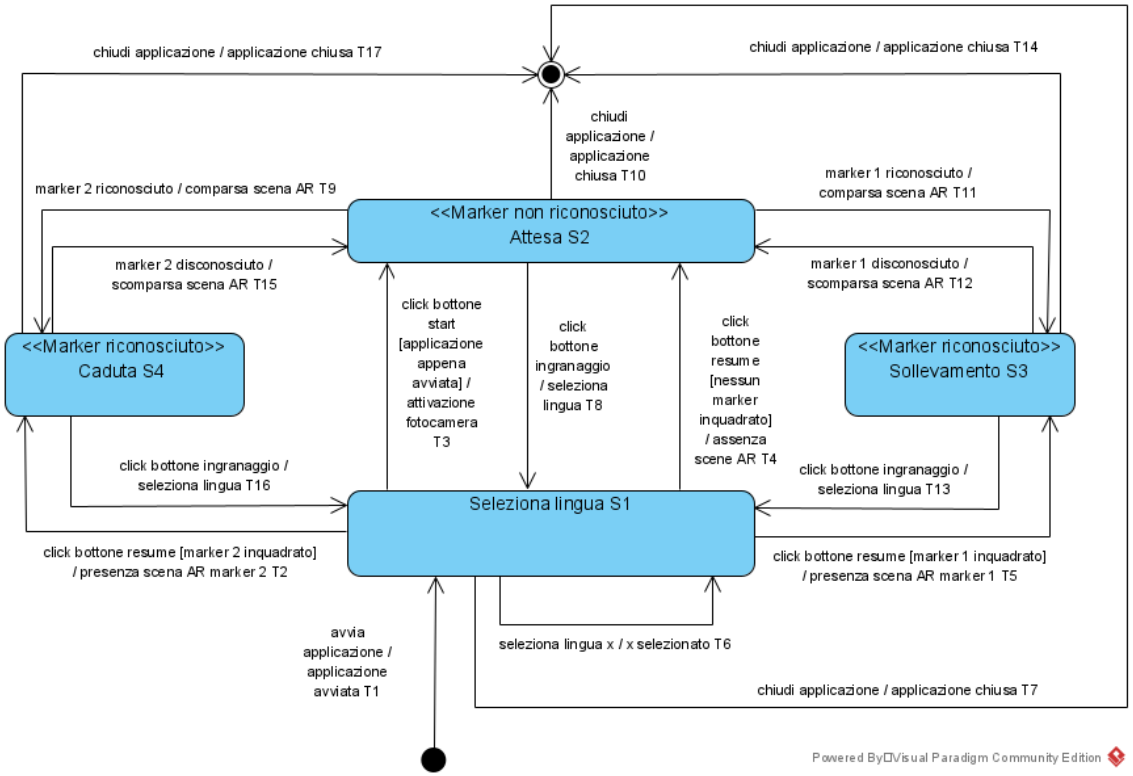
Modellazione

a) Scena AR 1



b) Scena AR 2

X = {Inglese, Italiano, Lituano, Urdu}



Caso di studio 2: Point AR

Analisi di copertura e qualità test suite

						Branch code coverage			Rilevazione Malfunzionamento		
		N° Test	Percentuale Stati coperti	Percentuale Transizioni coperte	Tempo di esecuzione	AutoFocus	Translate	Playbackspeed	Mutante 1	Mutante 2	TER
Test suite	All States Coverage	4	100%	20%	28s	75%	12,5%	100%	No	No	0
	All Transitions Coverage	15	100%	100%	1m 18s	75%	62,5%	100%	Si	No	0,5
	All One Loop Paths Coverage	64	100%	100%	10m 20s	75%	62,5%	100%	Si	Si	1

- Mutante 1: malfunzionamento che impatta sull'apertura del menù a tendina per selezionare la lingua della traduzione
- Mutante 2: malfunzionamento relativo all'errata traduzione, selezionando la lingua lituana, del marker inerente al sollevamento dei carichi

Caso di studio 2: Point AR

Test suite migliore

					Branch code coverage			Rilevazione Malfunzionamento		
	N° Test	Percentuale Stati coperti	Percentuale Transizioni coperte	Tempo di esecuzione	AutoFocus	Translate	Playbackspeed	Mutante 1	Mutante 2	TER
All One Loop Paths Coverage	64	100%	100%	10m 20s	75%	62,5%	100%	Si	Si	1

Miglioramento test suite  dopo analisi di copertura

					Branch code coverage			Rilevazione Malfunzionamento		
	N° Test	Percentuale Stati coperti	Percentuale Transizioni coperte	Tempo di esecuzione	AutoFocus	Translate	Playbackspeed	Mutante 1	Mutante 2	TER
All One Loop Paths Coverage estesa	66	100%	100%	11m 3s	100%	75%	100%	Si	Si	1

Conclusioni

- Fattibilità dell'approccio adottato, confermato dai risultati dei casi di studio mostrati in precedenza
- Validità delle metriche proposte utili al fine di scegliere la test suite migliore in termini di efficacia ed efficienza
- Automazione nell'esecuzione delle test suite in modo tale da evitare l'interazione, ripetitiva, da parte di tester umani

Sviluppi futuri

- Tecniche di reverse engineering per la modellazione dell'applicazione in esame
- Implementazione automatica della test suite relativa allo specifico criterio di copertura applicato sul modello;
- Estensione del programma «Analisi Log» con la funzionalità di inserimento automatico delle sonde negli script;
- Miglioramento dei tool di testing viste le limitazioni attualmente presenti (inserimento di input sonori, accesso a tutte le componenti dei GameObject)

Grazie per l'attenzione