Simulazione in VR tramite Google Cardboard del Museo di Capodimonte

Mariano Pascarella
M63000098

Contents

1 Premessa

Il progetto che ho realizzato nell'ambito del corso di *Sistemi Multimediali*, consiste in un'applicazione che simula, tramite realtà virtuale, la visita di una sala del Museo di Capodimonte.

Il motivo che mi ha portato a sviluppare questo tipo di applicazione è stato quello di voler conoscere ed usare la piattaforma di sviluppo *Google Cardboard*. Fin dalla sua presentazione, infatti, sono stato molto colpito dalla semplicità con la quale si poteva provare l'esperienza della realtà virtuale.
L'SDK fornito da Google e il visore Cardboard, sono nati proprio per incoraggiare lo sviluppo di applicazioni in ambito VR. Bastano pochi euro per acquistare un visore o addirittura usare oggetti economici e facilmente reperibili per realizzarne uno.

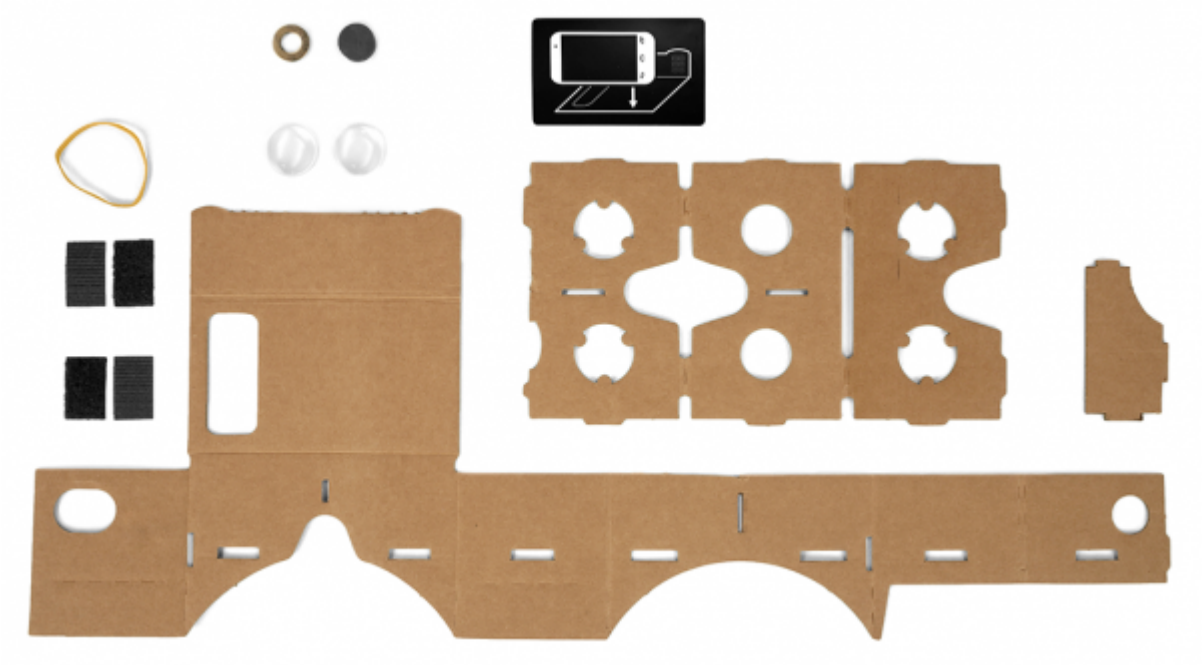

Fig. 1: Esempio Cardboard da assemblare

2 Introduzione

L'applicazione sviluppata permette si simulare, mediante l'uso della realtà virtuale, la visita di un museo, in particolare la Sala 2 del Museo di Capodimonte, dove sono conservate le opere di Tiziano appartenenti alla collezione Farnese.

Questa esperienza di realtà virtuale è ottenuta mediante la tecnologia sviluppata da Google, che attraverso la piattaforma di sviluppo *Google VR SDK* e al visore *Cardboard,* permette di accedere alla realtà virtuale sfruttando uno smartphone Android o IOS, in modo semplice ed economico.

Fig. 2: Google Cardboard VR

Google non fornisce solo l'SDK per Android ma ne implementa anche uno per la piattaforma Unity, un ambiente di sviluppo per realizzare giochi interattivi in ambienti 3d che possono essere facimente esportati su qualsiasi piattaforma.

Unity si è rivelato essere un potente strumento, permettendo di usare un unico ambiente di sviluppo per la realizzazione dell'intera applicazione. Grazie ad Unity è stato possibile realizzare l'ambiente 3d, ricreando quasi fedelmente l'architettura della sala 2 del museo, ed implementare l'interazione con l'ambiente virtuale mediante l'uso delle librerie di Google contenute nell'SDK per Unity.

Per quanto riguarda la gestione degli elementi multimediali come immagini e testi, è stato usato SQLite, un DBMS molto leggero tanto da essere usato come soluzione embedded per applicazioni che girano su sistemi con limitate risorse. L'interfacciamento al database è stato ottenuto usando codice C# che ha consentito l'uso delle librerie .dll presenti nel MonoDevelop-Unity editor.

Durante la fase preliminare del progetto sono state raccolte molte informazioni relative all'uso di Unity e del Google VR SDK. Durante questa fase sono state eseguite diverse prove per realizzare delle Scene Unity, poi eseguite come App androind e visualizzzate con l'ausilio di un visore Cardboard. Inizialmente ci sono stati diversi problemi dovuti alla fase non ancora matura di integrazione tra Unity e l'SDK VR, fortunatamente superati anche se con non poche difficoltà.
La seconda fase è stato basata sull'acquisizione del materiale multimediale per l'appilcazione, compreso il rilevamento delle misure della sala e dei diverse materiali con la quale poi realizzare l'architettura della Scena mediante Unity.
La terza fase è stata dedicata ad integrare l'uso di un DBMS per la gestione degli elementi multimediali, che ha introdotto una notevole semplificazione dell'architettura oltre al fatto di poter risparmiare risorse in termini di occupazione di memoria. Infatti, quando Unity compila l'applicazione gestisce i diversi Assets, come le immagini, in modo poco efficiente, mentre la loro gestione in un DBMS ne riduce l'occupazione di memoria.
Questa scelta ha permesso anche di ottenere un certo dinamismo nel progetto, infatti, è possibile usare un diverso database per ottenere la stessa scena con diversi elementi multimediali.

3 Architettura

Come detto nell'introduzione l'applicazione permette di muoversi all'interno di un ambiente 3d che simula la sala 2 del museo di Capodimonte. La navigazione avviene mediante l'uso di uno smartphone, dove viene eseguita l'applicazione, e del visore Cardboard, quest'ultimo permette di trasformare un qualsiasi smartphone in un visore per la realtà virtuale.
Le interazione con il mondo virtuale avvengono quasi tutte attraverso lo smartphone che permette, sfruttando il suo giroscopio, di muoversi all'interno della scena e osservare a 360° quello che circonda il soggetto.

L'applicazione, come mostrato in figura, è composta da tre Scene, realizzate mediante Unity. Quest'ultimo integra il Google VR SDK che provvede ad introdurre:

- un supporto per il tracciamento dei movimenti del Visore
- un rendering stereoscopico della scena
- un gestore delle interazioni da parte dell'utente attraverso trigger o pulsanti virtuali

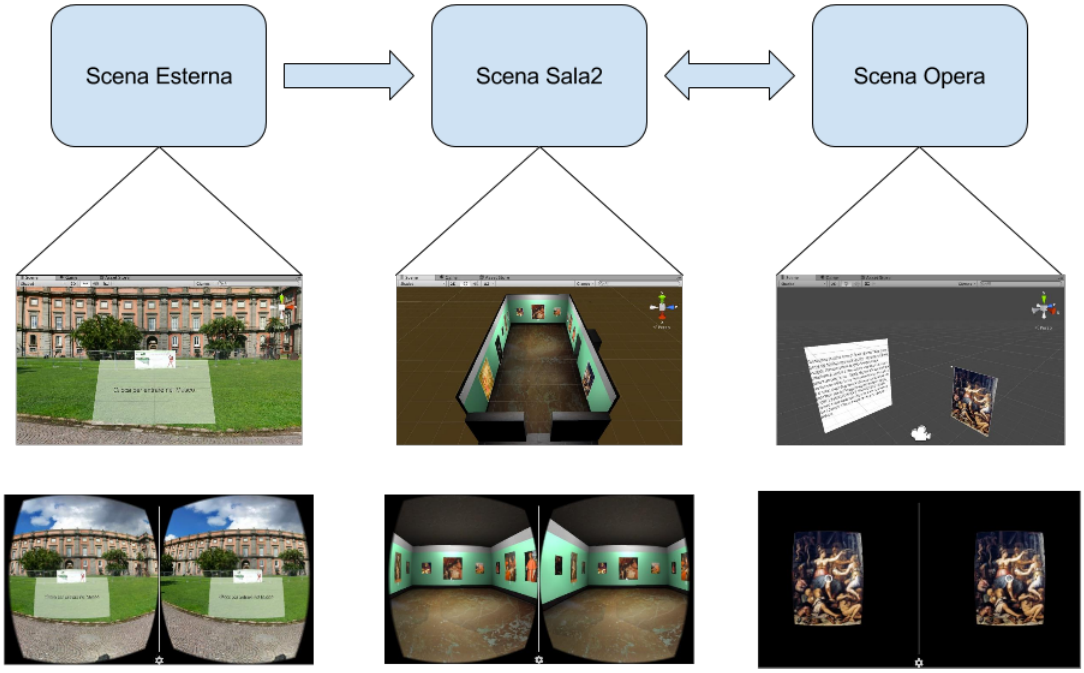

Fig. 3: Architettura Scene

3.1 Scena Esterna

La scena Esterna è la prima che viene visualizzata dall'applicazione, essa ha lo scopo di introdurre l'utente nell'ambiente VR mostrando una vista a 360° dell'esterno del museo. L'immagine è stata acquisita sul posto attraverso un dispositivo che supporta le photo sphere, una particolare immagine panoramica che riesce a rappresentare a 360° il mondo circostante.
Una volta in possesso dell'immagine panoramica stereoscopica, si passa a:

1. settare Unity importanto i Plugins e i Prefabs del Google VR SDK.
2. importare l'immagine panoramica a 360° e modificare, nell'inspector, la Texture Type da Sprite a Cubemap. Poi selezionare il Mapping con Latitude_Longitude Layout Cylindrical. Dopo di chè è possibile creare un

nuovo Material e inserire come Texture l'immagine importata. Alla fine di ciò è possibile modificare le proprietà dello Skybox da Windows/Lighting andando a selezionare la nostra immagine.

3. Come ultimo passaggio si sostituisce la Camera di default di Unity con la Camera implementata da Google, Quest'ultima può essere inserita nella scena copiando il Prefab GvrMain contenuto nella cartella Google.

3.1.1 GvrMain

A differenza della normale Camera usata da Unity per sviluppare giochi o applicazioni, la camera GvrMain permette di simulare la visione Binoculare del sistema visivo Umano, con la quale è possibile trasmettere l'illusione della tridimensionalità. Tutto questo è ottenuto attraverso la tecnica della Stereoscopia, nota già dal XV secolo.

Il principio è basato sulla visione binoculare che permette ai nosrti occhi di percepire la realtà a tre dimensioni. Gli occhi vedono lo stesso soggetto da due posizioni differenti, il cervello unisce queste due immagini ed elabora la profondità.
Per poter riprodurre l'effetto proprio della visione binoculare è perciò necessario creare una illusione: l'illusione stereoscopica, per creare la quale è necessario disporre di due immagini del medesimo soggetto riprese alla stessa distanza ma scostate lateralmente con uno scarto pari alla distanza binoculare (stereoscopia naturale) o ad una maggiore o minore distanza (stereoscopia artificiale).

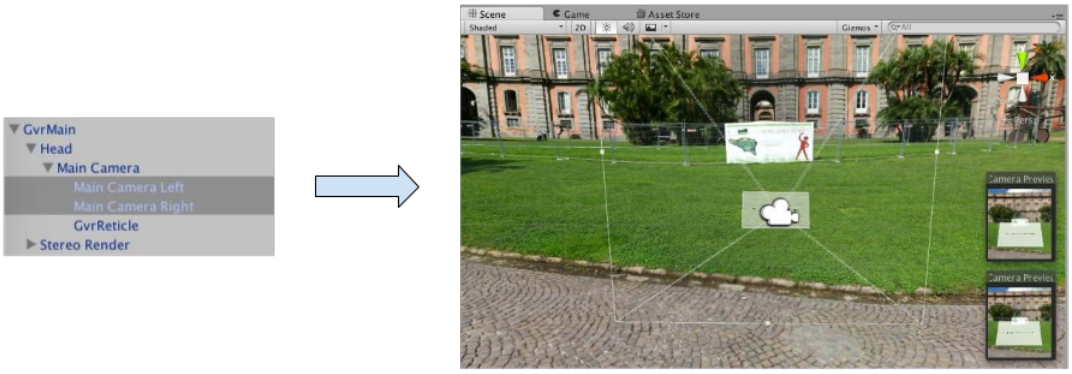

Fig. 4: GvrMain con doppia Camera

Quindi la GvrMain usata nella nostra scena non fa altro che implementare due camere leggermente sfasate, cosicché la nostra applicazione non mostrerà un unica scena ma due, una per ogni occhio.

Fig. 5: Scena mostrata dall'applicazione

3.1.2 Interazione con l'App

Per poter interagire con la scena e quindi con l'applicazione, c'è bisogno di aggiungere degli script alla Camera.

GvrHead è uno script che è collegato direttamente con l'oggetto GvrMain, questo permette di rilevare i movimenti della testa nel mondo reale e riprodurli nel mondo virtuale.
Per ricreare fedelmente i movimenti si esegue un headTracking sfruttando il giroscopio dello smartphone e successivamente si aggiorna la scena con continuità.

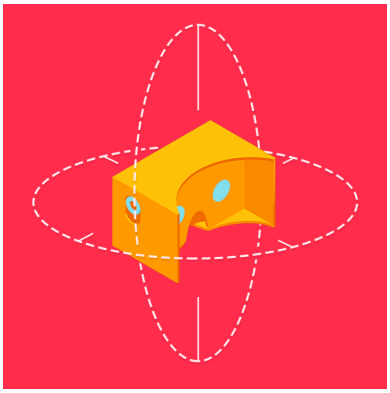

Fig. 6: Giroscopio

GazeInputModule è uno script che provvede ad implementare la gestione degli input, in questo modo è possibile interagire con interfaccie basate sui Canvas UI, elementi che possono essere selezionati semplicemente guardandoli o cliccati attraverso trigger visivi. Quest'ultima operazione è ottenuta toccando

lo schermo, cosa che avviene attraverso l'unico tasto presente sul cardboard.
A questo punto se si vuole interagire con altri oggetti 3d presenti nella scena bisogna aggiungere al GvrMain un **PhysicsRaycaster**, un componente che implementa un raggio con la quale creare collisioni e quindi interagire con gli oggetti che vengono fissati.
Gli eventi che GazeInputModule sviluppa sono: Enter, Exit, Down, Up, Click, Select, Deselect, e UpdateSelected. Scroll, move. Eventi alla quale è possibile assegnare degli script.

GvrReticle è uno script collegato alla MainCamera, permette di disegnare un cerchio davanti all'oggetto che viene fissato. Inoltre, il cerchio si dilata se l'oggetto è cliccabile, cioè sugli oggetti che presentano un Collider e un Event Trigger.

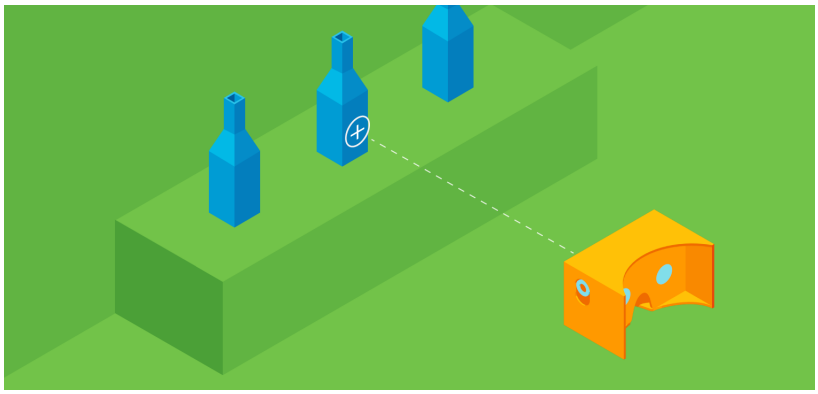

Fig. 7: Reticle

Nella nostra scena abbiamo un unico elemento con la quale è possibile interagire, si tratta del Canvas che realizza l'interfaccia utente con la quale si può accedere all'interno del museo. L'interazione avviene impostanto sull'oggetto interagente un Event Trigger, l'evento scelto è di tipo onClick che invoca un metodo appartenente alla classe LoadScene, quest'ultima è legata alla vita stessa dell'oggetto.

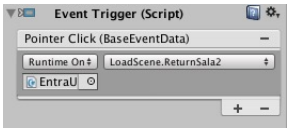

Fig. 8: es. Event Trigger

```
public void ReturnSala2(){
                SceneManager.LoadScene("Sala2");
}
```

3.2 Scena Sala2

La scena Sala2 costituisce l'elemento centrale dell'aplicazione, essa riproduce la sala del museo permettendo di interagire con i diversi quadri presenti. Anche in questo caso si utilizza la GvrMain Camera dell'SDK di Google che permette di simulare l'ambiente virtuale con una visuale stereoscopica. L'impressione che si ottiene è quella di trovarsi realmente nell'ambiete virtuale, dove l'utente può scegliere di spostarsi, grazie alla classe AutoWalk, oppure interagire con i quadri in modo da ricevere maggiori informazioni.

Attraverso Unity si realizzano gli elementi strutturali della scena, che oltre alle semplici pareti, pavimento e tetto, si compone anche dei Quadri. Quest'ultimi oltre che a mostrare l'opera, rappresentano anche gli elementi di interazione con l'utente.

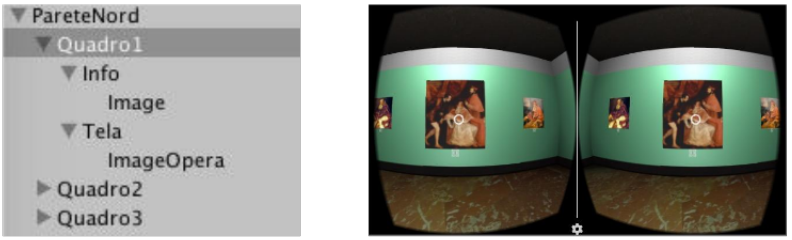

Fig. 9: Oggetto Quadro

L'oggetto Quadro è costituito da Canvas che rappresentano gli elementi della UI, implementano diversi tipi di interazione con l'utente. Abbiamo un icona che reagisce quando viene puntata e quando si smette di fissarla, nel primo caso vengono mostrate delle informazioni sintetiche riguardanti l'opera, nel secondo caso vengono nascoste.
L'altro elemento di interazione è la tela, dove viene mostrata l'opera e da dove è possibile aprire una nuova scena, tramite un click, dove si ha la possibilità di osservare l'opera più dettagliatamente. Queste interazioni sono gestite da due classi:

- LoadScene
- InfoOpere

Una terza classe, LoadOpereSala, permette di riempire i quadri nella sala. Infatti, la sala inizialmente appare vuota, non ci sono immagini ma solo spazi bianchi. Le immagini relative alle opere sono caricate a run time andando a recuperare gli elementi multimediali direttamente da un database.
Questa scelta progettuale si rivela essere molto importante dal momento in cui si vuole realizzare una nuova sala o semplicemente cambia la disposizione dei quadri, basta infatti aggiornare il DataBase e la sala rifletterà i cambiamenti

andando anche a modificare le dimensioni dei Canavas che ospitano le nuove opere.

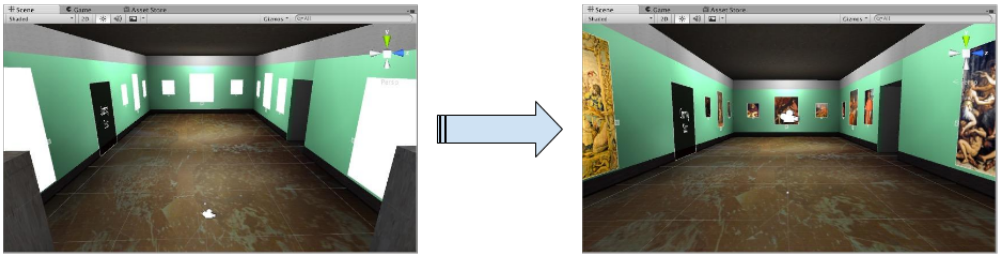

Fig. 10: Caricamento Immagini

3.2.1 AutoWalk

AutoWalk è una classe che permette di compiere movimenti all'interno della scena. Essa è legata alla Camera e permette, tramite un inclinazione, di spostarsi in avanti nella direzione che si sta fissando. La classe permette di impostare la velocità di movimento e l'angolo di soglia oltre la quale iniziare a camminare. Lo spostamento è ottenuto attraverso questi semplici comandi aggiornati continuamente.

```
Vector3 direction = new Vector3(head.transform.forward.x, 0, head.transform.forward.z).normalized *
            speed * Time.deltaTime;
Quaternion rotation = Quaternion.Euler(new Vector3(0, -transform.rotation.eulerAngles.y, 0));
transform.Translate(rotation * direction);
```

L'ultimo funzione permette di spostarsi di una certa distanza in una certa direzione.

3.2.2 DataBase

Un altro elemento presente in questa scena è uno script per recuperare le immagini dal database, LoadOperaSala. Quest'ultimo è collegato con le Tele di tutti i quadri e riceve un valore idIm diverso in modo da eseguire la query per ottenere la giusta opera da mostrare.

```
string filePath = Application.persistentDataPath + "/MuseoCapodimonte.db";
connectionString = "URI=file:" + Application.persistentDataPath + "/MuseoCapodimonte.db";

connDB.Open();
string sqlQuery = "SELECT immagine FROM Opere WHERE codIm =" + idIm;
commDB.CommandText = sqlQuery;
while (reader.Read()) {
        byte[] image = (byte[])reader["immagine"];
        textureI = new Texture2D(2, 2);
    textureI.LoadImage(image);
    tempoI = Sprite.Create (textureI, new Rect (0, 0,textureI.width,
textureI.height), new Vector2 (0.5f, 0.5f), 1.0f);
        imageSqlite.sprite = tempoI;
        connDB.Close();
        reader.Close();
```

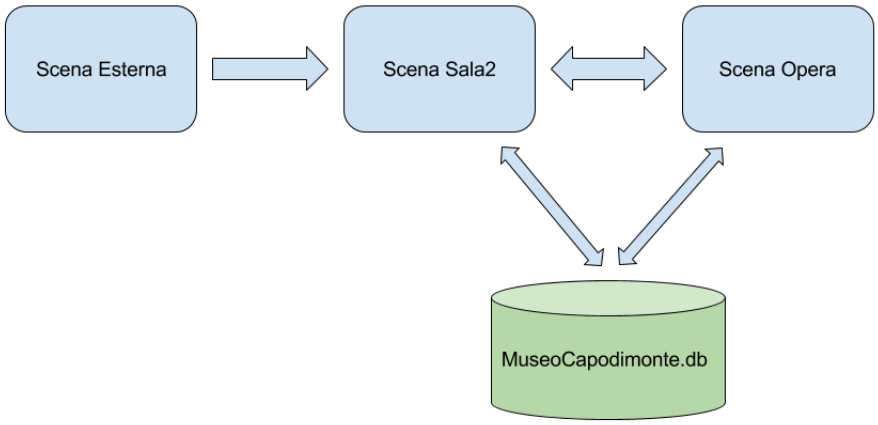

Fig. 11: Scene e SQLite

Il dataBase usato è gestito SQLite, un DBMS molto leggero che può essere incorporato in applicazioni che girano su hardware limitato come quello degli smartphone. Per funzionare su android il database sfrutta le librerie già presenti in android dato che quest'ultimo fa ampio uso di questa soluzione.
Il database che viene incorporato nell'app è costituito da una sola tabella dove sono contenuti i dati relativi ad ogni quadro. L'immagine e i testi essendo contenuti multimediali, e quindi dati non strutturati, sono gestiti attraverso Blob.

Opere		CREATE TABLE `Opere` (`codIm` INTEGER, `titolo` TEXT, `autore` TEXT, `nSala` INTEGER, `descrizione` TEXT, `immagine` BLOB NOT NULL, PRIMARY KEY(codIm))
codIm	INTEGER	`codIm` INTEGER
titolo	TEXT	`titolo` TEXT
autore	TEXT	`autore` TEXT
nSala	INTEGER	`nSala` INTEGER
descrizione	TEXT	`descrizione` TEXT
immagine	BLOB	`immagine` BLOB NOT NULL

Fig. 12: Schema Opera

Il database MuseoCapodimonte.db è gestito tramite le librerie Mono.Data.Sqlite-Client e System.Data, attraverso codice c#.

3.3 Scena Opera

L'ultima scena realizzata è Opera, essa viene caricata quando l'utente clicca su un quadro per visualizzare maggiori informazioni, infatti in questa scena si ha una visione più pulita dell'immagine e si può accedere ad una breve descrizione. La scena è unica, ma questo non vieta che possa riprodurre 11 opere diverse, infatti gli elementi multimediali mostrati nella scena sono recuperati da un database mediante una quesry. Quest'ultima sfrutta un parametro passato dalla sala 2 per effettuare la giusta query, ovvero mostrare l'opera in funzione del quadro con la quale si vuole interagire.

4 Codice