Documentazione

1 Introduzione 4

1.1 Informazioni sul progetto 4

1.2 Abstract 4

1.3 Scopo 4

2 Analisi 5

2.1 Analisi del dominio 5

2.2 Analisi e specifica dei requisiti 5

2.3 Use case 6

2.4 Pianificazione 1

2.5 Analisi dei mezzi 1

2.5.1 Software 1

2.5.2 Hardware 1

3 Progettazione 2

3.1 Design dell’architettura del sistema 2

3.2 Design dei dati e database 4

3.3 Design delle interfacce 5

3.4 Design procedurale 5

4 Implementazione 7

4.1 Menu e Interfacce Grafiche 7

4.1.1 Start Page GUI 8

4.1.2 GUI di configurazione del foglio (GUI “Configurazione Tela”) 9

4.1.3 Creazione file 10

4.1.4 Impostazioni Main Camera 11

4.1.5 Caricamento immagini 11

4.1.6 Caricamento tela disegnabile 11

4.2 Leap Motion 12

4.2.1 Prova Leap Motion Controller 12

4.2.2 Tracciamento mani 13

4.2.3 Lettura movimenti 15

4.2.4 HandReset e HandFinish 15

4.3 Disegno 16

4.3.1 Azione Disegno 16

4.3.2 Selezione colore 16

4.3.3 Selezione strumento e dimensione della penna 18

4.3.3.1 Visualizzazione dello strumento in uso 18

4.4 Azioni varie ed inventario 19

4.4.1 Selezione strumento 19

4.4.2 ColorPicker 19

4.4.3 Exit 20

4.4.4 Salvataggio 20

4.4.5 Salvataggio con nome 20

4.4.6 Impostazioni 21

4.4.7 Ctrl + Z 21

4.4.8 Riempimento 21

4.4.9 Rotazione della tela 21

4.4.10 Zoom della tela 21

4.4.11 Info Panel 22

5 Test 23

5.1 Protocollo di test 23

5.2 Risultati test 26

5.3 Mancanze/limitazioni conosciute 26

6 Consuntivo 26

7 Conclusioni 26

7.1 Sviluppi futuri 26

7.2 Considerazioni personali 26

8 Bibliografia 26

8.1 Sitografia 26

9 Allegati 26

# Introduzione

## Informazioni sul progetto

|  |  |
| --- | --- |
| Allievi coinvolti nel progetto : | Karim Galliciotti, Zeno Darani, Stefano Mureddu e Sara Bressan |
| Docente responsabile : | Luca Muggiasca |
| Classe : | 3 AA-AC della sezione di Informatica della Scuola Arti e Mestieri di Trevano |
| Data di inizio del progetto : | 2021-01-14 |
| Data di fine del progetto : | 2021-05-20 |

## Abstract

E’ una breve e accurata rappresentazione dei contenuti di un documento, senza notazioni critiche o valutazioni. Lo scopo di un abstract efficace dovrebbe essere quello di far conoscere all’utente il contenuto di base di un documento e metterlo nella condizione di decidere se risponde ai suoi interessi e se è opportuno il ricorso al documento originale.

Può contenere alcuni o tutti gli elementi seguenti:

* Background/Situazione iniziale
* **Descrizione del problema e motivazione**: Che problema ho cercato di risolvere? Questa sezione dovrebbe includere l'importanza del vostro lavoro, la difficoltà dell'area e l'effetto che potrebbe avere se portato a termine con successo.
* **Approccio/Metodi**: Come ho ottenuto dei progressi? Come ho risolto il problema (tecniche…)? Quale è stata l’entità del mio lavoro? Che fattori importanti controllo, ignoro o misuro?
* **Risultati**: Quale è la risposta? Quali sono i risultati? Quanto è più veloce, più sicuro, più economico o in qualche altro aspetto migliore di altri prodotti/soluzioni?

Esempio di abstract:

As the size and complexity of today’s most modern computer chips increase, new techniques must be developed to effectively design and create Very Large Scale Integration chips quickly. For this project, a new type of hardware compiler is created. This hardware compiler will read a C++ program, and physically design a suitable microprocessor intended for running that specific program. With this new and powerful compiler, it is possible to design anything from a small adder, to a microprocessor with millions of transistors. Designing new computer chips, such as the Pentium 4, can require dozens of engineers and months of time. With the help of this compiler, a single person could design such a large-scale microprocessor in just weeks.

## Scopo

Lo scopo di questo progetto consiste nello sviluppo di una tavolozza virtuale sulla quale sarà possibile disegnare a mano libera utilizzando un Leap Motion. Nella tavolozza sarà quindi possibile selezionare gli strumenti di disegno (ad esempio pennelli, matite ...) e selezionare il colora da utilizzare nel disegno. I movimenti delle mani corrisponderanno a determinate azioni sulla tavolozza (es. tracciare una riga), inoltre i disegni creati saranno salvabili ed esportabili su file separati.

Lo scopo di progetto consiste inoltre nell'imparare a gestire dei progetti in team e imparare ad utilizzare nuove risorse.

# Analisi

## Analisi del dominio

È stato richiesto lo sviluppo di un'applicazione per la creazione di contenuti grafici, attraverso l'utilizzo di un sensore Leap Motion. L'applicazione dovrà permettere dunque l'interazione dell'utente con essa tramite il solo movimento delle mani. Il programma dovrà essere simile a Paint e semplice da utilizzare, inoltre dovrà dare la possibilità di salvare ed esportare le opere artistiche create. L'applicazione sarà stand alone e non necessiterà particolari conoscenze da parte degli utenti.

## Analisi e specifica dei requisiti

|  |  |
| --- | --- |
| ID: REQ-001 | |
| Nome | Tracciamento delle mani |
| Priorità | 1 |
| Versione | 1.0 |
| Note | Utilizzo del sensore Leap Motion |

|  |  |
| --- | --- |
| ID: REQ-002 | |
| Nome | Possibilità di selezione di diversi strumenti |
| Priorità | 1 |
| Versione | 1.0 |
| Note |  |

|  |  |
| --- | --- |
| ID: REQ-003 | |
| Nome | Possibilità di selezione il colore con cui disegnare |
| Priorità | 1 |
| Versione | 1.0 |
| Note |  |

|  |  |
| --- | --- |
| ID: REQ-004 | |
| Nome | È possibile disegnare a mano libera |
| Priorità | 1 |
| Versione | 1.0 |
| Note |  |

|  |  |
| --- | --- |
| ID: REQ-005 | |
| Nome | I disegni si possono salvare |
| Priorità | 1 |
| Versione | 1.0 |
| Note |  |

|  |  |
| --- | --- |
| ID: REQ-006 | |
| Nome | Applicativo stand alone |
| Priorità | 1 |
| Versione | 1.0 |
| Note |  |

|  |  |
| --- | --- |
| ID: REQ-007 | |
| Nome | Possibilità di importare immagini da modificare |
| Priorità | 2 |
| Versione | 1.0 |
| Note |  |

## Use case



## Pianificazione

Prima di stabilire una pianificazione bisogna avere almeno una vaga idea del modello di sviluppo che si intende adottare. In questa sezione bisognerà inserire il modello concettuale di sviluppo che si seguirà durante il progetto. Gli elementi di riferimento per una buona pianificazione derivano da una scomposizione top-down della problematica del progetto.

La pianificazione può essere rappresentata mediante un diagramma di Gantt:

|  |
| --- |
| Macintosh HD:Users:Guest:Downloads:VirtualPainter-main 2:Documentazione:Immagini:VirtualPainter.png  Figura 1: Esempio di diagramma di Gantt. |

## Analisi dei mezzi

### Software

* Google Chrome
* Opera
* MacDown
* Visual Studio Versione 2019
* Unity Versione 2019.4.1
* Libreria SDK di Leap Motion
* Git Hub
* Trello

### Hardware

* Mac A422-05 scolastico
* Mac A422-06 scolastico
* Mac A422-07 scolastico
* Mac A422-08 scolastico
* PC personale di Karim (HP Omen)
* PC personale di Sara (Acer Aspire)
* PC personale di Zeno (Acer)
* Leap Motion Controller fornito dal docente

# Progettazione

## Design dell’architettura del sistema



La applicazione si avvia e per prima cosa controlla che il Leap Motion sia collegato.

Se è collegato entra nel menu altrimenti da un errore e chiede di riavviare il programma.



L’applicazione entra nel menu e chiede se si vuole creare un nuovo file o caricarne uno.

Se si crea un nuovo file chiede di definirne le dimensioni e il tipo e poi entra nella pagina di disegno.

Se si carica un file controlla se è in regola e poi entra nella pagina di disegno.



L’applicazione apre le impostazioni e permette di selezionare con l’indice della mano dominante una delle seguenti cose, selezione strumento, selezione colore, impostazioni generali, salvataggio, chiusura.

Se viene selezionato salvataggio controlla se è il primo salvataggio del file.

In caso sia il primo apre l’explorer per definire dove salvarlo, altrimenti sovrascrive il file.



L’applicazione apre l’area di disegno e controlla se la mano debole ha il palmo girato verso l’alto.

Se sì apre le impostazioni, altrimenti permette alla mano di fare varie azioni.

Nello stesso momento controlla se la mano dominante è completamente aperta.

Se si permette a questa di fare varie azioni, altrimenti controlla la posizione delle dita.

Se queste sono chiuse a pugno si potrà cancellare, altrimenti se l’indice e il pollice sono collegati si potrà disegnare.



L’applicazione si chiude.

Prima di chiudersi controlla se c’è stata una nuova modifica dopo l’ultimo salvataggio.

Se sì apre le impostazioni di salvataggio, e una volta selezionate chiude il programma, altrimenti il programma viene chiuso direttamente.

## Design dei dati e database

Descrizione delle strutture di dati utilizzate dal programma in base agli attributi e le relazioni degli oggetti in uso.

Schema E-R, schema logico e descrizione.

Se il diagramma E-R viene modificato, sulla doc dovrà apparire l’ultima versione, mentre le vecchie saranno sui diari.

## Macintosh HD:Users:Guest:Downloads:interfacce.pngDesign delle interfacce

La nostra idea sulle interfacce dell’applicazione consiste nel avere un menu iniziale per caricare o creare un nuovo foglio, e successivamente una tela su cui disegnare.

Il menu iniziale, il primo in alto a sinistra nell’immagine, consiste in soli due grandi bottoni. Uno per creare un nuovo foglio bianco, e uno per caricare un immagine esistente. In caso si clicchi su crea nuovo foglio la schermata del menu cambierà in quella in basso a sinistra, in cui potranno venir inseriti grandezza e altezza desiderati, e continua per creare un nuovo foglio.

Una volta caricato o creato il foglio su cui lavorare ci troveremo sulla tela dove potremmo disegnare. La tela si trova in alto a destra nell’immagine. Sulla tela potremmo vedere il foglio di lavoro e le nostre mani e, sulla mano forte, in questo caso rappresentata da quella di destra, potremmo vedere il colore e lo strumento selezionato.

Il menu, nell’immagine in basso a destra, partendo da sopra verso il basso, permetterà di selezionare lo strumento, il colore, la mano forte, salvare, salvare con nome e uscire per tornare al desktop. Il menu comparirà automaticamente sulla mano debole quando questa è rivolta con il palmo verso l’alto.

## Design procedurale

Descrive i concetti dettagliati dell’architettura/sviluppo utilizzando ad esempio:

* Diagrammi di flusso e Nassi.
* Tabelle.
* Classi e metodi.
* Tabelle di routing
* Diritti di accesso a condivisioni …

Questi documenti permetteranno di rappresentare i dettagli procedurali per la realizzazione del prodotto.

# Implementazione

L’implementazione parte con la creazione del progetto su Unity con le prime impostazioni di base che poi verrà esportato su gli altri pc. Ogni volta che qualcuno completerà qualcosa sul proprio pc questa verrà esportata e implementata negli altri.

Prima di iniziare a creare il progetto bisogna impostare il timestep a 0.0111111 (l’equivalente di 90 frames per secondo) come consigliato dalla documentazione ufficiale del Leap Motion Controller. Questa impostazione è da cambiare in “Edit 🡪 Project Settings 🡪 Time”.

Per il corretto funzionamento del programma è necessario impostare la risoluzione dello schermo a 19:6, in questo modo la tela andrà a coprire interamente lo schermo.

## Menu e Interfacce Grafiche

Inizialmente pensavamo di creare diverse scene per ogni interfaccia grafica, ma in seguito per evitare eventuali errori durante il passaggio di informazioni tra scene abbiamo deciso di collassare tutto il programma in un’unica Scene (scena di Unity) chiamata “Tela”. Abbiamo quindi copiato e incollato i due Canvas già creati (“start\_page” e “configurazione\_tela”) nella scena appena creata.

A questo punto abbiamo creato un GameObject vuoto chiamato “ActionController” che si occupa di gestire e far partire i diversi script del programma non strettamente legati ad un oggetto (ad esempio lo script per creare la tela).

Per fare in modo che si possano selezionare i bottoni dei vari Canvas è necessario metterli in Layer differenti (uno più avanti e l’altro dietro).

Dopo aver creato le due GUI descritte nei capitoli seguenti abbiamo creato uno script nominato “ShowMenu” il quale gestisce la visualizzazione a schermo della GUI iniziale e nasconde ciò che non deve essere visto inizialmente come ad esempio le mani virtuali.

### Start Page GUI

Per prima cosa è stato creato il Canvas “start\_page”, che corrisponde alla GUI del primo menù che apparirà all’avvio del nostro programma.

Al canvas è stato impostato il render mode: Screen Space - Camera, in questo modo il canvas sarà sempre grande come la Main Camera, quindi grande come l'applicazione. Il Canvas ha inoltre un Vertical Layout Group, che consentirà di disporre il Label con il titolo del programma e i bottoni uno sopra l'altro, e sia l'altezza che la larghezza dei componenti all'interno di questo layout sono definiti dalla grandezza del Canvas (altezza e larghezza sono spuntate nelle impostazioni: "Control Child Size" e "Child Force Expand").

In seguito, abbiamo aggiunto nel Canvas un Label con scritto “Virtual Painter” e un pannello che andrà a contenere i due bottoni con la quale caricare o creare un nuovo file (tela).

Il pannello ha a sua volta un Horizontal Layout Group che serve a disporre i due bottoni in fila in modo che si adattino alla dimensione del pannello, in questo caso è stato inserito anche un padding di 10 su tutti i lati (in questo modo i bottoni sono separati tra loro e dai bordi della finestra).

I due bottoni nominati in precedenza consistono nel bottone “NuovoFoglio”, il quale servirà a creare una nuova tela, e il bottone “FoglioEsistente”, che servirà invece ad aprire l’explorer per selezionare un immagine preesistente da inserire nella tela.

Per inserire le immagini nei bottoni abbiamo importato un package dall’Asset Store di Unity (il pacchetto è gratuito ed è installabile dalla seguente pagina: <https://assetstore.unity.com/packages/2d/gui/icons/simple-icon-pastel-tone-107568>), il pacchetto è stato quindi importato tramite l’Unity Hub.

Ad ogni bottone è stato poi assegnato un Grid Layout Group (con “Chid Alignment” settato a “Middle Center”) nella quale sono stati inseriti un’immagine (presa dal Package installato in precedenza) e un testo.

La GUI appena creata serve all’utente per scegliere se aprire un file esistente sulla quale disegnare oppure se creare un nuovo file.

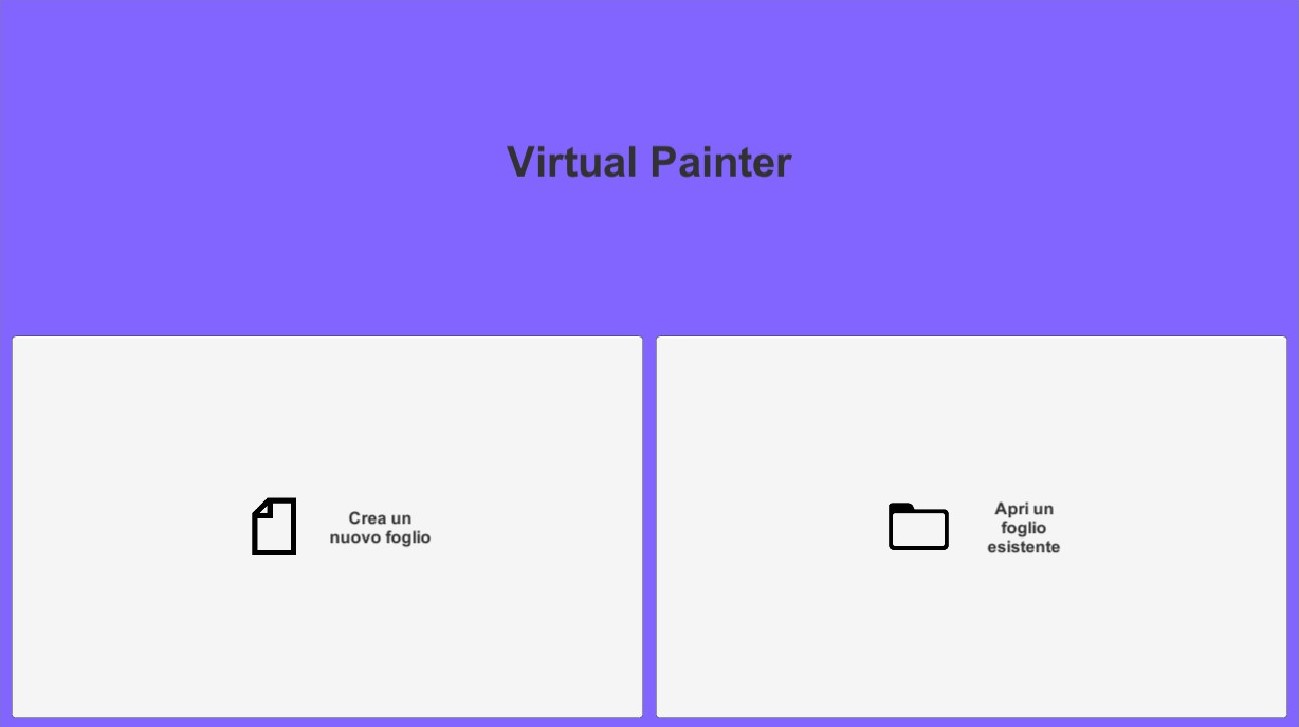


Figura 2 - Start Page GUI

Il menu implementato segue fedelmente il Design progettato e comprende i due bottoni con il titolo.

### GUI di configurazione del foglio (GUI “Configurazione Tela”)

Nella scena del progetto è stato aggiunto il Canvas “configurazione\_tela” fungerà da GUI per l’inserimento delle dimensioni del nuovo disegno che andrà creato (vedi capitolo “Creazione file”).

Abbiamo poi aggiunto al Canvas appena creato un Vertical Layout Group, il quale permette agli elementi al suo interno di essere messi uno sotto l’altro. All’interno del Canvas abbiamo quindi aggiunto i seguenti elementi in ordine: un Label (testo “Grandezza foglio”), un pannello che va a contenere gli Input Field e i Label di descrizione per definire le dimensioni della tela (disegno), infine abbiamo aggiunto il bottone “continua”, la dimensione delle componenti appena descritte viene definita dal Canvas che le contiene e hanno la grandezza massima contenibile dal Canvas stesso.

Gli Input Field che definiscono altezza e larghezza della tela accettano solamente numeri interi, il tipo di input accettato è stato selezionato nel componente sotto “Input Field” dove c’è scritto “Content Type”.

Fintanto che questa GUI è visibile bisogna disattivare l’azione “MoveTela”, altrimenti il premere i tasti sul tastierino numerico del PC potrebbe traslare la tela.



Figura - Configurazione Tela GUI

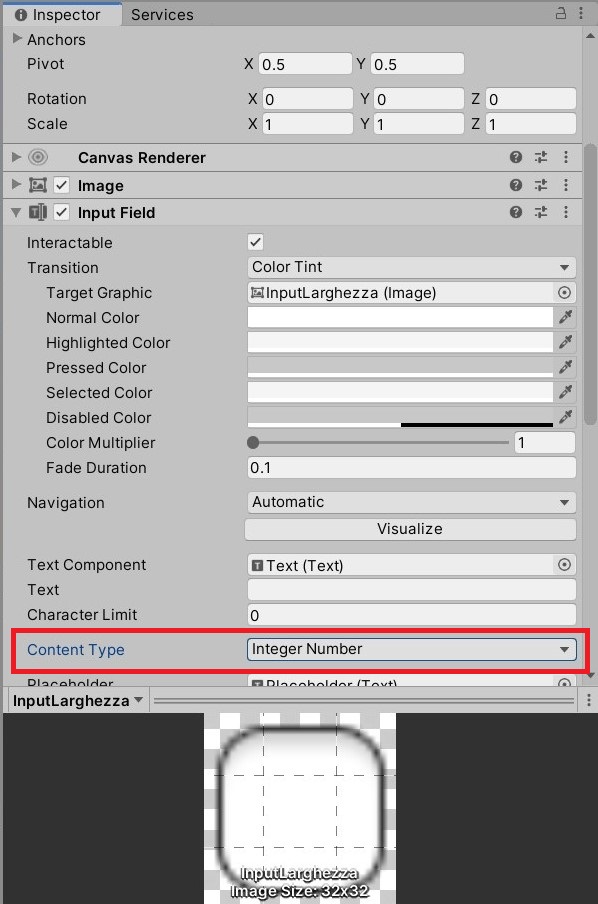


Figura - Impostazioni Input Field

### Creazione file

Per la creazione file si utilizza uno script richiamato cliccando il bottone continua del menu di creazione.

Lo script che si chiama “CreateFile” al momento del click prende i dati inseriti dai form “configurazione\_tela” e li converte in int, questi vengono poi forniti come parametri ai metodi per modificare la grandezza della tela e per creare una nuova texture da applicare ad essa.

Al momento del click il programma apre automaticamente l’Explorer del sistema operativo utilizzato e fa salvare all’utente il foglio appena creato con già un nome e una posizione. Successivamente crea una texture di dimensioni volute e la assegna all’oggetto.

Una volta fatto questo trasforma i dati che sono stati inseriti nei form come pixel e li trasforma in unità di Unity dividendoli per 100 così da poter modificare la grandezza dell’oggetto in modo corretto e scalato.

### Impostazioni Main Camera

La telecamera principale mostra ciò che l’utente finale vedrà a schermo, per questo tutti i componenti che andremo ad inserire devono essere centrati nella sua visuale. Per poter visualizzare sia le mani che la zela contemporaneamente abbiamo messa la telecamera principale in posizione x = 0, y = 0 e z = 3.066.

### Caricamento immagini

Per il caricamento delle immagini si utilizza uno script, che si chiama “GetFile”, che viene richiamato quando viene cliccato il bottone nel menù principale per importare immagini esterne. Al momento del click apre l’explorer del sistema operativo e permette di importare qualunque immagine PNG o JPG. Appena viene scelta l’immagine lo script si prende la path e, se non è mai stata utilizzata, la salva nel file “paths.json”, dove vengono salvati tutti i percorsi delle immagini modificate. Se invece il percorso è già presente nella lista lo sposta solamente all’ultima posizione. Dopodichè crea la texture basata sull’immagine presa, la applica alla tela, nasconde il menù principale e infine scala la tela in modo da avercela della dimensione massima possibile.

### Caricamento tela disegnabile

Il caricamento della tela disegnabile avviene scalando i pixel. In partenza la nostra tela non è altro che un GameObject a forma di rettangolo bianco chiamato “TelaDisegnabile” ed è posizionata nella scena principale in posizione x = 0, y = 140. La rotazione del GameObject corrisponde invece a x = 90, y = 180 e z = 0, mentre la scala iniziale (che verrà modificata automaticamente in seguito) corrisponde a x = 12, y = 14 e y = 12.

Una volta che i dati per la grandezza vengono inseriti nel form “configurazione\_tela” sulla tela verrà applicata una texture delle dimensioni volute. Per fare in modo che la tela e la texture applicata ad essa siano proporzionate bisogna ridimensionare la tela. Per farlo si rende la tela alta e larga quanto i pixels inseriti diviso 100. Inoltre, per fare in modo che la tela occupi tutto lo schermo, in orizzontale o in verticale, bisogna scalarla nuovamente ingrandendola o diminuendola mantenendo però sempre le proporzioni.

## Leap Motion

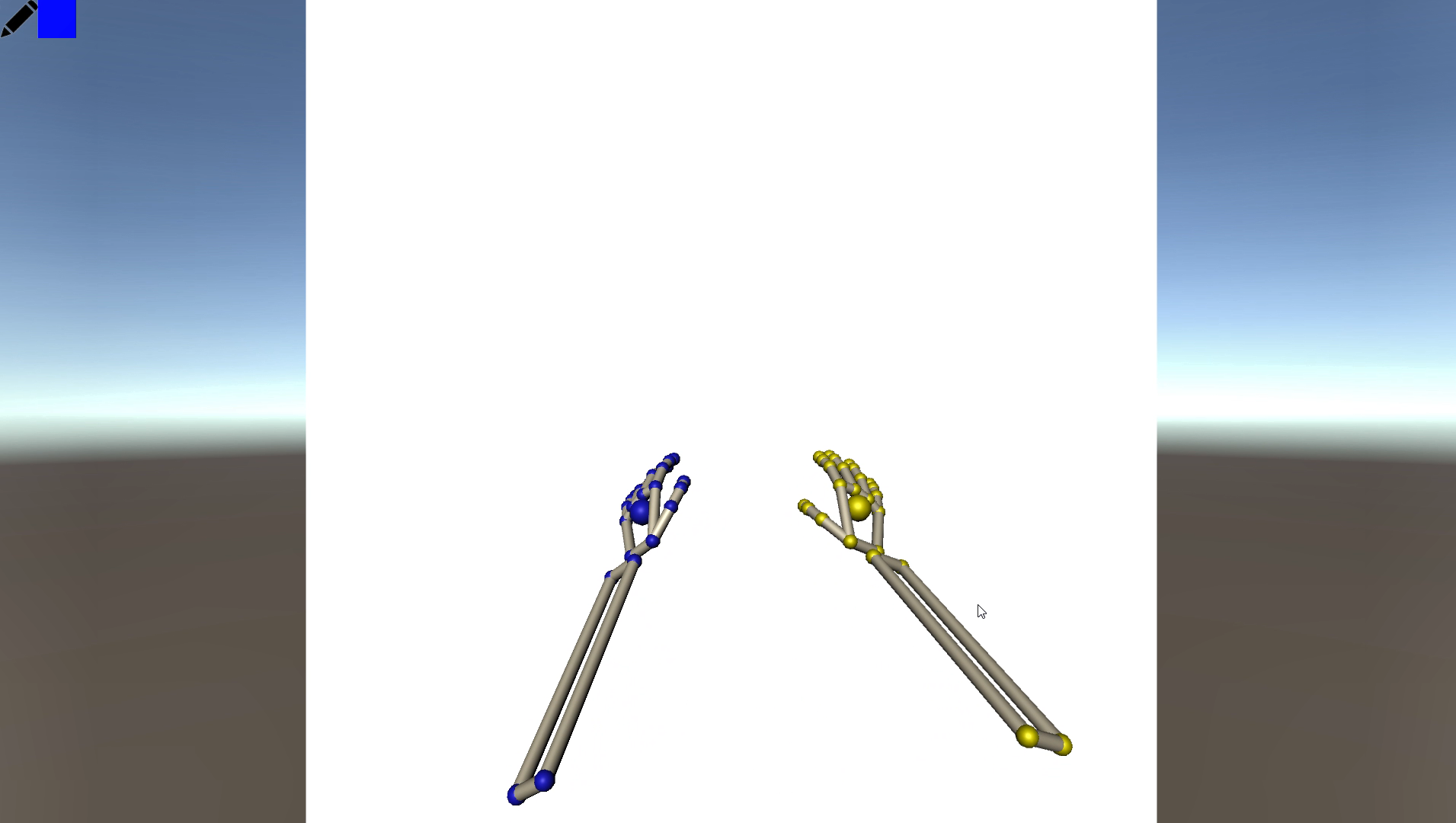


Figura 5 - Mani e tela

### Prova Leap Motion Controller

Per testare il funzionamento del Leap Motion lo abbiamo attaccato al portatile e abbiamo avviato il programma di test ufficiale scaricato insieme ad i pacchetti di base necessari a questo per funzionare.

I test sono stati un successo e Leap Motion vedeva correttamente le mani.

### Tracciamento mani

Per il tracciamento delle mani inizialmente bisogna scaricare gli Unity module packages[[1]](#footnote-1), ovvero i pacchetti per gli sviluppatori utilizzanti un Leap Motion. Una volta scaricati e importati si procede ad installare il plugin “XR plugin Management[[2]](#footnote-2)”, nel nostro caso la versione 3.2.16, e a selezionarlo sotto la scheda “Project Settings”.

Prima di creare il programma da zero si è provato a testare il funzionamento delle configurazioni settate, usando la scena di test già fornita dal Leap Motion “Capsule Hands (Desktop)[[3]](#footnote-3)”. In questo scenario abbiamo verificato che il Leap Motion interagisse con le mani presenti nella scena.

Una volta confermato il successo dei test siamo passati a ricreare la scena da soli.

Per prima cosa si trascina il prefab LeapHandController[[4]](#footnote-4) nella scena e si crea un nuovo GameObject di nome “HandsModel”. Questo dovrà contenere “RigidRoundHand\_L[[5]](#footnote-5)”, “RigidRoundHand\_R”, “Capsule Hand Left” [[6]](#footnote-6)e Capsule Hand Right”.

HandsModel dovrà poi utilizzare lo script “Hand Model Manager” e dopo aver modificato il campo Size di Models Pool da 0 a 2 dovranno comparire 2 nuovi gruppi di campi. Questi dovranno essere chiamati “Graphics\_Hands” e “Physics\_Hands”. Sotto “Graphics\_Hands, nei campi Left Model e Right Model dovranno venir selezionati “RigidRoundHand\_L” e “RigidRoundHand\_R”.

Bisogna inoltre spuntare in entrambi i gruppi solo la casella “IsEnabled”.

#### HandsModel

Dopo aver eseguito la prova descritta in precedenza ed esserci assicurati il corretto tracciamento dei movimenti abbiamo inserito l’HandsModel e il LeapHandController creato in precedenza all’interno di un GameObject vuoto chiamato “Hands” (Il quale funge quindi da contenitore per le due mani virtuali).  
La posizione di “Hands” equivale a x = -1.04, y = 0 e z = -0.04.

#### Attachment Hands

All’interno del HandsModel abbiamo inserito il prefab “Attachment Hands” che si trova nel progetto nella directory "Assets/Plugins/LeapMotion/Core/Prefabs" (vedi prefab presenti nel pacchetto del Leap Motion). All’interno di questo prefab si trovano due Attachment Hand (Left e Right), uno per la mano sinistra e uno per la mano destra. Nella mano sinistra abbiamo inserito un GameObject chiamato “Palm Forward Transform”, il quale verrà utilizzato in seguito come target per indicare la posizione della mano e del suo palmo.

In seguito nella mano sinistra, abbiamo inserito il GameObject "Palm UI Pivot Animation", il quale controlla la rotazione della mano per gestire la visualizzazione (o la non visualizzazione) dell'inventario (pannello di bottoni attaccato alla mano sinistra). All'interno del GameObject appena creato abbiamo messo due GameObject (Hidden e Visible) contenenti le coordinate che indicano la posizione per la quale l’inventario risulta visibile.

"Palm UI Pivot Animation" ha al suo interno lo script "Transform Tween Behaviour" fornito dal Package scaricato in precedenza del LeapMotion Controller, le proprietà inserite sono:

* Target transform: Palm Forward Transform (GameObject creato in precedenza)
* Start Transform: Hidden
* End Transform: Visible
* Le altre impostazioni sono quelle di default.

Dopo aver eseguito i passaggi precedenti abbiamo creato il "Palm UI Pivot Anchor" il quale andrà a contenere la grafica dell'inventario, al suo interno ha lo script "Simple Facing Camera Callbacks" il quale ha le seguenti caratteristiche:

* To face camera: Palm Forward Transform
* Camera to face: none
* On Begin Facing Camera(): -> Runtime Only -> Palm UI Pivot Animation -> TranformTweenBehaviour.PlayForward
* On End Facing Camera(): -> Runtime Only -> Palm UI Pivot Animation -> TranformTweenBehaviour.PlayBackward

All'interno di questo contenitore abbiamo messo il GameObject "Palm UI" avente lo script "Ignore Collisions In Children" che serve a prevenire collisioni tra gli elementi della UI.

Dentro "Palm UI" abbiamo inserito un GameObject chiamato "Button Panel" che corrisponde all'inventario. Dentro l'inventario ho inserito una serie di bottoni 3d (6 bottoni con immagini diverse per ogni funzione dell'inventario) creati con dei cubi alla quale è stato assegnato lo script "Interaction Button" (Script del Package del LeapMotion), il Manager indicato delle proprietà dello script corrisponde all'InteractionManager (vedi capitoli seguenti).

Per gestire in modo più efficace l’orientamento della mano e la conseguente visualizzazione del menù inventario abbiamo aggiunto una script chiamato “ManageLeft” nella mano “RigidRoundHand\_L” il quale gestisce i movimenti della mano sinistra, il codice controlla l’orientamento del palmo della mano (la direzione y del palmo deve essere maggiore o uguale a 0.6), a questo punto se il palmo risulta girato verso l’alto l’inventario risulta attivo (cioè visibile), altrimenti viene disattivato. Se la mano non è visualizzata di conseguenza bisogna rendere invisibile il menù inventario.   
Sempre nello script “ManageLeft” dopo aver controllato che la mano sinistra sia effettivamente visibile, viene controllato se entrambe le mani hanno il pollice e l'indice che si toccano (Pinching). Se le dita si toccano allora viene controllata la distanza tra le due mani:

* Se la distanza è maggiore di 0.38 unità di Unity allora viene eseguito un ZoomOut
* Se la distanza è minore di 0.2 unità di Unity allore viene eseguito un ZoomIn

Vedi metodi pubblici della classe Zoom per ZoomIn e ZoomOut.

Abbiamo in seguito creato il nuovo script "ManageRight" che si occupa di gestire i movimenti della mano destra. Usando la stessa metodologia di “ManageLeft”, ManageRight è stato assegnato alla mano destra "RigidRoundHand\_R" e tramite il controller Leap Motion viene captato l'oggetto mano. Dopo aver controllato che la mano esista nel contesto corrente e che sia effettivamente la mano destra viene salvata in una variabile la posizione della mano. In seguito viene controllato se l'indice tocca il pollice:

* Se è vero la variabile pinch viene settata a true ed inoltre viene salvata la distanza tra le due dita.
* Se è falso controlla la rotazione della mano e a dipendenza di quanto è ruotata richiama il metodo RotateRight o RotateLeft della classe Rotate. Se la mano è a pugno chiuso la rotazione torna come era inizialmente con il metodo NormalizeRotation della classe Rotate.

Questa classe ha inoltre i tre metodi pubblici: IsHandPinching, GetPinchDistance,GetPalmNormal che ritornano le tre variabili settate in precedenza.

In seguito abbiamo constatato che è meglio applicare lo Zoom e la Rotation solo attraverso i tasti della tastiera e abbiamo quindi disattivato queste funzionalità negli script “ManageLeft” e “ManageRight”.

#### InteractionManager

All'interno dell'HandModel abbiamo creato il GameObject "InteractionManager" inserendo al suo interno lo script "Interaction Manager" e lasciando le impostazioni di default, in seguito all'interno del GameObject appena creato abbiamo inserito i Prefab: “Interaction Hand Left” ed “Interaction Hand Right” entrambe queste Prefab sono trovabili nel progetto in "Assets/Plugins/LeapMotion/Modules/InteractionEngine /Prefabs/Interaction Controllers".

A questo punto è possibile modificare lo script "ShowMenu" aggiungendo come parametri in entrata "Hands" e "TelaDisegnabile", nel metodo Start dello script questi due GameObject sono stati impostati come non attivi (in questo modo non risultano visibili in contemporanea al menù iniziale).

Inoltre, si può aggiungere al GameObject “ActionController” i seguenti script:

* "StartDrawing" che serve a disattivare i menù iniziali (Start\_Page e impostazioni) e, a mostrare mani e tela quando si pigia il bottone "Continua" oppure il bottone "FoglioEsistente".
* “CreateFile”
* “GetFile”

### Lettura movimenti

La lettura dei movimenti avviene tramite metodi già presenti nel package del Leap Motion, che riconoscono se la mano è in una determinata posizione. Questo può esserci molto utile, visto che non dovremo fare altro che inserire il metodo per controllare la posizione voluta nella condizione dell’if per gestire, quando fare, o non fare, una determinata azione.

### HandReset e HandFinish

Sia "ManageLeft" che "ManageRight" devono implementare "HandTransitionBehavior", a questo punto bisogna importare i due seguenti metodi:

* "HandReset" viene richiamato all'inserimento della mano nella scena
* "HandFinish" viene richiamato all'uscita della mano dalla scena

Questi due metodi notificano se le mani vengono inserite oppure se vengono levate dalla scena.

## Disegno

### Azione Disegno

L’azione di disegnare è una se non la parte più importante del progetto. Per farlo utilizza i RayCast, ovvero dei vettori orizzontali che partono dalla mano e arrivano fino alla tela. Nel momento in cui la mano soddisfa la condizione isPinching(), ovvero ci sono una o più dita il punto toccato sulla tela viene colorato del colore selezionato. Il metodo si chiama DrawCircle e per disegnare il puntino si entra in un ciclo definito dalla grandezza di esso che cambia tutti i pixel al suo interno.

Immagine che contiene testo

Descrizione generata automaticamente

Figura - Azione di disegno

### Selezione colore

#### Tavolozza dei colori

La tavolozza dei colori è la componente che permette di definire i colori da usare per dipingere. È infatti possibile scegliere le singole componenti RGBA dei colori, e avere fino a un massimo di 12 colori.

#### Creazione della GUI

In un primo momento è stata pensata ad una GUI 2D strutturata con 3 slider in una parte superiore e una serie di pallini rappresentanti i colori nella parte inferiore.

Immagine che contiene testo, orologio, dispositivo, calibro

Descrizione generata automaticamente

Figura - 2D Color Picker

Poi, in seguito, si è optato per una versione 3D così da poter implementare facilmente l’interattività con il Leap Motion. Infatti, la versione 3D è composta da componenti presi dai progetti Unity d’esempio di utilizzo del Leap Motion. Gli slider hanno la componente InteractionSlider mentre i button hanno la InteractionButton. La struttura è stata modificata poiché ritenuta più comoda da usare con il Leap Motion. Adesso è divisa in due parti verticali dove a sinistra ci sono gli slider uno sotto all’altro e a destra i button sono disposti in una matrice 3 per 4.



Figura - 3D Color Picker

#### La classe ColorPicker

La classe ColorPicker serve a gestire le funzioni della paletta di colori ed è un suo componente.

Ogni volta che un colore (parte destra) viene selezionato, cliccandoci sopra con un dito, quello diventa il colore che verrà usato per disegnare. Il colore selezionato è messo in evidenza poiché leggermente più grande degli altri. Quando un colore selezionato gli slider prendono i valori delle componenti RGBA[[7]](#footnote-7) di quel colore. Questa serie di istruzioni vengono eseguite dal metodo OnColorSelected() che viene aggiunta come azione ai button.

button.GetComponent<InteractionButton>().OnPress = new Action(OnColorSelected);

Ad ogni cambiamento di valore degli slider viene aggiornato sia il valore numerico rappresentato a destra che il valore effettivo del colore. Per fare ciò ci sono i tre metodi che eseguono questi aggiornamenti di valori: OnChangedRed(), OnChangedGreen(), OnChangedBlue(), OnChangedAlpha(). Anche in questo caso i metodi vengono aggiunti come azione da eseguire con la modifica del valore degli slider.

SliderRed.HorizontalSlideEvent = new Action<float>(OnChangedRed);

SliderGreen.HorizontalSlideEvent = new Action<float>(OnChangedGreen);

SliderBlue.HorizontalSlideEvent = new Action<float>(OnChangedBlue);

SliderAlpha.HorizontalSlideEvent = new Action<float>(OnChangedAlpha);

### Selezione strumento e dimensione della penna

La selezione dello strumento e della dimensione della penna è eseguita tramite un pannello composto da 3 bottoni ed uno slider. Selezionando uno dei tre bottoni è possibile cambiare lo strumento (penna, gomma o filler), purtroppo abbiamo dovuto disattivare la funzione filler in quanto rallentava troppo l’esecuzione del programma.  
Selezionando la penna lo script “ManageRight” imposta il colore selezionato dal Color Picker come colore dei pixel da modificare (vedi capitolo sull’azione disegno), mentre selezionando la gomma il colore diventa bianco così da dare l’impressione di cancellare il disegno.

Attraverso lo slider invece si modifica la grandezza del punto che si andrà a disegnare (quantità di pixel).

La seguente immagine mostra il pannello, il primo bottone è il bottone penna, il secondo la gomma, mentre il terzo è il filler.

Immagine che contiene testo, parete, interni, galleria

Descrizione generata automaticamente

Figura - Pannello Selezione Strumento

### Visualizzazione dello strumento in uso

Per prima cosa bisogna creare un Canvas nella scena, questo Canvas ha “RenderMode” in modalità “MainCamera”, in questo modo pur cambiando la dimensione dello schermo lo strumento selezionato sarà sempre in alto a sinistra.

Nel canvas bisogna in seguito creare tre oggetti immagini (gomma, penna e riempi) le quali mostrano ognuna uno strumento (le immagini sono le stesse che vengono mostrate nel pannello di selezione dello strumento visto in precedenza), ogni qualvolta viene selezionato uno strumento diverso dal precedente verrà mostrato solamente l'oggetto immagine dell'oggetto in uso. In seguito, è stato aggiunto un altro oggetto “RawImage” alla quale viene cambiato il colore secondo il colore del pennello attualmente in uso.

## Azioni varie ed inventario

Come annunciato nei precedenti capitoli l’inventario consiste in un pannello di bottoni attaccati alla mano.  
Ogni bottone ha uno script dedicato, il quale ascolta quando il bottone viene premuto (“isPressed”) dalla mano destra, ogni bottone ha un diverso scopo. Per fare in modo che i bottoni funzionano bisogna assicurarsi prima che nell’Interaction Hand della mano destra sia attivo il “Contact Enabled”.

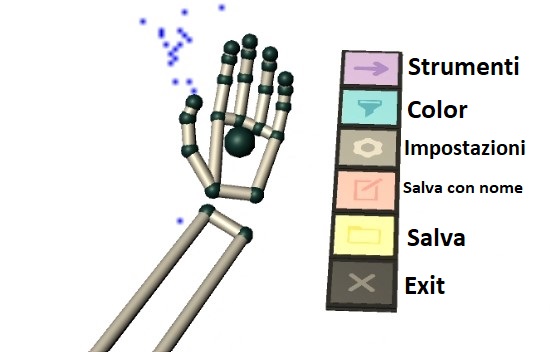


Figura - Pannello Inventario

### Selezione strumento

Selezionando il bottone per la selezione dello strumento verrà visualizzato pannello per la scelta dello strumento corrente e la scelta della grandezza della penna, vedi capitoli precedenti.  
Nel caso in cui il bottone sia nuovamente cliccato il pannello scomparirà applicando le impostazioni settate.

### ColorPicker

Selezionando il bottone per la selezione dei colori verrà visualizzato il Color Picker descritto in precedenza, se il bottone viene cliccato nuovamente il Color Picker scomparirà e verrà impostato il colore scelto.

### Exit

Cliccando questo bottone viene visualizzata la GUI di uscita tramite la quale si può scegliere se uscire dal programma e salvare oppure non salvare il disegno in corso, in alternativa si può annullare l’azione di uscita.  
Lo script "Close" è stato quindi creato ed assegnato all'ActionController. Se il bottone "si" (che indica se si vuole salvare e uscire) viene cliccato viene richiamato il metodo "salvaEdEsci" il quale salva la texture (tela) e chiude il programma. Se invece viene cliccato il bottone "no" (non salvare, ma uscire ugualmente) il programma viene chiuso. Infine, se viene cliccato il tasto "Annulla", le mani vengono nuovamente mostrate e l'UI non viene più visualizzata.



Figura - Exit GUI

L’uscita dal programma viene eseguita con “Application.Quit()”.

Non è possibile testare l'uscita dal programma (Application.Quit()) se non è ancora stata eseguita una build del programma.

### Salvataggio

Il salvataggio rapido avviene tramite il metodo salva() richiamato al momento del clic del bottone dedicato. Essendo che al momento della creazione della tela avviene già un pre-salvataggio, non è necessario controllare ed eventualmente far salvare con nome nel caso sia il primo salvataggio. Lo script funziona in modo molto semplice, per prima come controlla la path dell’ultimo file aperto nel file json così da sapere che file sovrascrivere, e successivamente prende la nuova texture dalla tela e la sovrascrive al file. Visto che alla tela possono venir assegnate texture prese da file sia jpg che png prima di salvare fa un controllo per vedere di rimanere con la stessa estensione, visto che i metodi di salvataggio per jpg e quello per png sono due metodi distinti.

### Salvataggio con nome

Il salvataggio con nome avviene appena viene premuto l’apposito bottone, prendendo per prima cosa la texture attuale dalla tela, poi apre l’explorer e ritorna la path selezionata, dopo di che controlla l’estensione e salva la texture. Una volta fatto questo va a scrivere all’ultimo index nel file json la path così da sapere dove fare un eventuale salvataggio rapido in futuro ed elimina eventuali doppioni all’interno di esso.

### Impostazioni

Il bottone impostazioni consente di mostrare nuovamente il la GUI “Configurazione Tela” tramite la quale è possibile ridimensionare il disegno.

### Ctrl + Z

Il ctrl Z o Undo funziona tramite una lista grande fino a 15 texture su cui vengono salvate e instanziate le texture dopo ogni modifica. Una volta che viene salvata la 16 la 1 viene eliminata facendo si che ce ne siano sempre 15. Al momento del ctrl Z viene assegnata alla tela la penultima texture salvata e poi questa viene eliminata dalla lista così che al prossimo ctrl z la texture presa sarà quella ancora prima.

### Riempimento

L’azione di fill, o riempimento, è stata sviluppata in modo simile all’azione disegno, infatti prevedeva il controllo tramite isPinching() della mano destra per sapere quando farlo e prendeva il punto utilizzando i RayCast per saper dove farlo.

Per sapere quando fare il riempimento e quando disegnare è stata aggiunto uno strumento selezionabile nella sezione di selezione strumento.

Il funzionamento non prevedeva altro che partire dal punto indicato e continuare a colorare in tutte le direzioni finchè non si fosse incontrato un pixel di colore diverso.

Questa azione, tuttavia, è risultata troppo pensate e abbiamo dunque deciso di rimuoverla dal progetto.

### Rotazione della tela

Per gestire la rotazione della tela è stato creato un nuovo script chiamato "Rotate". All'interno di questa tela ho messo 3 metodi pubblici che ruotano la tela cambiando la rotazione con "tela.transform.Rotate(new Vector3(0, {rotazione} ,0));"

La tela è ruotabile anche premendo i bottoni Right e Left Arrow della tastiera e la barra spaziatrice (resetta la rotazione).

Questo script è stato aggiunto all'ActionController.

### Zoom della tela

Per gestire lo Zoom della tela abbiamo creato un nuovo script chiamato "Zoom". All'interno di questa tela sono stati inseriti tre metodi pubblici che avvicinano/allontanano la tela dalla telecamera così da ottenere un effetto di zoom. Questo script è stato aggiunto all'ActionController.

### Info Panel

Abbiamo creato un pannello che mostra i comandi principali per l'utilizzo dell'applicazione, questo menù è visibile selezionando il tasto "i" e diventa invisibile utilizzando la stessa modalità.

Immagine che contiene testo

Descrizione generata automaticamente

Figura - Info Panel

# Test

## Protocollo di test

|  |  |  |  |
| --- | --- | --- | --- |
| Test Case:  Riferimento: | TC-001  REQ-001 | Nome: | Tracciare le mani |
| Descrizione: | Controllare che prenda correttamente le mani. | | |
| Prerequisiti: | Il programma Virtual Painter ed il sensore Leap Motion | | |
| Procedura: | 1. Attaccare il Leap Motion al pc e avviare il programma. 2. Mettere le mani sopra il Leap Motion e muoverle guardando se nell’applicazione si muovono correttamente. | | |
| Risultati attesi: | Le mani nell’applicazione si muovono come le mani reali. | | |

|  |  |  |  |
| --- | --- | --- | --- |
| Test Case:  Riferimento: | TC-002  REQ-002 | Nome: | Cambio strumento |
| Descrizione: | Controllare che il cambio dello strumento avvenga correttamente. | | |
| Prerequisiti: | Il programma Virtual Painter ed il sensore Leap Motion | | |
| Procedura: | 1. Disegnare una riga. 2. Aprire il pannello gestione strumenti e selezionare la gomma. 3. Cancellare la riga. | | |
| Risultati attesi: | Lo strumento selezionato passa da matita a gomma, e la linea inizialmente disegnata è cancellata. | | |

|  |  |  |  |
| --- | --- | --- | --- |
| Test Case:  Riferimento: | TC-003  REQ-003 | Nome: | Cambio colore di disegno |
| Descrizione: | Controllare che il cambio del colore avvenga correttamente. | | |
| Prerequisiti: | Il programma Virtual Painter ed il sensore Leap Motion | | |
| Procedura: | 1. Tracciare una linea con il colore di default d’avvio. 2. Aprire il menu girando il braccio sinistro. 3. Selezionare con l’indice della mano destra il pulsate per aprire la tavolozza (secondo dall’alto) 4. Cambiare le componenti di rosso, verde, blu e trasparenza trascinando gli slider. 5. Rischiacciare il pulsante del punto 3 e disegnare una linea 6. Verificare che colore selezionato e quello della linea sia lo stesso. 7. Aprire nuovamente la tavolozza come nei punti 2 e 3, e selezionare un altro slot colore a destra. Cambiare le componenti e quindi disegnare una linea. 8. Verificare che il colore sia quello nuovo. 9. Riaprire la tavolozza e selezionare il primo colore utilizzato nel punto 6. 10. Disegnare una linea e verificare che il colore sia effettivamente quello usato ne punto 6. | | |
| Risultati attesi: | Nella tavolozza vengono impostati due colori e sulla tela vengono utilizzati per disegnare. La tavolozza memorizza i colori precedenti e possono venire riutilizzati e modificati. | | |

|  |  |  |  |
| --- | --- | --- | --- |
| Test Case:  Riferimento: | TC-004  REQ-004 | Nome: | Disegno a mano libera |
| Descrizione: | Controllare la possibilità di disegnare a mano libera. | | |
| Prerequisiti: | Il programma Virtual Painter ed il sensore Leap Motion con i suoi driver e software. | | |
| Procedura: | 1. Con la mano destra unire le dita e muoverla per tracciare una linea. | | |
| Risultati attesi: | Una linea viene disegnata. | | |

|  |  |  |  |
| --- | --- | --- | --- |
| Test Case:  Riferimento: | TC-005.1  REQ-005.1 | Nome: | Salvataggio disegno |
| Descrizione: | Controllare che il disegno venga salvato correttamente | | |
| Prerequisiti: | Il programma Virtual Painter ed il sensore Leap Motion con i suoi driver e software. | | |
| Procedura: | 1. Modificare il disegno disegnandoci. 2. Aprire il menu girando il braccio sinistro 3. Selezionare con l’indice della mano destra il pulsate per salvare (quarto dall’alto) 4. Verificare che il file sia stato salvato correttamente nel percorso scelto in precedenza | | |
| Risultati attesi: | Un file JPG o PNG contenente il disegno è presente nel percorso corretto | | |

|  |  |  |  |
| --- | --- | --- | --- |
| Test Case:  Riferimento: | TC-005.2  REQ-005.2 | Nome: | Salvataggio con nome |
| Descrizione: | Controllare che il disegno venga salvato correttamente nel giusto percorso | | |
| Prerequisiti: | Il programma Virtual Painter ed il sensore Leap Motion con i suoi driver e software. | | |
| Procedura: | 1. Modificare il disegno disegnandoci. 2. Aprire il menu girando il braccio sinistro 3. Selezionare con l’indice della mano destra il pulsate per salvare con nome (quinto dall’alto) 4. Verificare che il file sia stato salvato correttamente nel percorso scelto. | | |
| Risultati attesi: | Un file JPG o PNG contenente il disegno è presente nel percorso corretto | | |

|  |  |  |  |
| --- | --- | --- | --- |
| Test Case:  Riferimento: | TC-007  REQ-007 | Nome: | Importare un’immagine |
| Descrizione: | Verificare che l’importazione di un’immagine funzioni e che sia disegnabile | | |
| Prerequisiti: | Il programma Virtual Painter ed il sensore Leap Motion con i suoi driver e software. con i suoi driver e software | | |
| Procedura: | 1. All’avvio del programma selezionare “Apri un foglio esistente” (Il pulsante a destra) 2. Selezionare un file JPG o PNG 3. Modificare l’immagine disegnandoci | | |
| Risultati attesi: | L’immagine viene caricata correttamente sul piano ed è possibile disegnarci | | |

|  |  |  |  |
| --- | --- | --- | --- |
| Test Case:  Riferimento: | TC-008  REQ-006 | Nome: | Applicativo stand alone |
| Descrizione: | Controllare la possibilità eseguire l’applicazione al di fuori dell’ambiente di sviluppo | | |
| Prerequisiti: | Una build del programma Virtual Painter ed il sensore Leap Motion con i suoi driver e software. | | |
| Procedura: | 1. Prendere la build dell’applicazione e avviarla sul pc. | | |
| Risultati attesi: | L’applicazione si avvia e funziona senza problemi. | | |

## Risultati test

Tabella riassuntiva in cui si inseriscono i test riusciti e non del prodotto finale. Se un test non riesce e viene corretto l’errore, questo dovrà risultare nel documento finale come riuscito (la procedura della correzione apparirà nel diario), altrimenti dovrà essere descritto l’errore con eventuali ipotesi di correzione.

## Mancanze/limitazioni conosciute

Descrizione con motivazione di eventuali elementi mancanti o non completamente implementati, al di fuori dei test case. Non devono essere riportati gli errori e i problemi riscontrati e poi risolti durante il progetto.

# Consuntivo

Consuntivo del tempo di lavoro effettivo e considerazioni riguardo le differenze rispetto alla pianificazione (cap 1.7) (ad esempio Gannt consuntivo).

# Conclusioni

Quali sono le implicazioni della mia soluzione? Che impatto avrà? Cambierà il mondo? È un successo importante? È solo un’aggiunta marginale o è semplicemente servita per scoprire che questo percorso è stato una perdita di tempo? I risultati ottenuti sono generali, facilmente generalizzabili o sono specifici di un caso particolare? ecc

## Sviluppi futuri

Migliorie o estensioni che possono essere sviluppate sul prodotto.

## Considerazioni personali

Cosa ho imparato in questo progetto? ecc

# Bibliografia

## Sitografia

* <https://www.ultraleap.com/> , Sito ufficiale del Leap Motion Controller, 14-01-2021

# Allegati

* Diari di lavoro
* Codici sorgente
* Progetto in Unity
* QdC

# Glossario

1. https://developer.leapmotion.com/unity [↑](#footnote-ref-1)
2. Windows -> Package Manager [↑](#footnote-ref-2)
3. Cartella->Assets/Plugins/LeapMotion/Core/Example [↑](#footnote-ref-3)
4. Cartella->Assets/Plugin/LeapMotion/Core/Prefab [↑](#footnote-ref-4)
5. Cartella->Assets/Plugin/LeapMotion/Core/Prefab/HandModelPhysical [↑](#footnote-ref-5)
6. Cartella->Assets/Plugin/LeapMotion/Core/Prefab/HandModelNonHuman [↑](#footnote-ref-6)
7. RGBA **R**ed **G**reen **B**lue **A**lpha [↑](#footnote-ref-7)