



## **POLITECNICO DI BARI**

**DIPARTIMENTO DI INGEGNERIA ELETTRICA E  
DELL'INFORMAZIONE**

**CORSO DI LAUREA TRIENNALE IN  
INGEGNERIA INFORMATICA E DELL'AUTOMAZIONE**

**TESI DI LAUREA  
IN  
ALGORITMI E STRUTTURE DATI IN JAVA**

**CREAZIONE DI UN AVATAR PER INTERAZIONE VOCALE  
TRAMITE CHATBOT**

***Relatore:***

***Prof. Ing. Tommaso DI NOIA***

***Correlatore:***

***Ing. Antonio STELLA***

***Laureando:***

***Gabriele COLAPINTO***

***ANNO ACCADEMICO: 2022 – 2023***



**LIBERATORIA ALLA CONSULTAZIONE DELLA TESI DI LAUREA DI CUI ALL'ART.4  
DEL REGOLAMENTO DI ATENEO PER LA CONSULTAZIONE DELLE TESI DI  
LAUREA (D.R. n. 479 del 14/11/2016).**

Il sottoscritto Colapinto Gabriele matricola 571861

Corso di Laurea Ingegneria informatica e dell'automazione (D.M.270/04)

autore della presente tesi di Laurea dal titolo Creazione di un avatar per interazione vocale tramite chatbot

Parola chiave: Algoritmi e strutture dati

Abstract

Lo scopo del progetto di tesi è la realizzazione di un programma per la creazione di un punto informativo con cui gli utenti possono interagire con la voce.

Autorizza

Non autorizza

la consultazione della presente tesi, fatto divieto a chiunque di riprodurre in tutto o in parte quanto in essa contenuto.

Bari, 20/09/2022

Firma

A handwritten signature in blue ink that reads "Gabriele Colapinto".

# Indice

## Sommario

<b>DESCRIZIONE DEL PROGETTO .....</b>	<b>1</b>
<b>TECNOLOGIE UTILIZZATE .....</b>	<b>1</b>
RASA .....	1
UNITY .....	2
BLENDER .....	2
PYTHON .....	2
GIMP .....	3
REGISTRO DI SISTEMA DI WINDOWS.....	3
INSTALLFORGE .....	4
<b>REALIZZAZIONE DEL MODELLO 3D .....</b>	<b>1</b>
<b>MODIFICA DELLA GIF AUDIO .....</b>	<b>17</b>
<b>GENERAZIONE DELLA MAPPA UV .....</b>	<b>21</b>
<b>CREAZIONE DELLE TEXTURE .....</b>	<b>28</b>
<b>REALIZZAZIONE DELLE SCRITTE 3D .....</b>	<b>32</b>
<b>REALIZZAZIONE INTERFACCIA .....</b>	<b>39</b>
<b>REALIZZAZIONE DELLE ANIMAZIONI .....</b>	<b>62</b>
<b>REALIZZAZIONE DELL'ICONA .....</b>	<b>69</b>

<b>CONFIGURAZIONE DI RASA .....</b>	<b>72</b>
SPIEGAZIONE DEL FUNZIONAMENTO DI RASA.....	72
INSTALLAZIONE DI RASA.....	74
FILE “CONFIG.YML” .....	75
FILE “DOMAIN.YML” .....	76
FILE “ENDPOINTS.YML” .....	77
FILE “RULES.YML” .....	78
FILE “NLU.YML” .....	79
FILE “STORIES.YML” .....	83
FILE “TEST_STORIES.YML” .....	84
FILE “ACTIONS.PY”.....	86
GERAZIONE DEL MODELLO .....	87
PROVA DEL MODELLO .....	88
<b>ATTIVAZIONE DELLA FUNZIONE SPEECH-TO-TEXT .....</b>	<b>89</b>
ATTIVAZIONE DI UNA VOCE PER LA FUNZIONALITÀ TEXT-TO-SPEECH .....	91
<b>CODICE.....</b>	<b>95</b>
STATI DELL’APPLICAZIONE.....	95
SCHEMA DI FUNZIONAMENTO DELL’APPLICAZIONE.....	98
FILE “TTS_ENGINE.PY” .....	100

FILE “NETWORKMANAGER.CS” .....	104
FILE “TEXTSCRIPT.CS” .....	113
<b>ESPORTAZIONE DELL’APPLICAZIONE IN UNITY .....</b>	<b>117</b>
<b>CARTELLA DELL’INSTALLER .....</b>	<b>121</b>
Sviluppo del file Configurazione.bat .....	123
Sviluppo dei file per la modifica del registro di sistema .....	126
<b>REALIZZAZIONE DELL’INSTALLER .....</b>	<b>128</b>
Sezione “General” .....	128
Sezione “Setup” .....	130
Sezione “Dialogs” .....	135
Sezione “System” .....	136
Sezione “Build” .....	137
<b>INSTALLAZIONE .....</b>	<b>140</b>
<b>RISULTATI OTTENUTI.....</b>	<b>149</b>
<b>SVILUPPI FUTURI.....</b>	<b>152</b>
<b>SITOGRAFIA .....</b>	<b>154</b>
RASA .....	154
UNITY .....	155
CODICE.....	157

BLENDER .....	159
PYTHON .....	160
REALIZZAZIONE DELLE IMMAGINI DA APPLICARE ALLO SCHERMO .....	161
REGISTRO DI SISTEMA.....	161
REALIZZAZIONE DELL'INSTALLER.....	162
ALTRI TENTATIVI .....	163



# **Introduzione**

## **Descrizione del progetto**

Lo scopo del progetto è la realizzazione di un programma per la creazione di un punto informativo con cui gli utenti possono interagire con la voce.

Le informazioni che può fornire la versione iniziale del progetto riguardano la posizione delle aule e le istruzioni per connettersi ad Eduroam.

## **Tecnologie utilizzate**

### **Rasa**

Rasa è un assistente contestuale, cioè un’evoluzione di un chatbot. Al contrario di un comune chatbot Rasa è in grado di analizzare il contesto in cui avviene una conversazione e ricavarne informazioni come farebbe un umano. Inoltre, è in grado di utilizzare l’intelligenza artificiale per consentire all’utente di non conversare in maniera predeterminata.<sup>1</sup>

---

<sup>1</sup> Sito ufficiale di Rasa, <https://rasa.com/>

## **Unity**

Unity è un motore grafico multipiattaforma sviluppato da Unity Technologies che consente lo sviluppo di videogiochi e altri contenuti interattivi, quali visualizzazioni architettoniche o animazioni 3D in tempo reale.<sup>2</sup>

## **Blender**

Blender è un software libero e multipiattaforma di modellazione, rigging, animazione, montaggio video, composizione, rendering e texturing di immagini tridimensionali e bidimensionali. Dispone inoltre di funzionalità per mappature UV, simulazioni di fluidi, di rivestimenti, di particelle, altre simulazioni non lineari e creazione di applicazioni/giochi 3D.<sup>3</sup>

## **Python**

Python è un linguaggio di programmazione di "alto livello", orientato agli oggetti, adatto, tra gli altri usi, a sviluppare applicazioni distribuite, scripting, computazione numerica e system testing.

Python consente la creazione di un ambiente virtuale in cui installare i moduli necessari alle singole applicazioni. Questa best practice serve a

---

<sup>2</sup> Unity (motore grafico), [https://it.wikipedia.org/wiki/Unity\\_\(motore\\_grafico\)](https://it.wikipedia.org/wiki/Unity_(motore_grafico))

<sup>3</sup> Blender (programma), [https://it.wikipedia.org/wiki/Blender\\_\(programma\)](https://it.wikipedia.org/wiki/Blender_(programma))

fare sì che i moduli richiesti da un'applicazione non interferiscano con i moduli richiesti da un'altra.<sup>4</sup>

## Gimp

GIMP (GNU Image Manipulation Program) è un software libero multipiattaforma per l'elaborazione digitale delle immagini.

Fra i vari usi possibili vi sono fotoritocco, fotomontaggio, conversioni tra molteplici formati di file, animazioni (ad esempio in formato GIF), e processamento in batch in linea.<sup>5</sup>

## Registro di sistema di Windows

Il registro di sistema di Windows è un database gerarchico che memorizza impostazioni di basso livello del sistema operativo e delle applicazioni che scelgono di utilizzarlo.<sup>6</sup>

---

<sup>4</sup> Python, <https://it.wikipedia.org/wiki/Python>

<sup>5</sup> GIMP, <https://it.wikipedia.org/wiki/GIMP>

<sup>6</sup> Registro di sistema, [https://it.wikipedia.org/wiki/Registro\\_di\\_sistema](https://it.wikipedia.org/wiki/Registro_di_sistema)

## **InstallForge**

InstallForge è un software gratuito per la creazione di installer. Il suo utilizzo si basa sulla compilazione di una serie di schede e questo lo rende accessibile ed efficiente.<sup>7</sup>

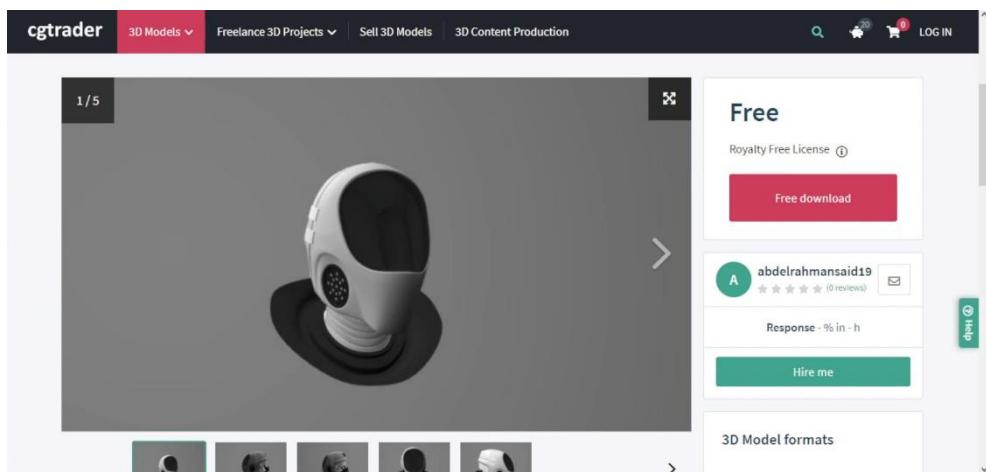
---

<sup>7</sup> InstallForge, <https://installforge.net/>

# Svolgimento

## Realizzazione del modello 3D

Per la realizzazione del modello 3D sono partito da un file scaricato da internet.<sup>8</sup>

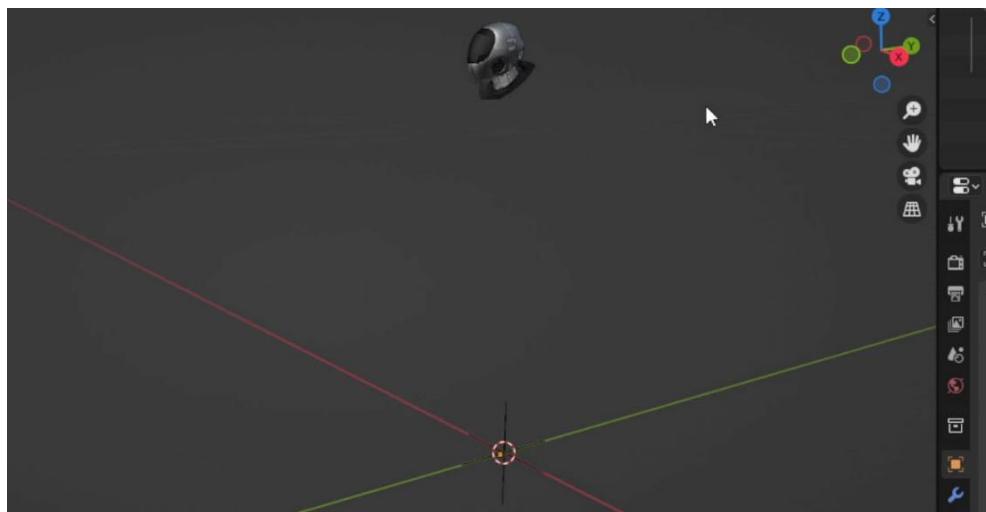


Ho importato questo file in Blender per adattarlo alle mie esigenze. Sin da subito ho notato che la testa è al di sopra di alcuni segmenti che si intersecano in un punto che il modello 3D usa come riferimento per le coordinate. Dato che esportare un modello siffatto comporterebbe il problema di gestire un oggetto poco compatto mi rendo conto di dover avvicinare la testa del robot al suo riferimento però lo farò alla fine

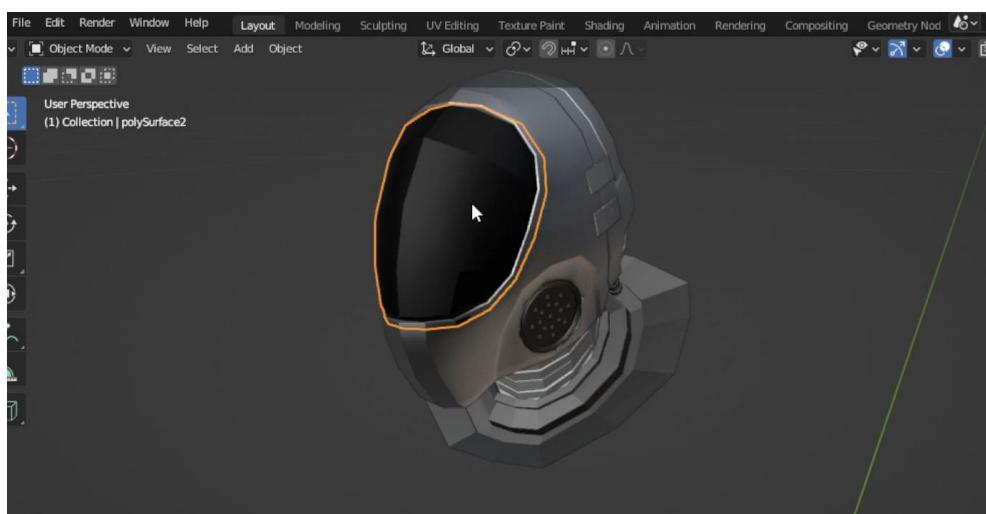
---

<sup>8</sup> CGTrader, Robot head Free 3D model, <https://www.cgtrader.com/free-3d-models/character/sci-fi-character/robot-head-5051d388-5f77-41c0-a5d1-14e9e96af41d>

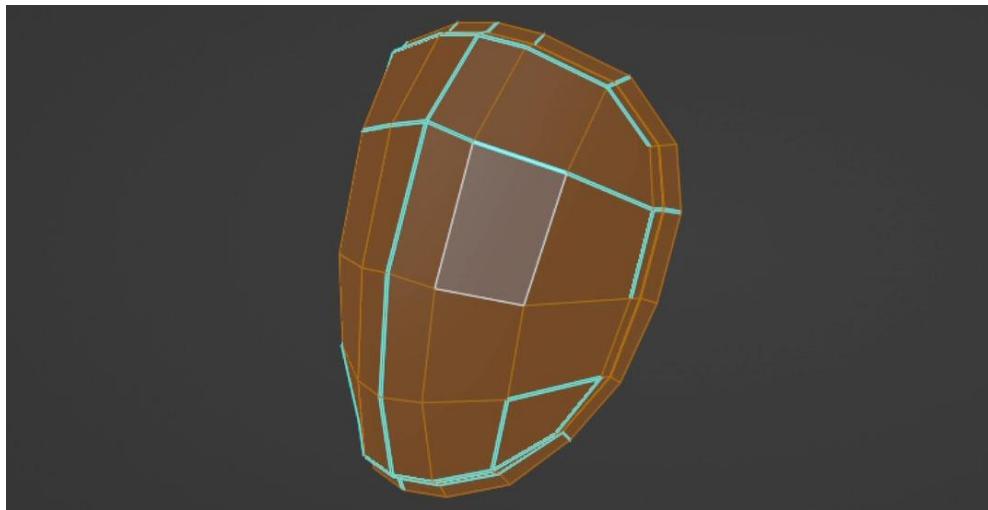
perché se lo facessi all'inizio potrei selezionare uno o più segmenti accidentalmente durante il lavoro.



La prima modifica da fare riguarda lo schermo. Osservandolo da vicino si notano degli spigoli vivi. Se un'immagine venisse applicata allo schermo questi produrrebbero un effetto sgradevole, quindi vanno rimossi.



Per assicurarmi di lavorare solo sullo schermo lo seleziono cliccandoci sopra e lo isolo premendo SHIFT+H (H sta per hide, nasconde tutti gli oggetti tranne quello selezionato). A questo punto passo alla modalità di editing (Edit Mode) tramite il menù a tendina in alto a sinistra.



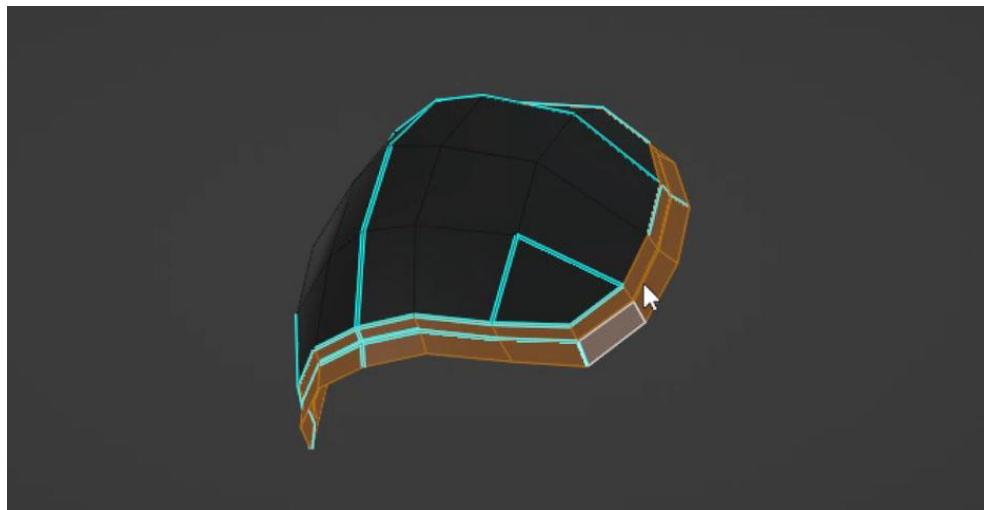
Nella modalità di editing si nota che alcuni spigoli sono evidenziati in azzurro. Questi sono contrassegnati come spigoli vivi (sharp edges), cioè spigoli non appianabili che verranno esportati così come sono indipendentemente dai modificatori applicati all'oggetto.

Dato che voglio rimuovere gli spigoli vivi solo dalla parte centrale dello schermo e voglio tenere il suo contorno intatto mi conviene separare queste due parti per rendere il lavoro più comodo.

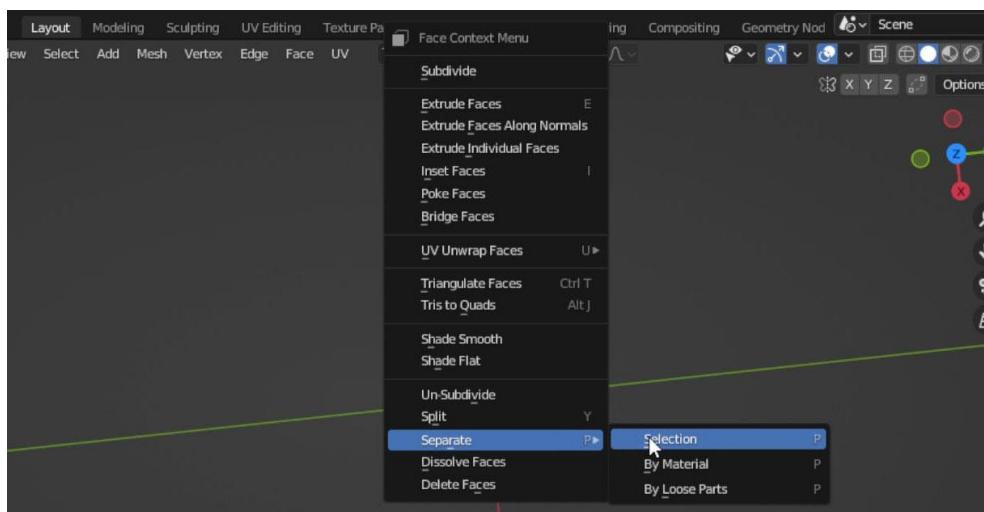
Passo alla modalità di selezione delle facce e seleziono il contorno dello schermo selezionando una faccia ed usando CTRL+CLICK su un'altra

faccia non adiacente alla prima per selezionare tutte le facce comprese tra quelle già selezionate e l'ultima selezionata.

Selezzionate tutte le facce del contorno il risultato è il seguente.

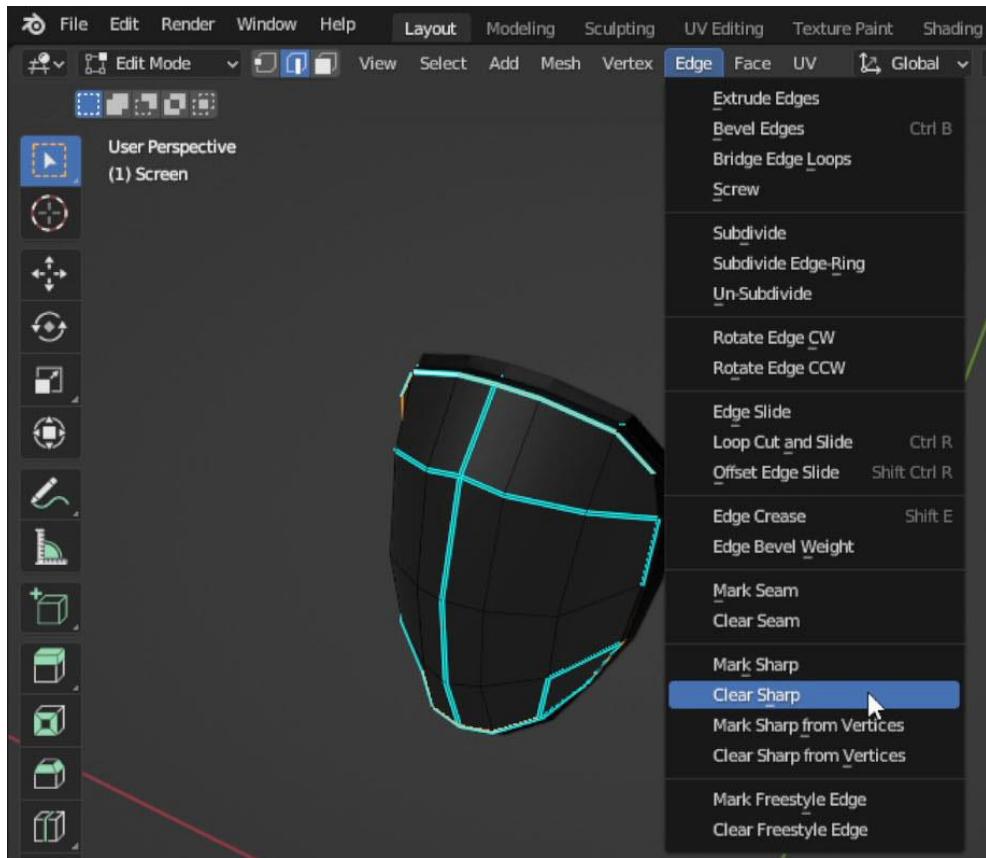


A questo punto posso separare le due parti usando l'apposito comando.



Ora che ho separato la parte centrale dello schermo dal suo contorno rinomino i due oggetti rispettivamente in “Screen” e “Screen container” (Di seguito il contorno dello schermo verrà rinominato “Screen recipient”). Il nome assegnato al contorno dello schermo può sembrare inappropriato perché non è un contenitore però in seguito renderò il centro dello schermo un oggetto figlio del suo contorno, quindi da un punto di vista gerarchico il contorno dello schermo contiene il suo centro.

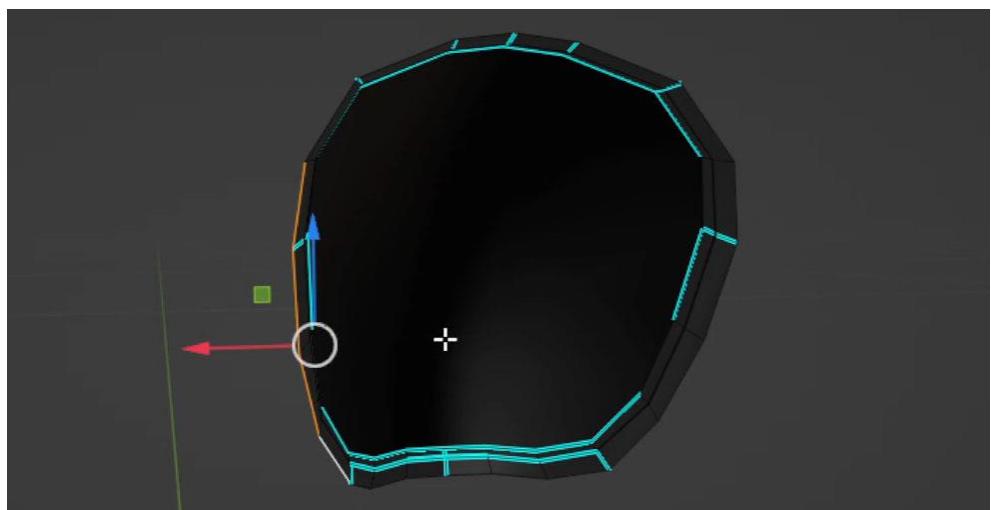
Ora che ho separato le parti dello schermo posso selezionarne il centro e rimuovere gli spigoli vivi. Per farlo uso la modalità di selezione degli spigoli, seleziono gli spigoli interessati ed uso il comando “Clear sharp” per renderli normali.



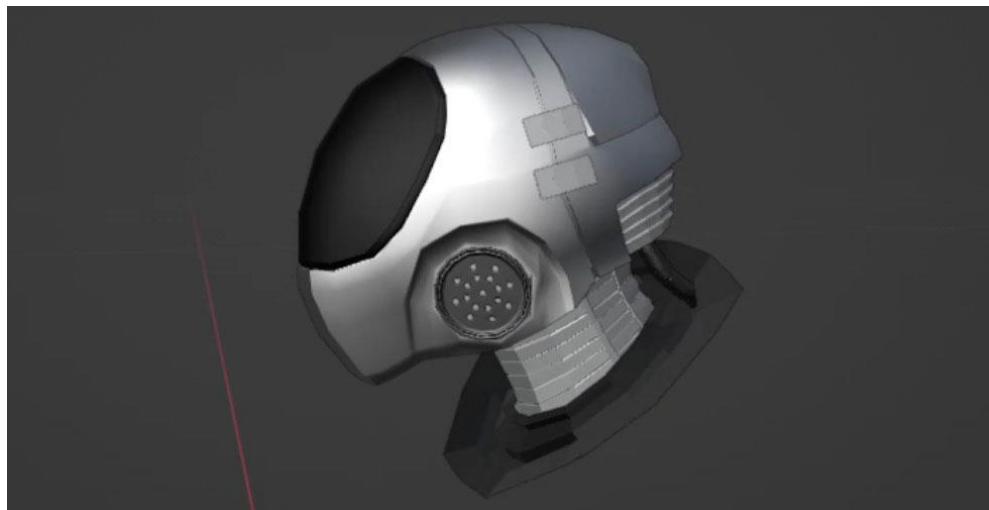
Dopo aver sistemato lo schermo rendo visibile il resto degli oggetti e noto che parte del contorno dello schermo esce dalla scocca della testa.



Per risolvere il problema seleziono gli spigoli interessati e li sposto verso l'interno.



Rendendo visibili il resto degli oggetti noto che il problema è stato risolto.

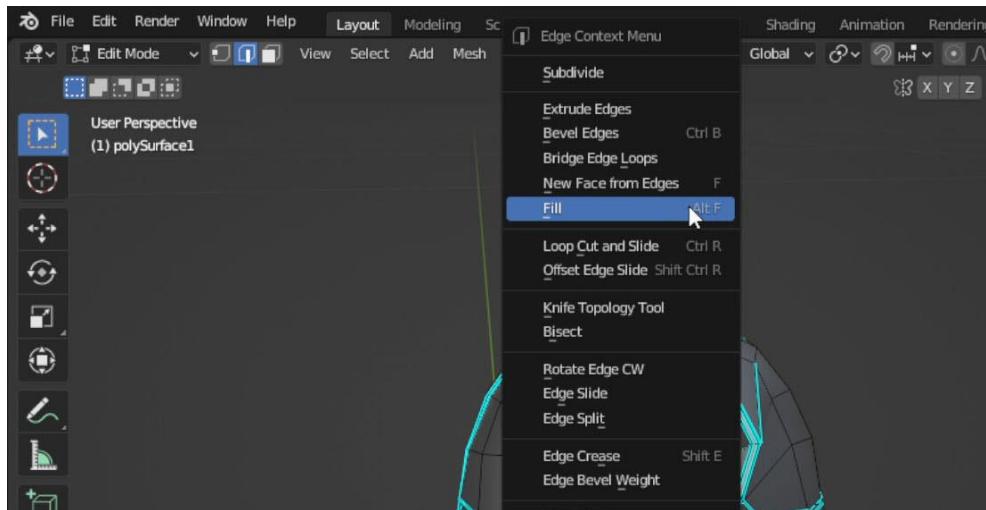


Se si osserva il retro della testa si nota una specie di tappo. Sotto quel tappo la scocca della testa è aperta, quindi guardandola da dietro si noterebbero delle parti vuote, perciò devo inserire una faccia nella scocca per chiudere il foro.

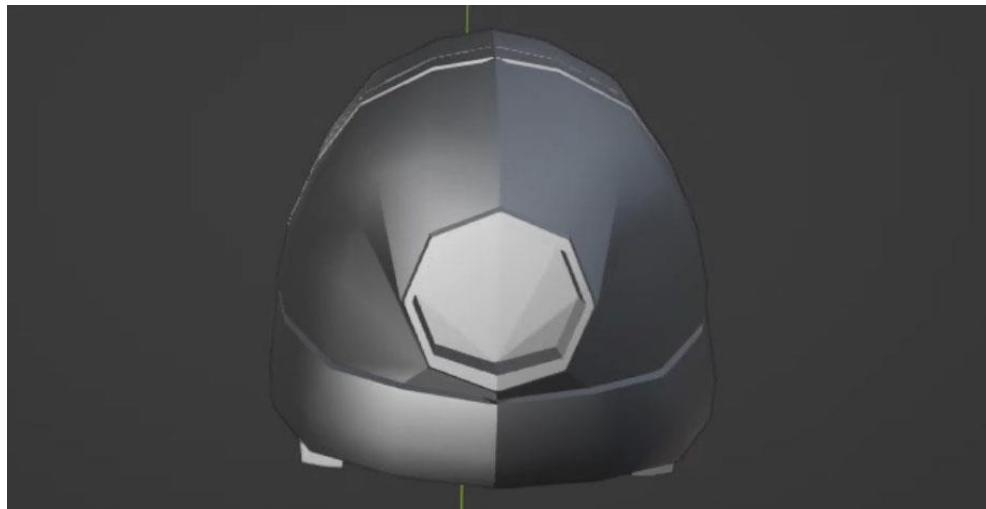


Per chiudere il foro devo isolare la scocca selezionandola e premendo SHIFT+H, devo passare in edit mode in modalità di selezione degli

spigoli, devo selezionare gli spigoli interni al foro sottostanti il tappo e devo inserire una faccia al loro interno usando il comando “Fill”.



A seguito del riempimento del foro ottengo il seguente risultato.

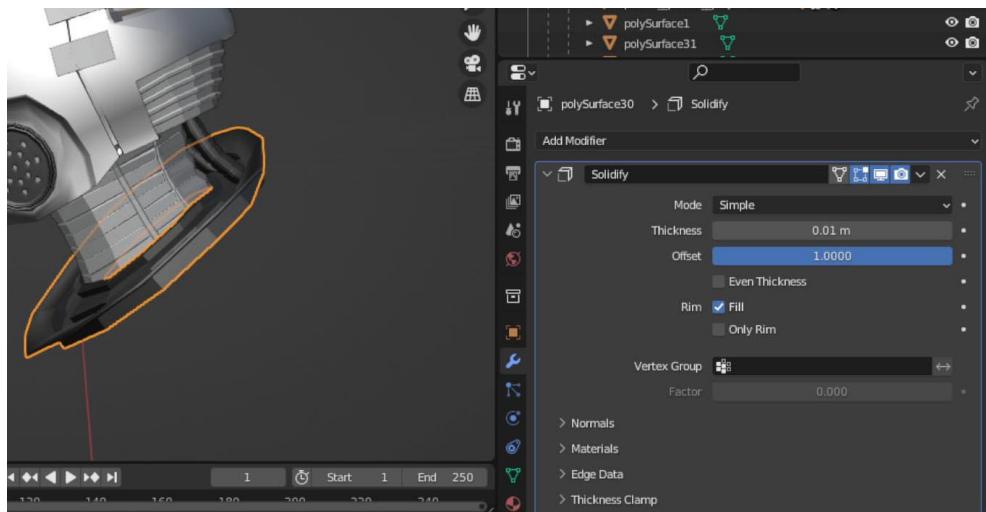


Adesso che ho adeguato lo schermo e la scocca alle mie esigenze devo considerare che il modello verrà inserito in Unity e verrà visto da più punti di vista.

Può capitare che importando un modello 3D in Unity alcune parti di questo diventino invisibili se le si guarda da una prospettiva in particolare. Questo è un problema di orientazione delle normali.

In Blender le normali sono dei vettori perpendicolari ai singoli punti che compongono una superficie (come in fisica) e servono ad indicare la direzione in cui l'oggetto riflette la luce che riceve così da poter essere visto da un osservatore.

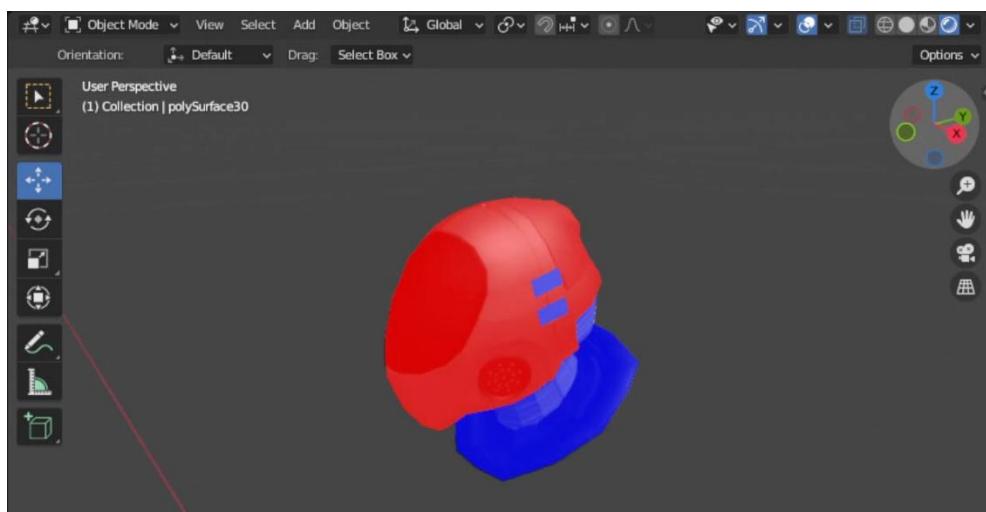
Se esportassi il modello 3D in questo stato e lo osservassi in Unity noterei che la base del collo non si vede osservandola da dietro perché è una maglia (mesh) con spessore infinitesimo e normali orientate nel verso della testa. Dato che la maglia ha uno spessore infinitesimo, se orientassi le normali nel verso opposto otterrei lo stesso problema ma da un punto di vista diverso, perciò devo dare uno spessore alla base del collo usando un modificatore chiamato “Solidify”.



All'interno del menù del modificatore tra i tanti parametri ce ne sono due che si modificano la maggior parte delle volte: Thickness ed Offset.

Thickness è semplicemente lo spessore che si vuole dare alla maglia ed Offset è lo scostamento, cioè il modo in cui questo spessore viene assegnato. Gli scostamenti agli estremi, cioè 1 e -1 danno spessore alla maglia esclusivamente da uno dei suoi lati, mentre gli scostamenti intermedi ripartiscono lo spessore tra i lati interno ed esterno.

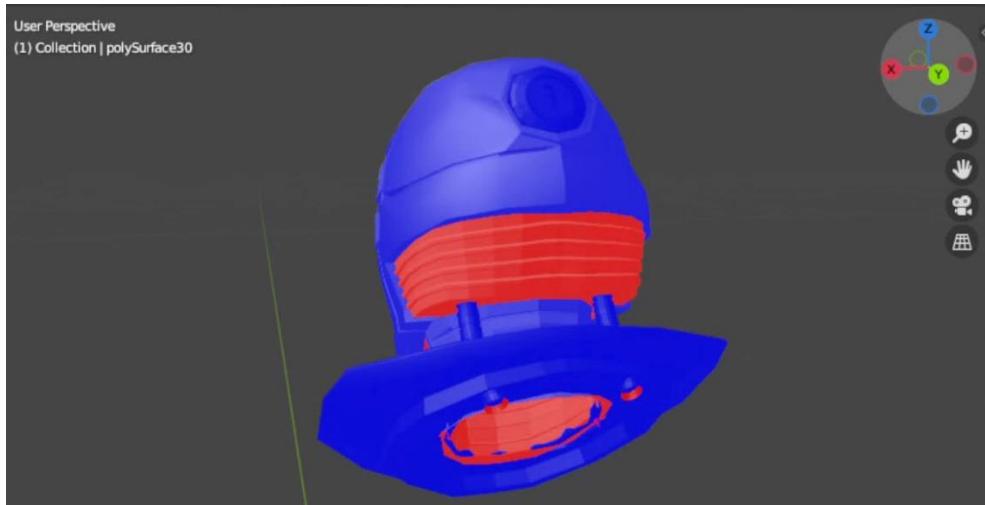
Per potermi occupare delle normali devo passare alla modalità di visualizzazione “Display render preview” e devo utilizzare l'overlay “Face orientation”.



Adesso la testa del robot è diventata rossa e blu. Il colore blu vuol dire che le normali sono orientate correttamente per la visualizzazione dell'oggetto dalla prospettiva corrente (Viewport), il colore rosso vuol dire il contrario.

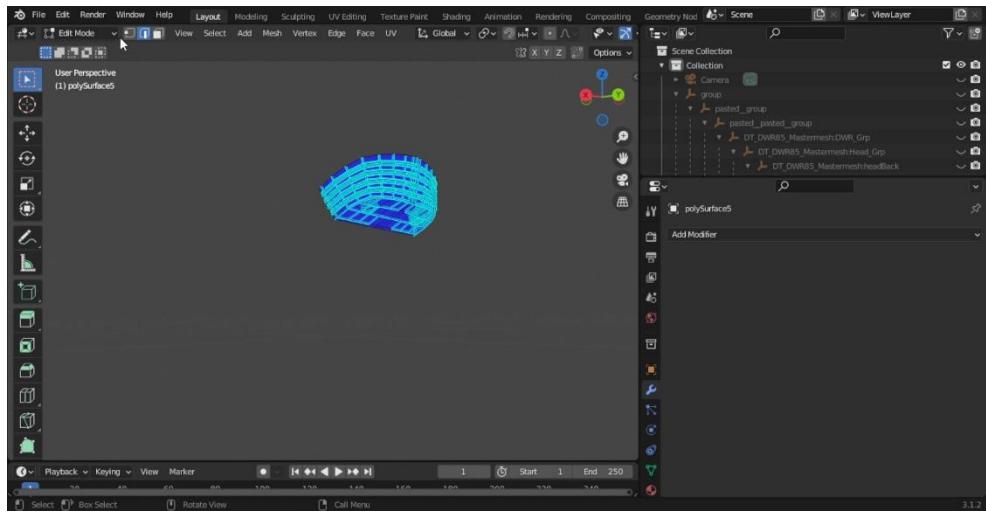
L’obiettivo di questa fase del lavoro è visualizzare la testa del robot da tutti i punti di vista interessati e vedere che ogni sua parte è blu.

Per prima cosa in questi casi bisogna passare in edit mode, selezionare tutti gli oggetti, premere CTRL+N e selezionare l’opzione “Recalculate outside” per ricalcolare le normali del lato esterno di tutti gli oggetti. Questo dovrebbe dare l’orientazione corretta alla maggior parte delle parti del modello 3D e ridurre il lavoro dell’utente alla considerazione di pochi oggetti.

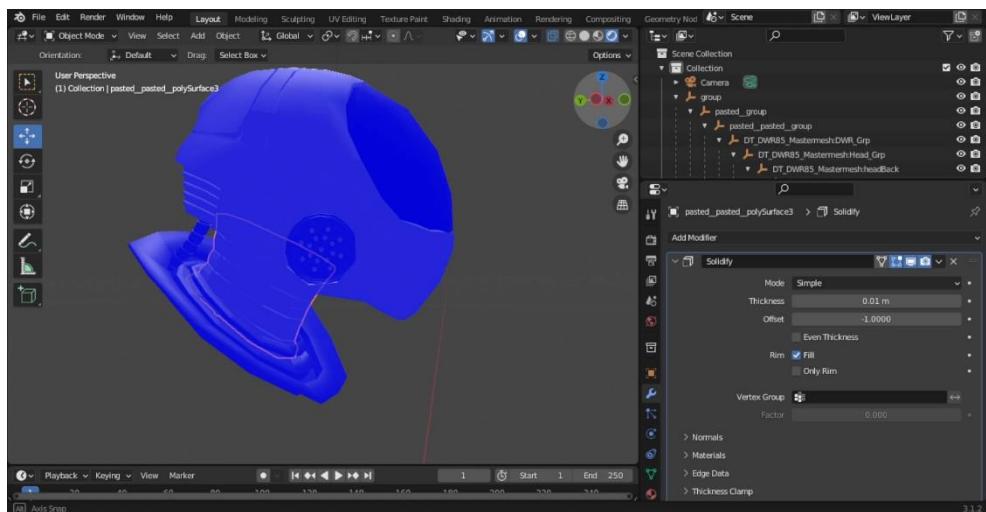


A seguito del ricalcolo generale le parti con le normali non orientate correttamente sono la nuca e l’interno del collo.

Per quanto riguarda la nuca basta ricalcolare le normali all’interno perché la faccia visibile all’esterno conta come faccia interna della maglia.

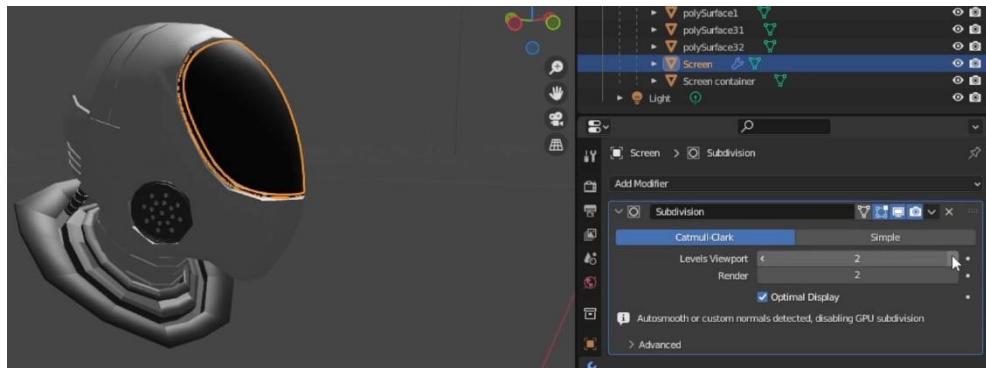


Per quanto riguarda il collo, anche questo è una maglia di spessore infinitesimo, perciò devo dargli uno spessore come fatto con la sua base.



Dopo essermi occupato delle normali devo rendere lo schermo più liscio per poterci applicare un immagine ottenendo un effetto gradevole.

Per farlo uso un altro modificatore chiamato “Subdivision” (All’interno del menù di selezione si chiama “Subdivision surface”).

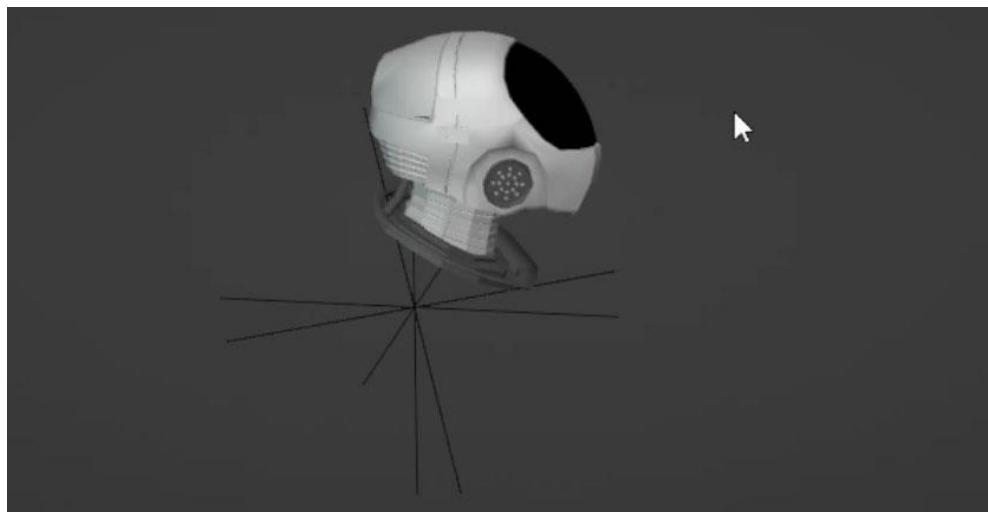


Questo modificatore suddivide le facce di una superficie in più facce e le usa per formare una superficie liscia. I livelli che si possono impostare nel menù sono il numero di volte in cui vengono eseguite queste operazioni. Ci sono due tipi di livelli, uno per la viewport e l’altro per il rendering. I livelli del rendering sono quelli che verranno effettivamente utilizzati durante l’esportazione del progetto e quelli della viewport sono quelli che vengono visualizzati dall’utilizzatore di Blender. Questi ultimi servono a dare all’utente la possibilità di valutare il numero di livelli da usare per il rendering ed anche per dargli la possibilità di ridurre i livelli utilizzati nella viewport per rendere il programma più performante dato che un elevato numero di livelli comporta un’elevata quantità di calcoli.

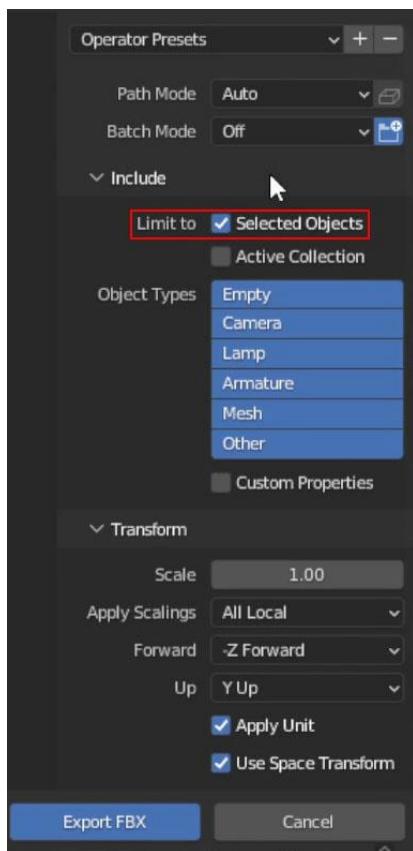
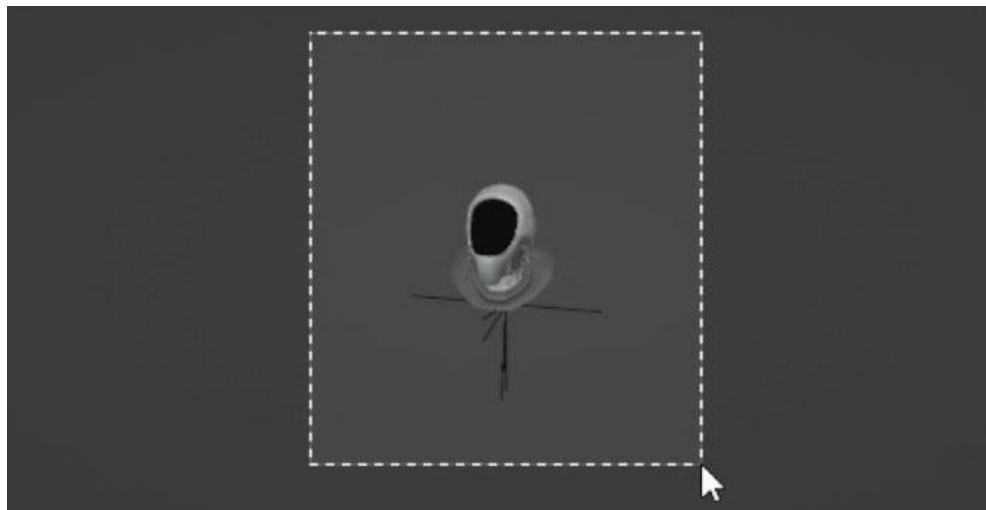
A questo punto si potrebbe pensare di usare un numero molto alto di livelli di rendering per ottenere il migliore effetto possibile. Questo è sbagliato perché non si tiene conto del fatto che questa operazione appesantirebbe il modello esportato, inoltre se si aumentano i livelli della viewport gradualmente si noterà che il miglioramento marginale del risultato diminuisce all’aumentare del numero dei livelli, quindi lo

scopo dell'utilizzatore di Blender è quello di trovare il numero minimo di livelli tale da ottenere un risultato soddisfacente. Generalmente questo numero è piccolo, infatti io ho usato 2 livelli.

A questo punto la testa del robot va solo avvicinata ai segmenti usati per il riferimento per renderla più facile da gestire in Unity. Anche se i segmenti per il riferimento si vedono in Blender questi non si vedranno nel modello esportato.



A questo punto il robot è pronto per essere esportato. Durante questo processo però non voglio esportare accidentalmente altri oggetti come la telecamera o la luce che Blender inserisce di default, quindi seleziono tutti gli oggetti che mi servono così da poter esportare solo quelli in seguito.



Per procedere con l'esportazione bisogna andare in File > Export > FBX. Nel menù a destra della finestra dell'esportazione devo selezionare la casella "Limit to Selected Objects" così da esportare solo gli oggetti desiderati e poi posso procedere con l'esportazione.

## **Modifica della gif audio**

Per la realizzazione della gif audio da inserire nello schermo del robot sono partito da una gif esistente che ho scaricato da internet.<sup>9</sup> Prima di scaricare questa gif ho consultato i termini di servizio del sito in cui c'è scritto che è possibile scaricare e modificare il materiale purché non si infrangano i termini di servizio dell'autore della gif e se l'autore non è specificato, come in questo caso, la gif è da considerarsi di pubblico dominio.<sup>10</sup>

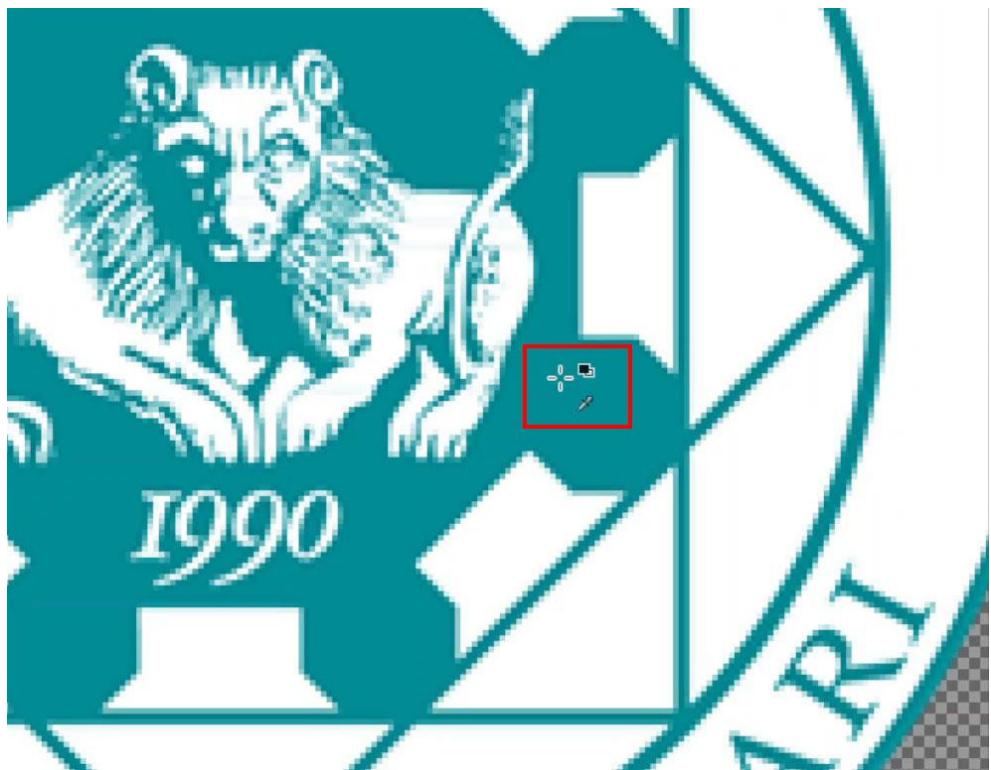
Dopo aver scaricato la gif ho cercato il logo del politecnico su Google e l'ho scaricato per avere a disposizione l'esatta tonalità di azzurro da utilizzare per la gif.

A questo punto ho aperto entrambi i file scaricati in Gimp e sono andato nella scheda contenente il logo del politecnico per prelevare la sua tonalità di azzurro usando lo strumento “Prelievo colore”.

---

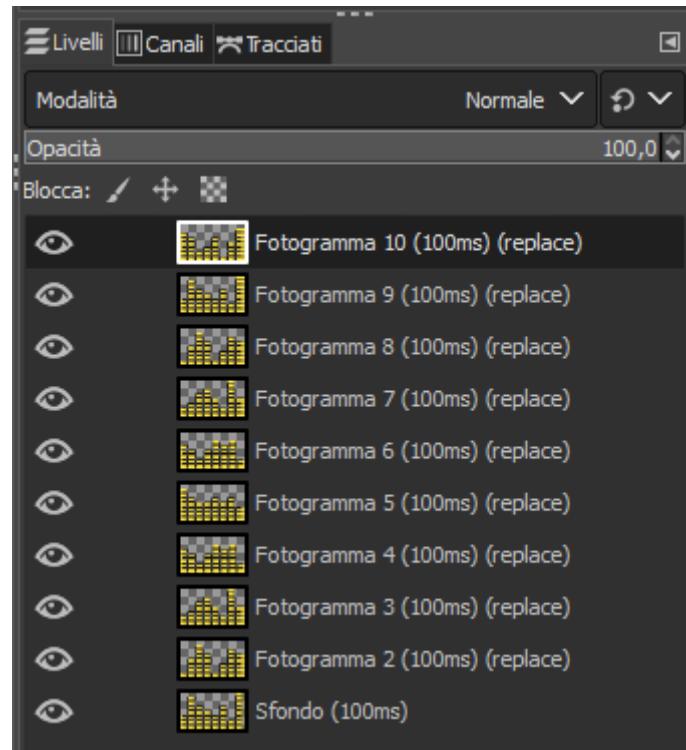
<sup>9</sup> GIFER, <https://gifer.com/en/Z23b>

<sup>10</sup> GIFER, Terms of service, <https://gifer.com/en/p/tos>

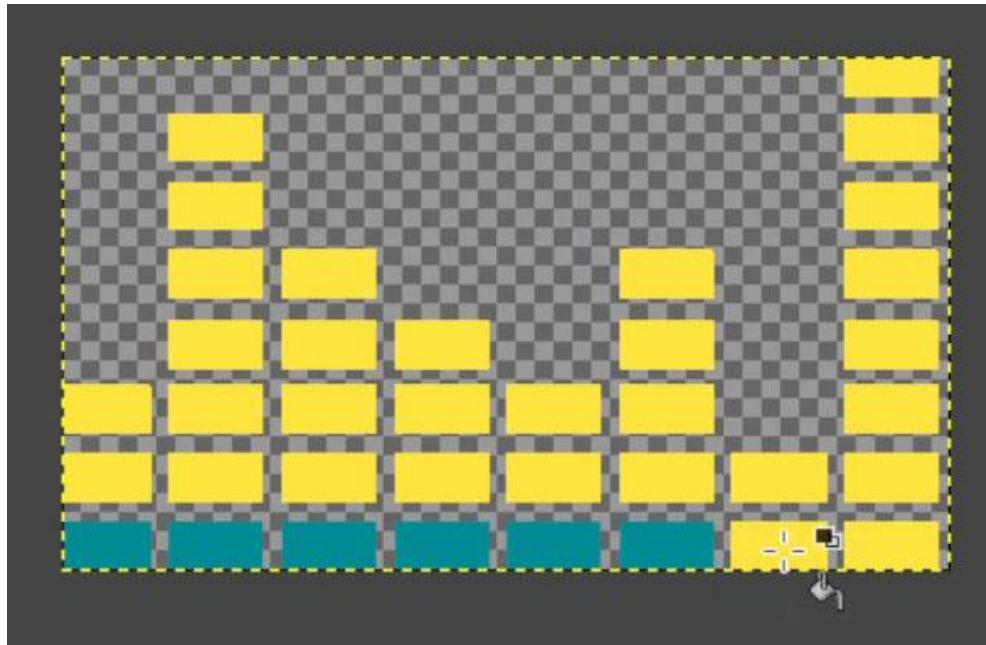


Ottenuto il colore sono passato alla scheda contenente la gif.

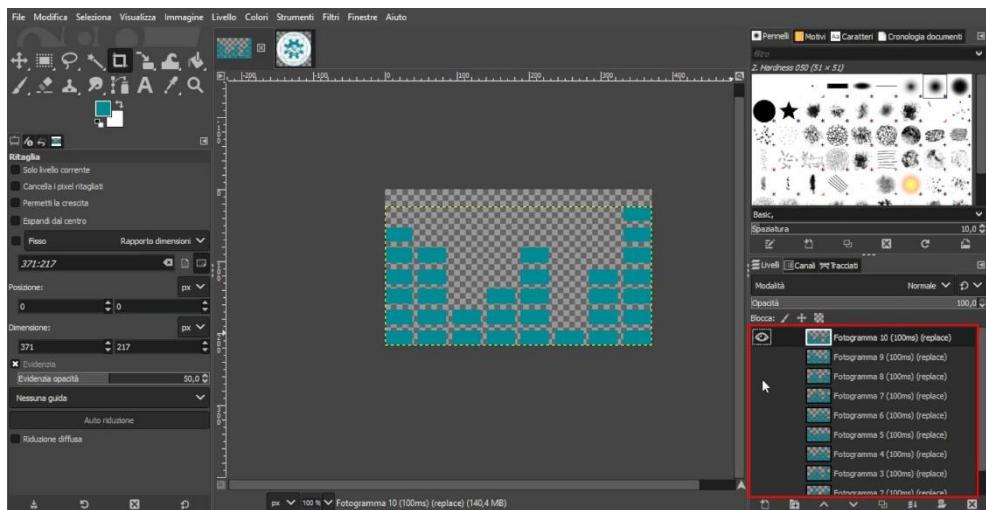
Le gif sono sequenze di immagini e Gimp le interpreta come tali. Infatti, nella scheda contenente la gif si nota che è presente un livello per ogni fotogramma.



Utilizzando lo strumento “Riempimento di colore” ho riempito i singoli rettangoli contenuti in ogni fotogramma.



Ad operazione completata posso rendere visibile un livello alla volta ed esportarlo.

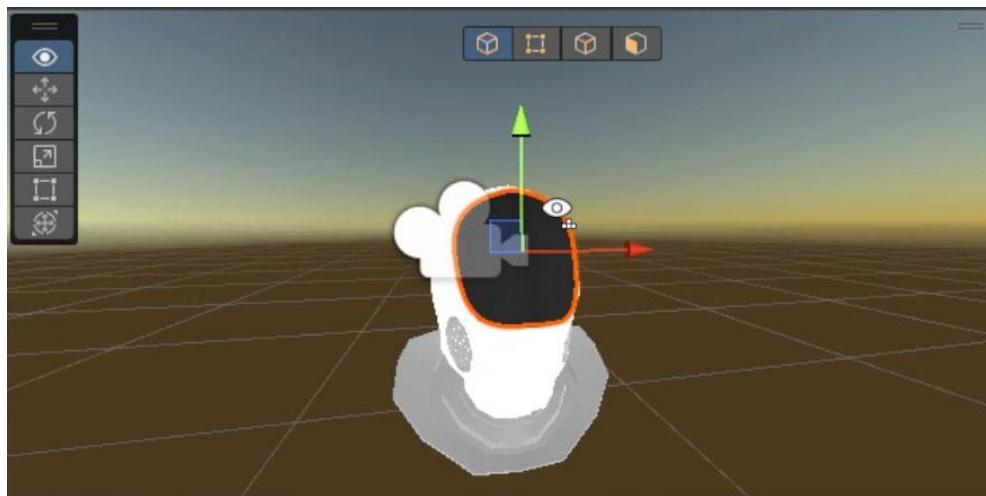


Terminata l'esportazione dei fotogrammi ho le immagini che mi servono per la realizzazione delle texture da applicare allo schermo.

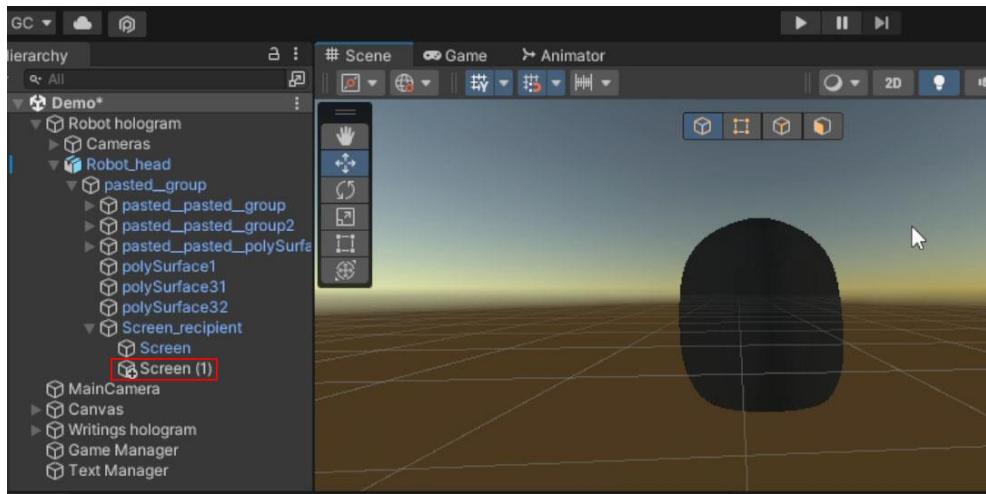
## Generazione della mappa UV

Una mappa UV è un'immagine che serve ad indicare ad un programma come applicare le texture ad un modello 3D.

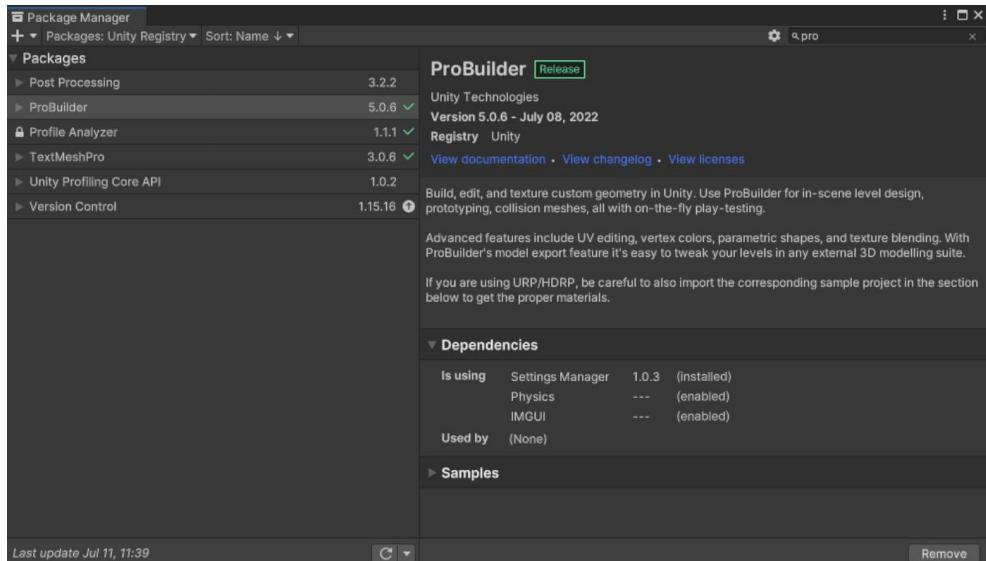
Per ricavare la mappa UV dello schermo del robot importo il modello 3D in Unity e seleziona l'oggetto in questione.



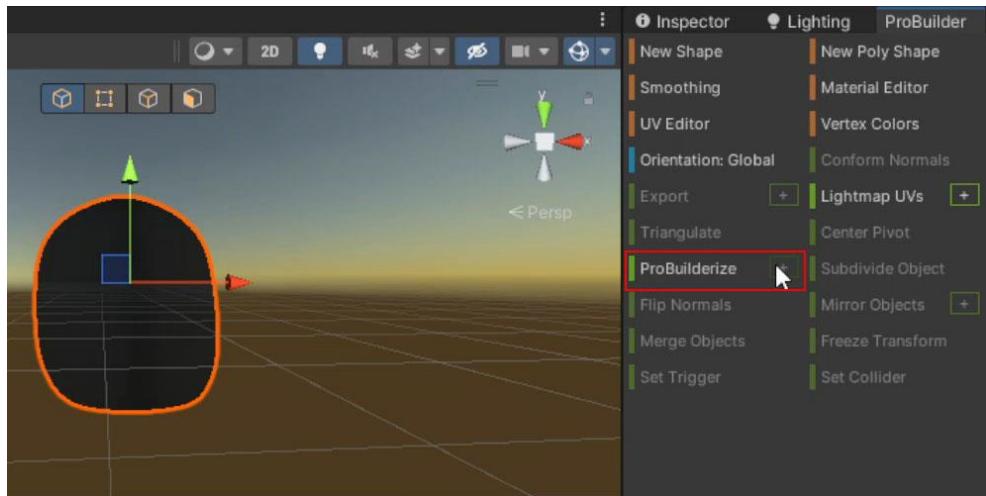
Dato che le operazioni necessarie a ricavare la mappa 3D potrebbero alterare lo schermo ne creo una copia e lo sposto in disparte così da poterci lavorare senza interferire con gli altri oggetti accidentalmente.



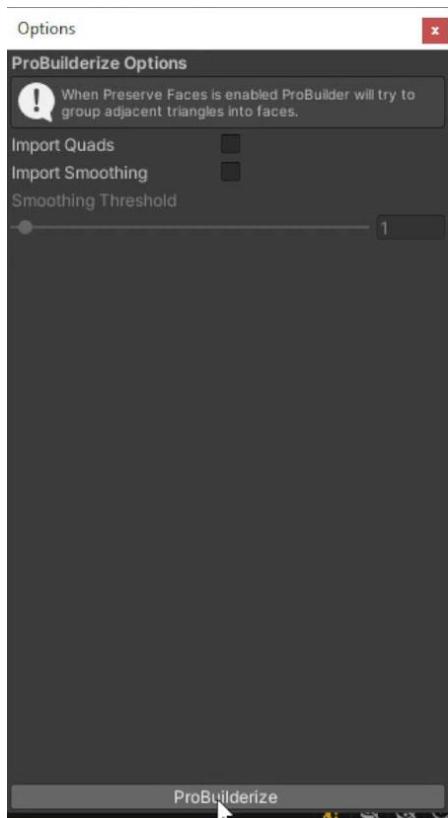
Per ricavare la mappa UV dallo schermo bisogna usare un'estensione di Unity chiamata “Probuilder”. Per scaricarla bisogna andare in Window > Package Manager e cercarla tra i pacchetti presenti nel registro di Unity.



Dopo aver installato Probuilder all'interno dell'apposito menù bisogna selezionare la voce “ProBuilderize”.



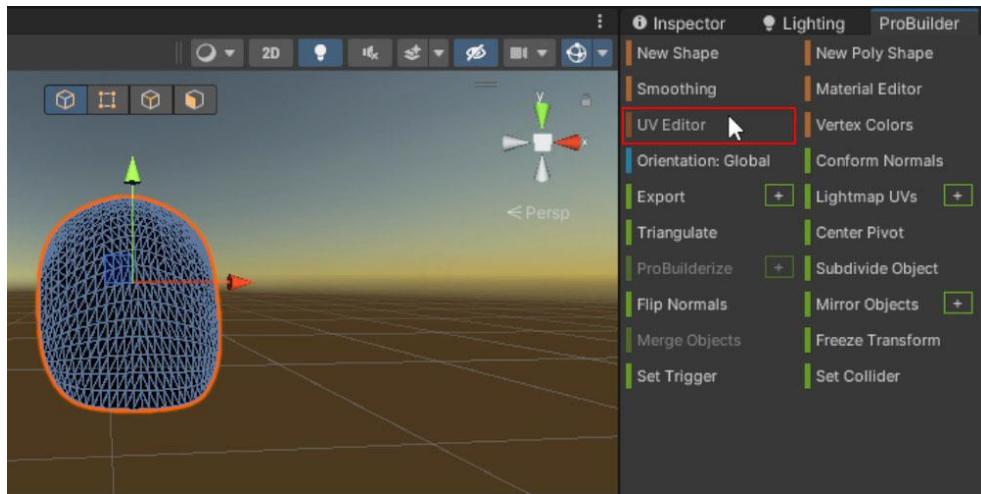
Questo comando fa sì che un oggetto si possa modificare con ProBuilder a seguito di un’operazione di importazione di cui si possono modificare i parametri nella finestra seguente.



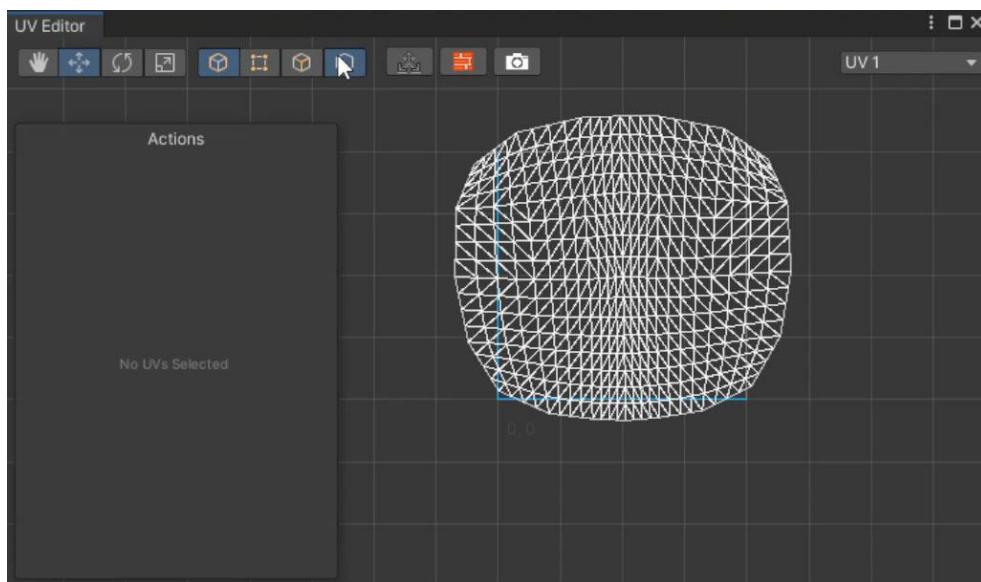
L’opzione “Import Quads” serve ad importare l’oggetto scomponendolo in quadrati. Nel caso dello schermo il risultato della scomposizione in quadrati non è soddisfacente quindi è meglio scomporlo in triangoli.

L’opzione “Import Smoothing” serve per rendere più liscio il modello importato. Non mi serve usare questa opzione perché lo schermo mi serve così com’è.

A questo punto posso procedere con l'importazione.



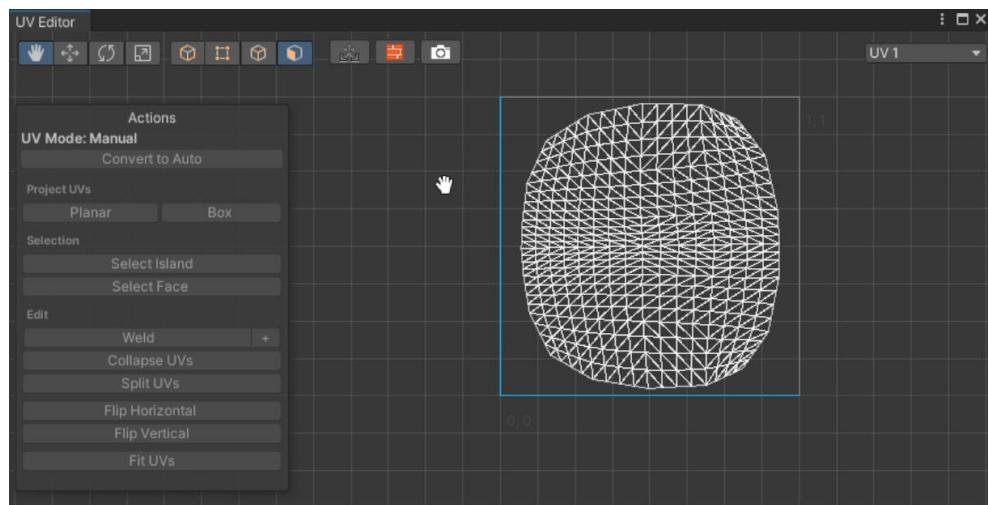
Per ricavare la mappa UV dello schermo devo usare lo UV Editor di ProBuilder.



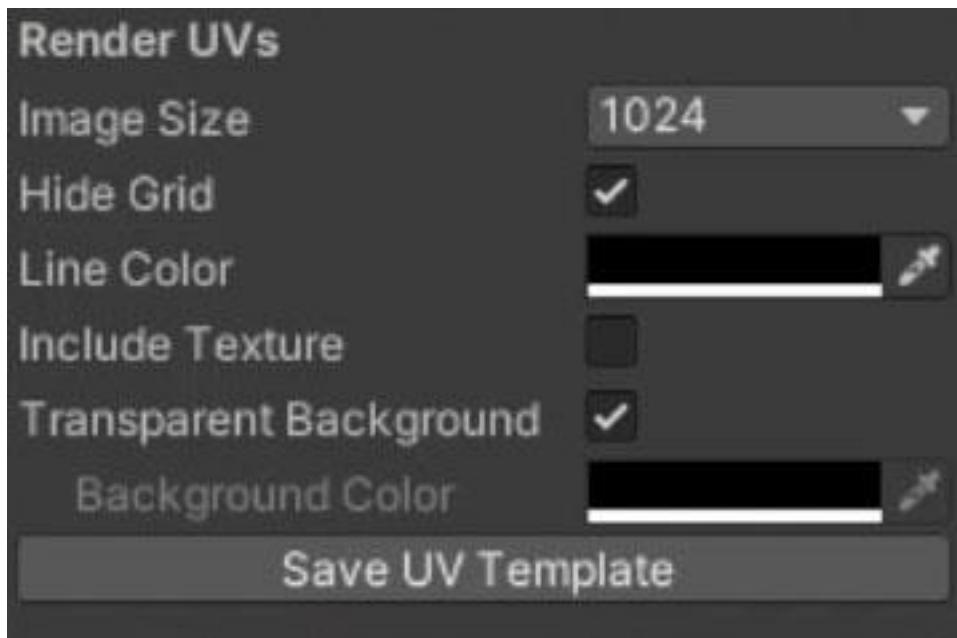
All'interno dello UV Editor si vede la maglia dello schermo e sotto di essa un quadrato blu. Il quadrato blu delimita l'area della griglia che

verrà esportata. Questo mi porta a modificare opportunamente la maglia per farla entrare nel quadrato massimizzando l'area occupata all'interno dello stesso.

Dopo aver scalato e ruotato la maglia di 90° ottengo il seguente risultato.



Ora posso procedere con l'esportazione della mappa UV usando le seguenti impostazioni.



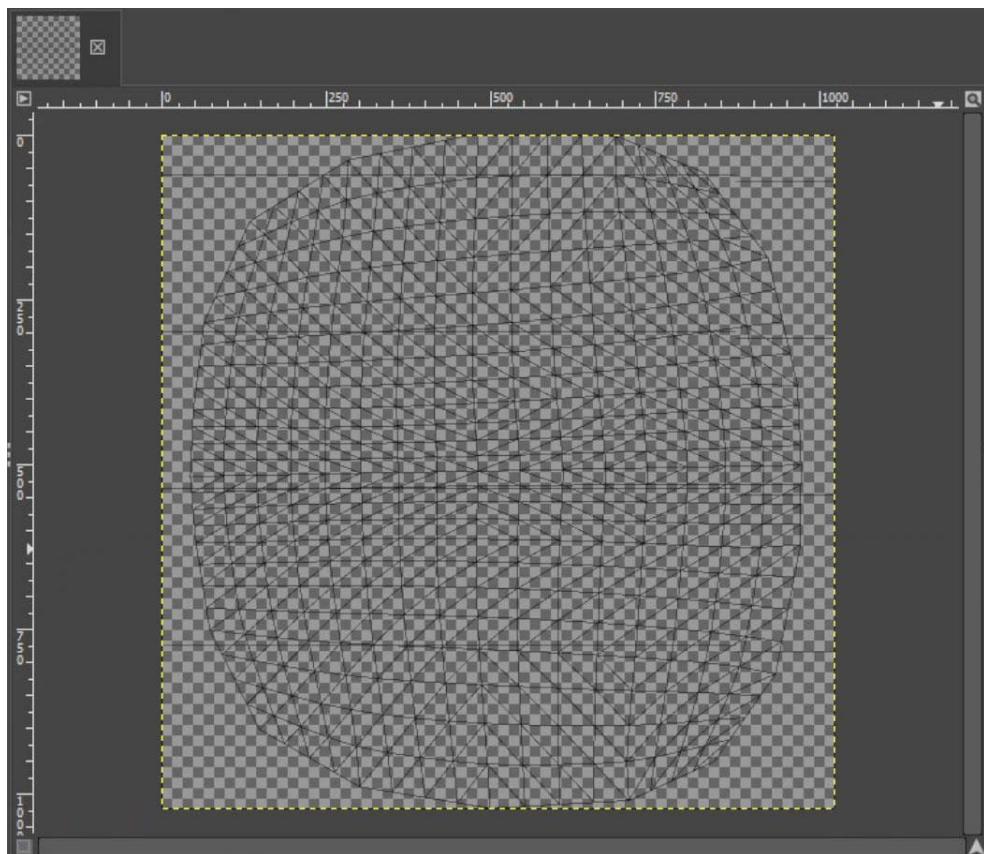
La dimensione dell’immagine è impostata a 1024 di default ed indica il numero di pixel da cui è composto il lato del quadrato esportato. Quindi in questo caso l’immagine esportata sarà di  $1024 \times 1024$  px.

Scelgo l’opzione “Hide Grid” perché tenere la griglia nell’immagine esportata avrebbe un effetto sgradevole.

Imposto il colore della linea a nero perché è un colore che si distingue facilmente dai colori presenti nelle immagini che ho intenzione da applicare allo schermo e questo mi tornerà utile in fase di sviluppo delle texture.

Rimuovo la spunta al parametro “Include Texture” perché mi serve esportare solo la maglia e procedo all’esportazione.

Aprendo l'immagine esportata in Gimp si nota che la maglia esportata è perfetta per le mie esigenze essendo visibile, discreta e con sfondo trasparente.

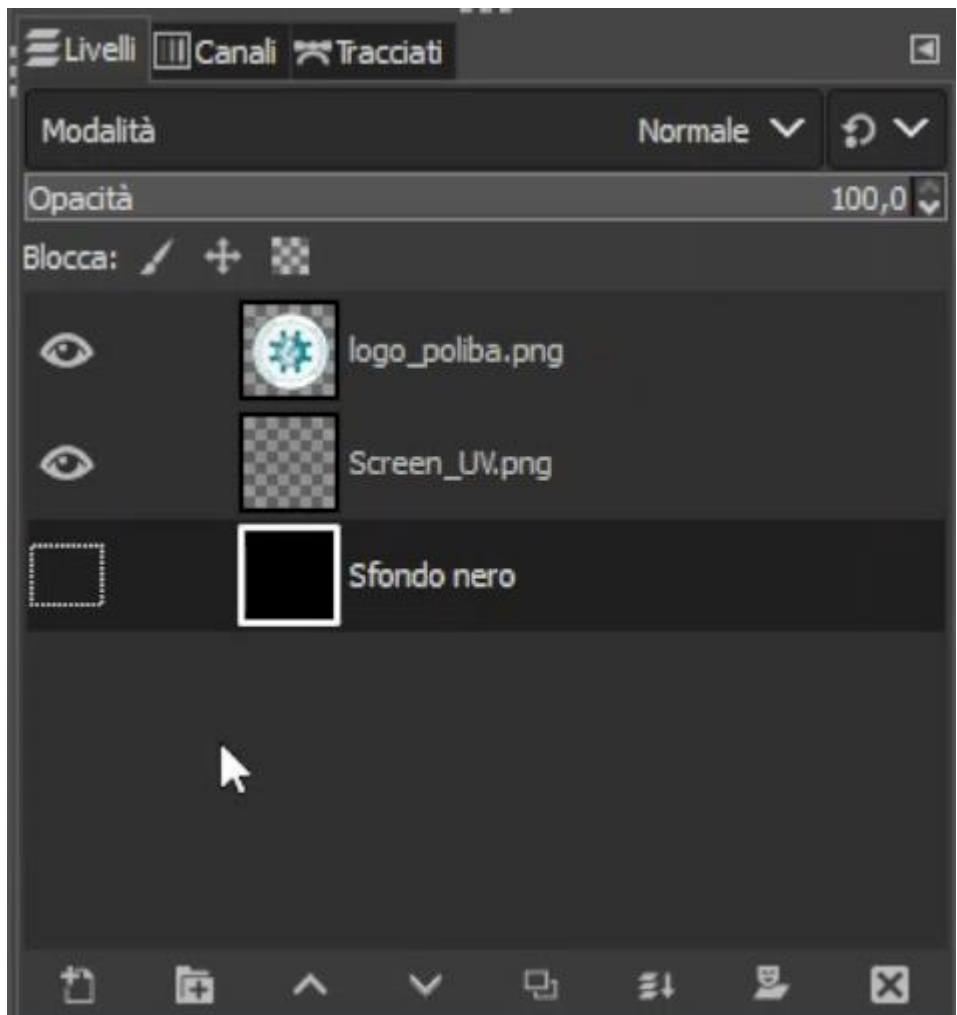


## Creazione delle texture

In Gimp apro il logo del politecnico, lo scalo, lo ruoto di 90° e lo sovrappongo alla maschera UV ottenendo questo risultato.

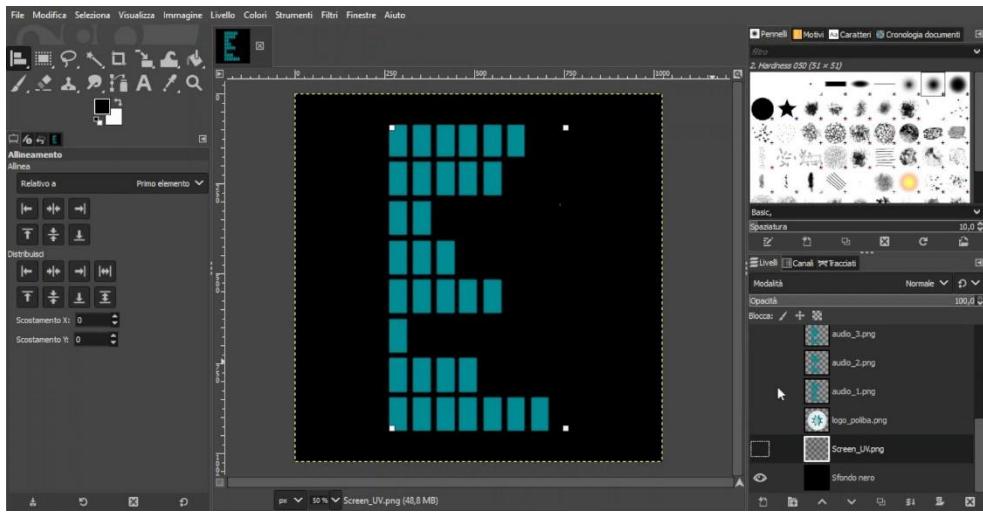


Per ottenere la texture da applicare allo schermo devo inserire uno fondo nero. Per farlo inserisco un altro livello sotto tutti gli altri e lo coloro di nero usando lo strumento “Riempimento di colore”.



Per ora non mi serve rendere visibile lo sfondo perché mi impedirebbe di vedere la maglia e quindi sarebbe scomodo lavorare.

Aggiungo i fotogrammi estratti dalla gif, li ruoto e li scalzo come fatto con il logo del politecnico.



A questo punto posso rendere visibile un solo fotogramma alla volta e lo sfondo nero e procedere all'esportazione. Al termine dell'esportazione di ogni fotogramma si hanno delle texture con l'immagine ruotata di 90° in senso orario.



A questo punto bisogna ricordare che la mappa UV è stata ricavata da una copia dello schermo e che la maglia è stata ruotata e scalata per usare al meglio lo spazio offerto da Unity (il quadrato). La mappa UV dello schermo originale è intatta, quindi la maglia dello schermo è orizzontale come visto all'apertura dello UV Editor. Per adattare le texture alla mappa UV dello schermo ho aperto le immagini usando il

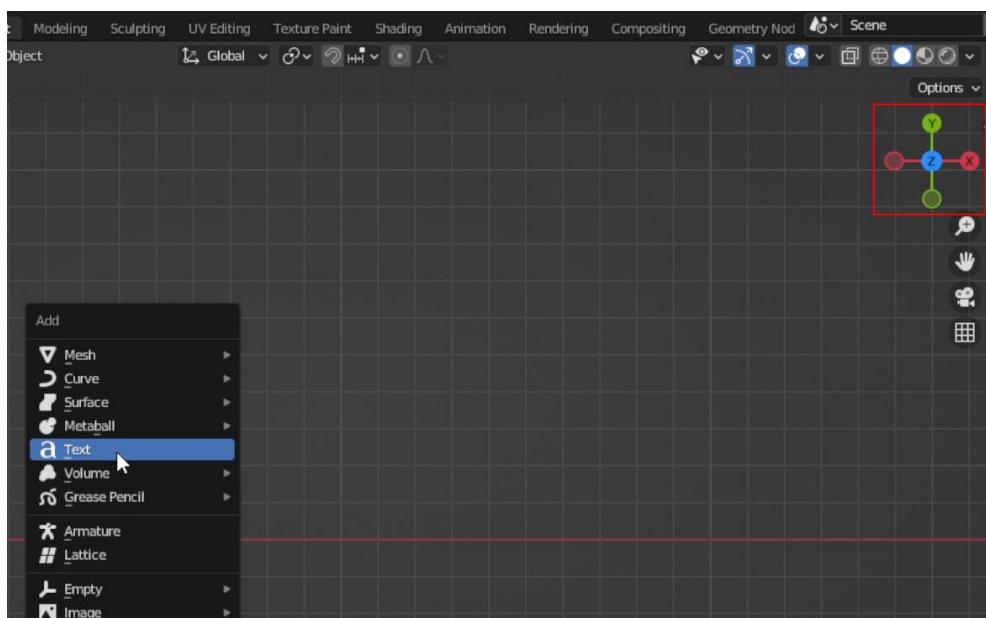
visualizzatore di immagini di Windows e le ho ruotate di 90° in senso antiorario per compensare la rotazione della maglia.

Se inserisco le texture nel progetto di Unity e le applico allo schermo del robot noto che queste sono state realizzate correttamente.

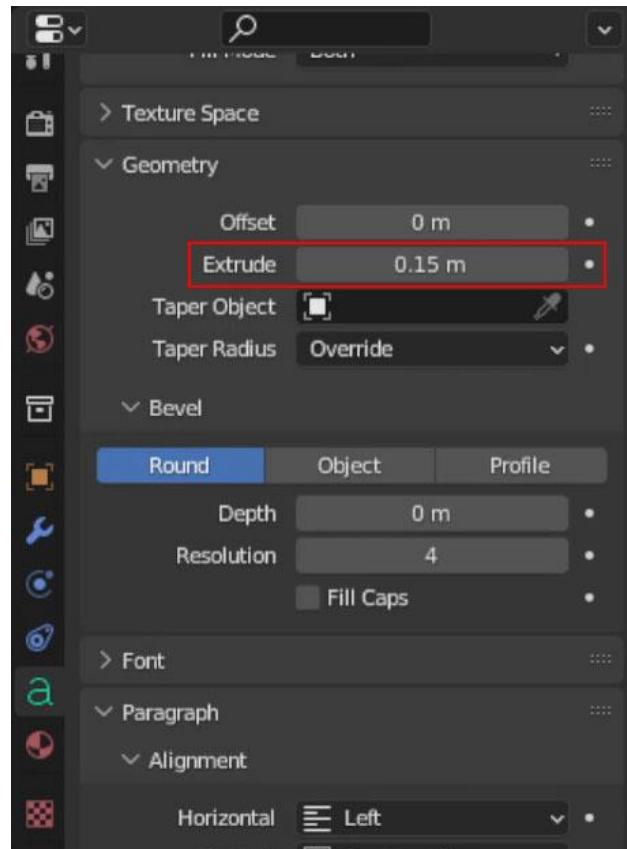


## Realizzazione delle scritte 3D

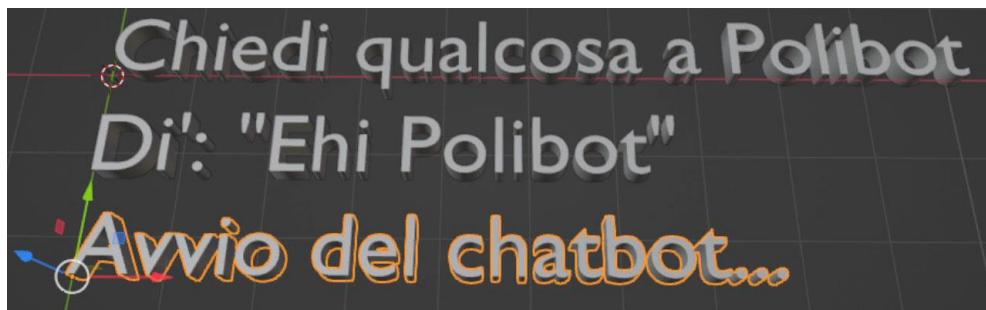
Per la realizzazione delle scritte tridimensionali apro un nuovo file di Blender. Clicco sulla Z nel gizmo per allineare la visuale all'asse Z e poi premo SHIFT+A per aggiungere il testo.



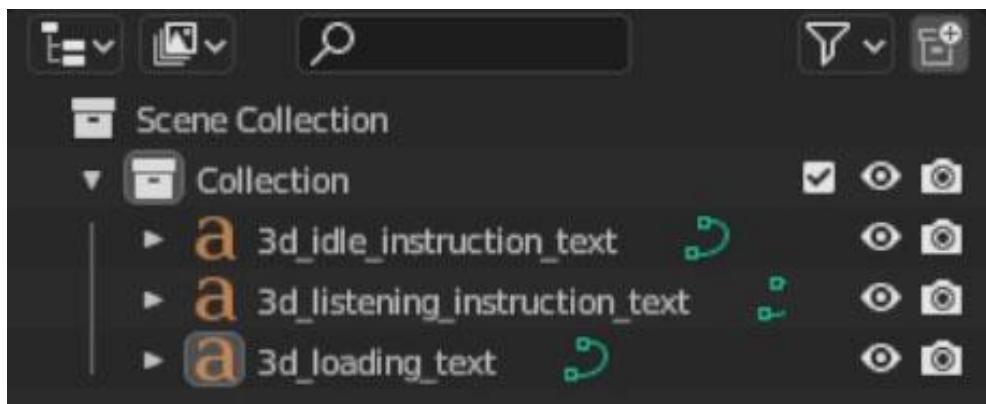
Scrivo una delle frasi che verranno inserite nell'interfaccia dell'applicazione e le assegno uno spessore modificando il parametro “Extrude”.



Allo stesso modo genero anche le altre frasi.

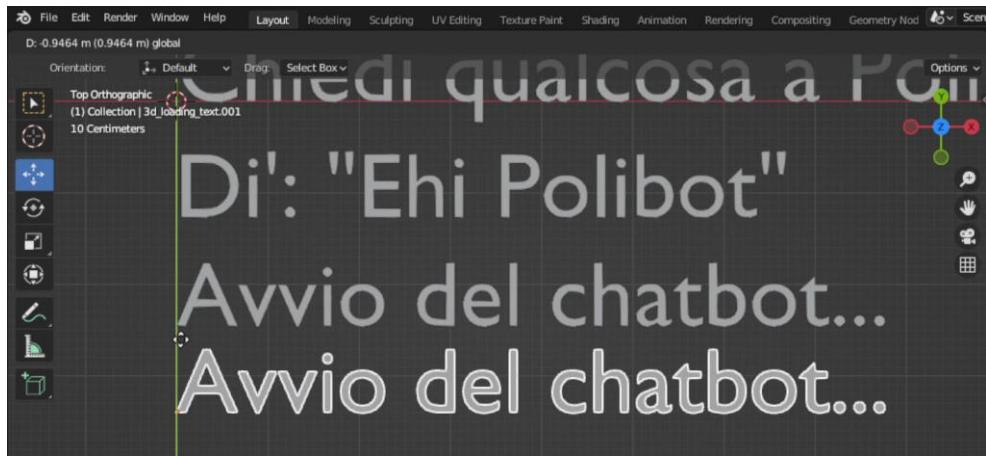


Rinomino le scritte in Blender per rendere il progetto più ordinato ed anche perché quando verranno importate in Unity conserveranno il nome assegnato in Blender.

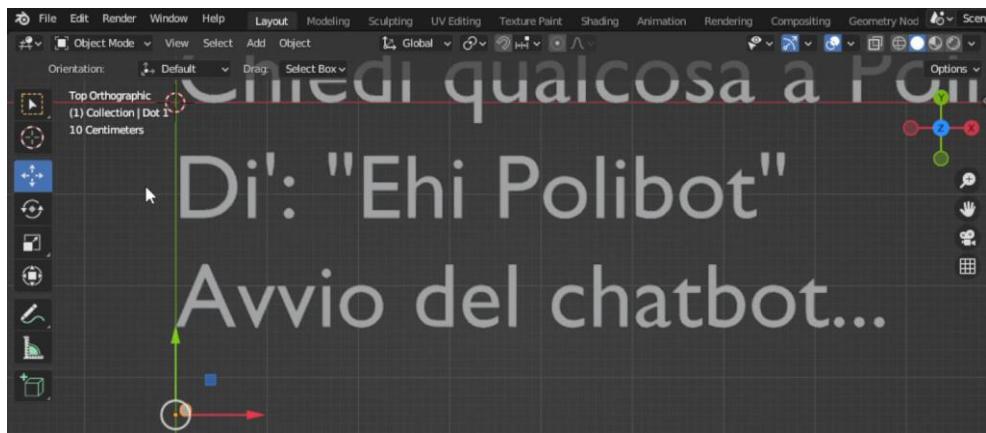


Dato che in Unity ho intenzione di realizzare un'animazione di caricamento facendo apparire progressivamente i puntini devo rendere questi indipendenti dal resto della frase in modo da poterli rendere visibili o invisibili singolarmente.

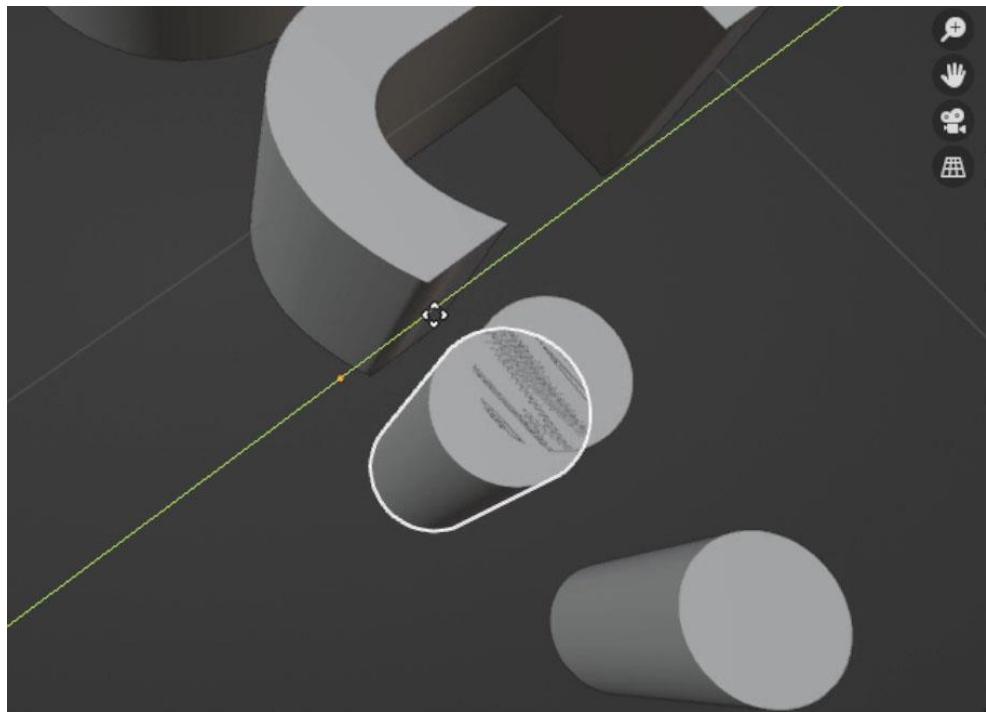
Per farlo duplico la frase di caricamento usando SHIFT+D e la sposto sotto la frase originale.



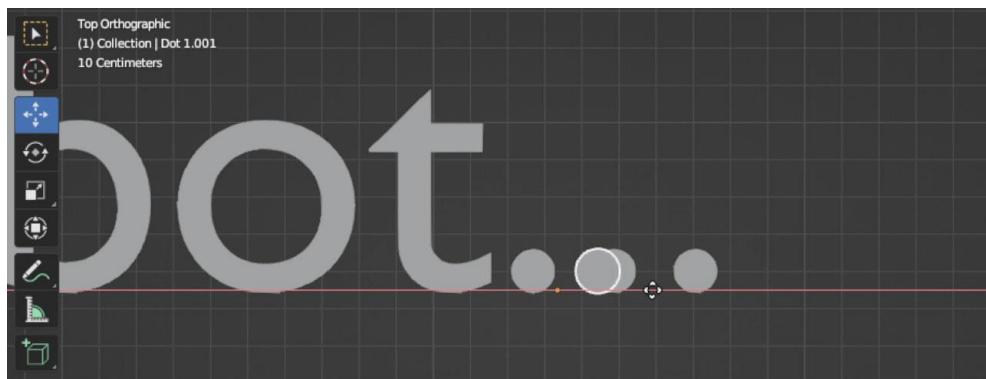
A questo punto devo modificare la frase appena copiata usando il tasto TAB e devo cancellare tutto tranne un punto.



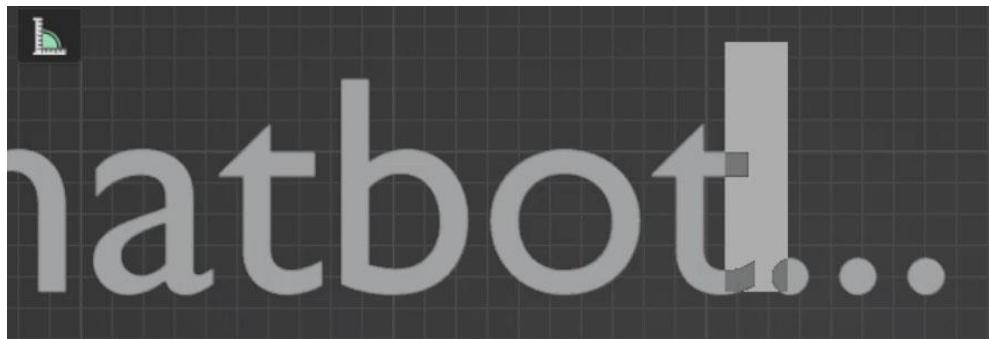
Ora devo spostare il punto per sovrapporlo al primo punto della frase originale.



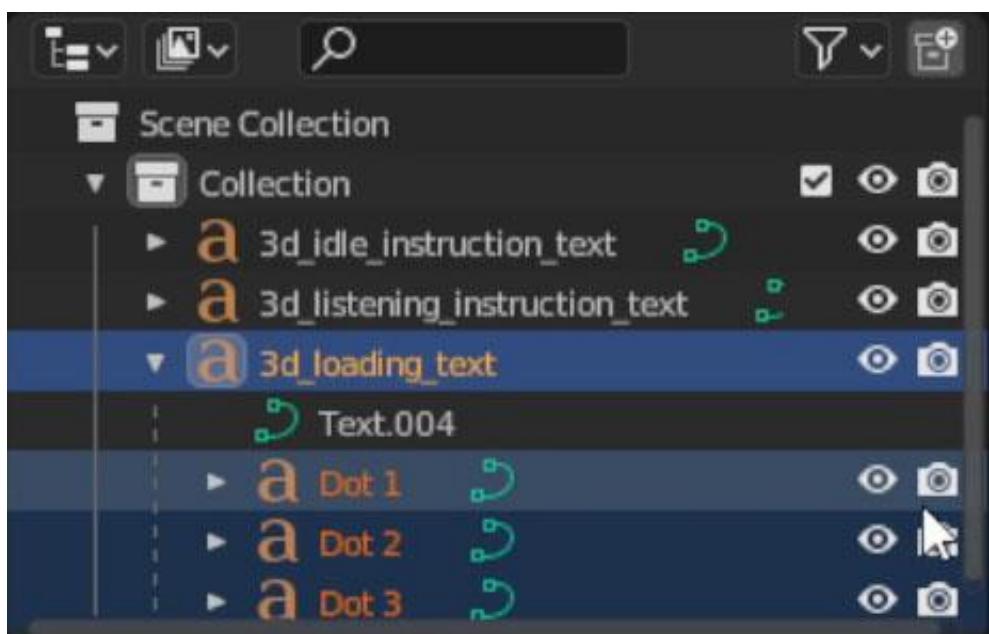
Dopo aver sovrapposto il primo punto lo duplico per sovrapporlo al secondo ed al terzo punto della frase originale.



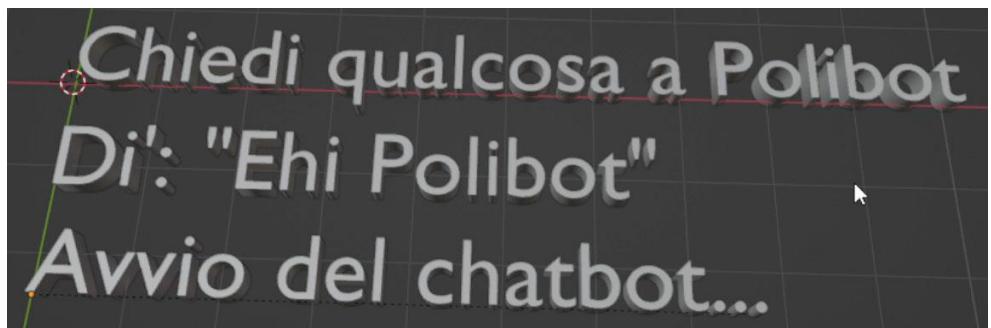
A questo punto devo modificare la frase originale per rimuovere i tre punti.



Per legare i tre punti alla frase devo renderli oggetti figli di quest'ultima e per mantenere il progetto ordinato devo dare loro un nome appropriato.



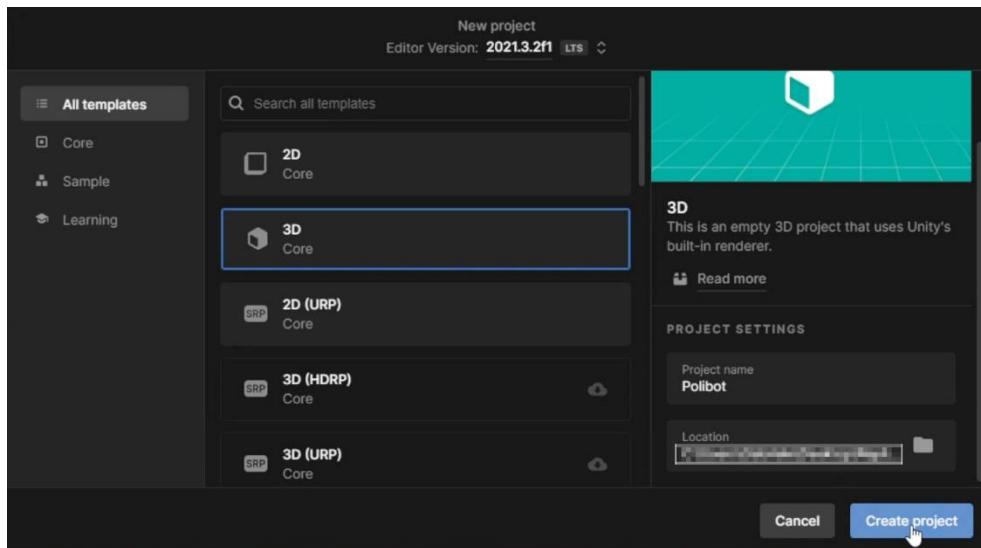
Nel risultato finale non si nota che i tre punti non fanno parte della frase originale.



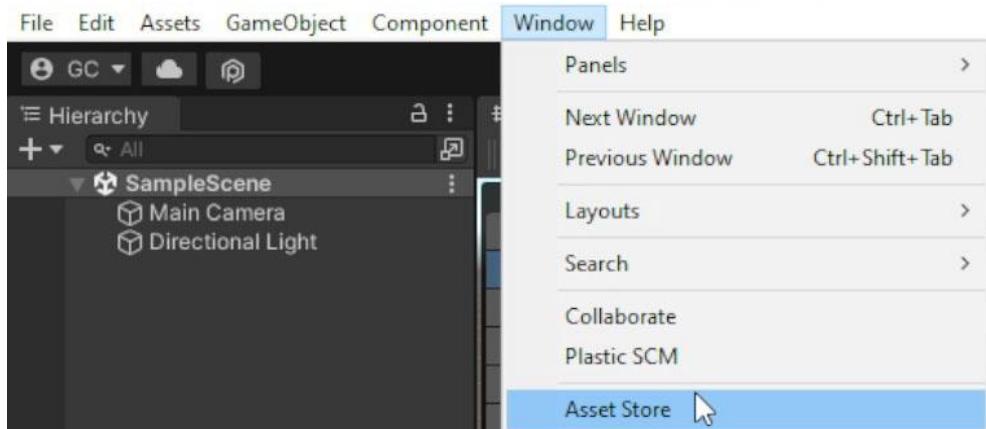
A questo punto posso selezionare una scritta per volta e procedere all'esportazione come fatto con la testa del robot.

## Realizzazione interfaccia

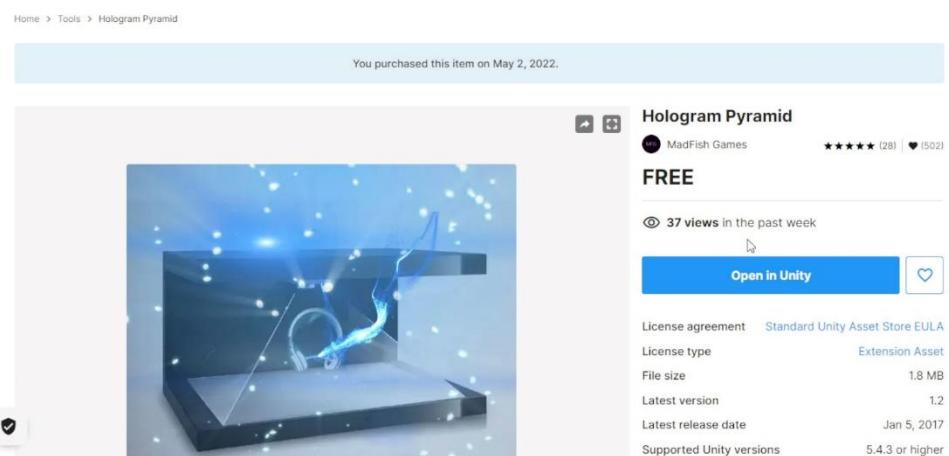
In Unity Hub creo un nuovo progetto, scelgo il template 3D e lo chiamo Polibot.



Per lo sviluppo dell’interfaccia parto da un asset che scarico dall’asset store chiamato “Hologram Pyramid”.



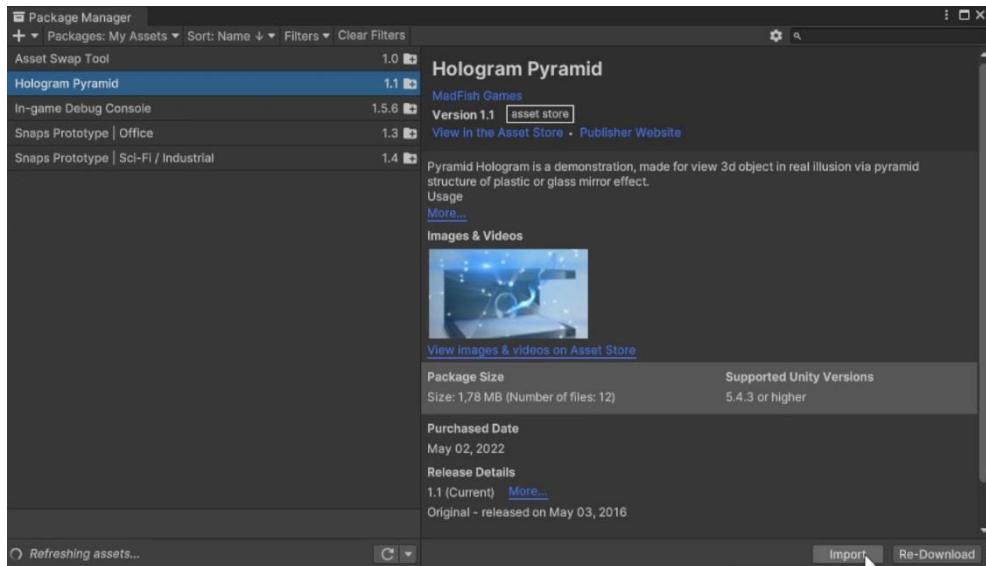
Nell'asset store cerco l'asset e lo apro in Unity.<sup>11</sup>



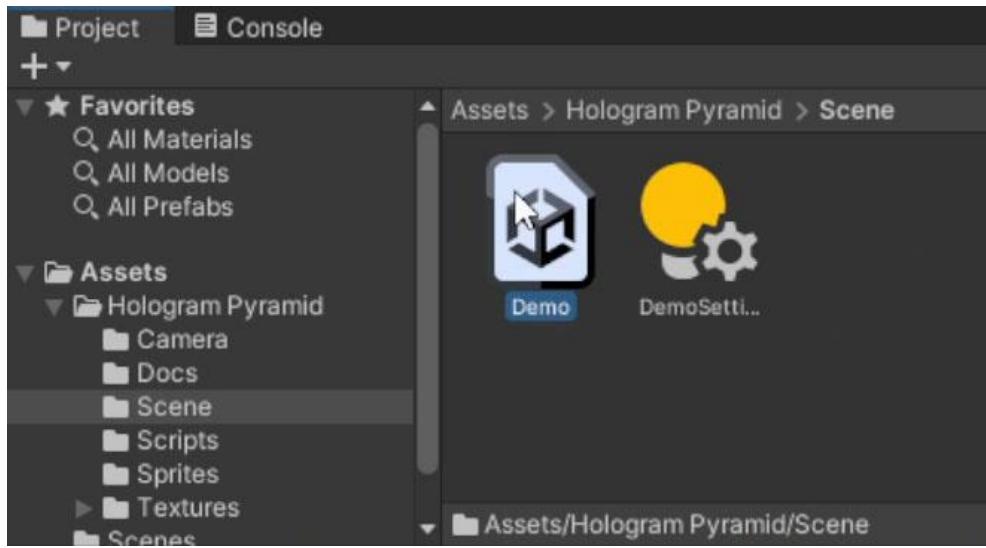
In Unity si apre la finestra del package manager in cui si può scaricare l'asset ed importarlo. Avendolo già scaricato io devo solo importarlo.

---

<sup>11</sup> Unity Asset Store, Hologram Pyramid,  
<https://assetstore.unity.com/packages/tools/hologram-pyramid-61735>



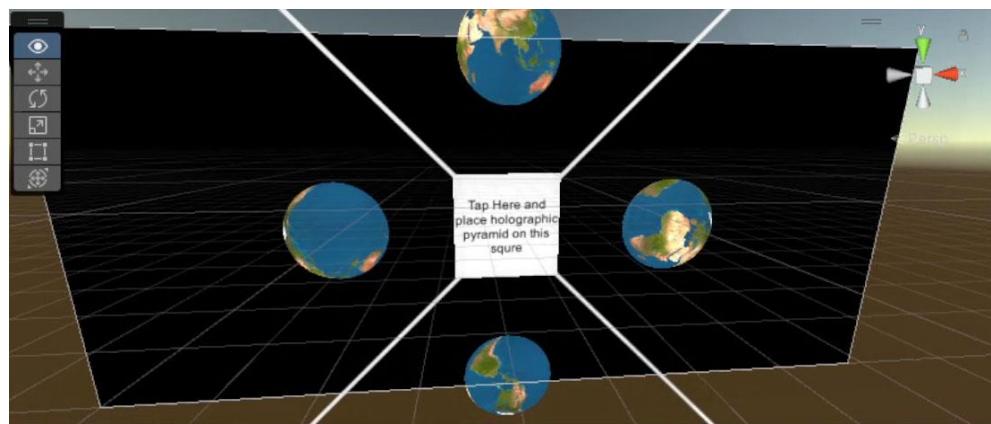
A questo punto nella cartella “Assets” compare la cartella “Hologram Pyramid” al cui interno è presente la cartella “Scene” contenente la scena da cui partirò per lo sviluppo del progetto.



La scena di default è uno sfondo nero con una croce al centro che lo divide in quattro parti. Al centro c’è un messaggio che dice di

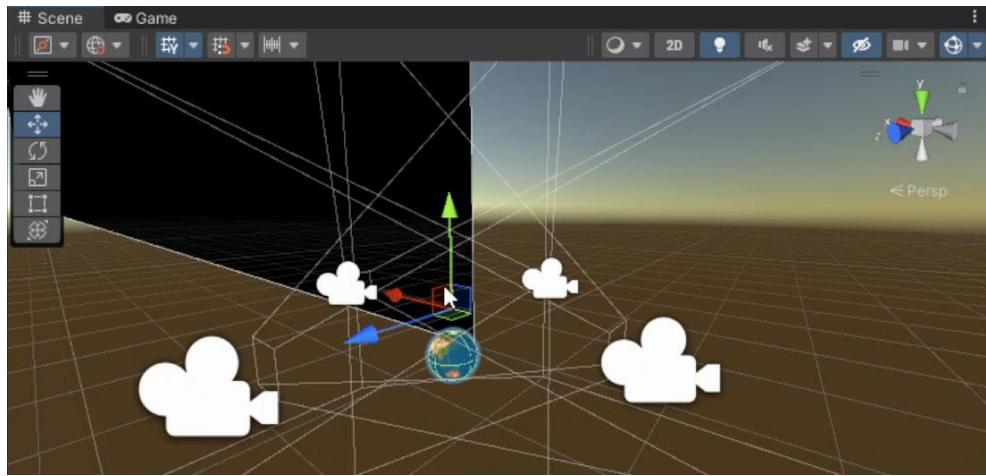
posizionare la piramide olografica al centro dello schermo e poi toccarlo per fare partire l'animazione. Da qui si capisce che questo asset è nato per la realizzazione di video da visualizzare sui dispositivi mobili.

Nelle quattro parti della scena ci sono quattro diverse prospettive di un modello 3D della Terra a cui è assegnato uno script che la fa girare quando si preme play nell'editor.



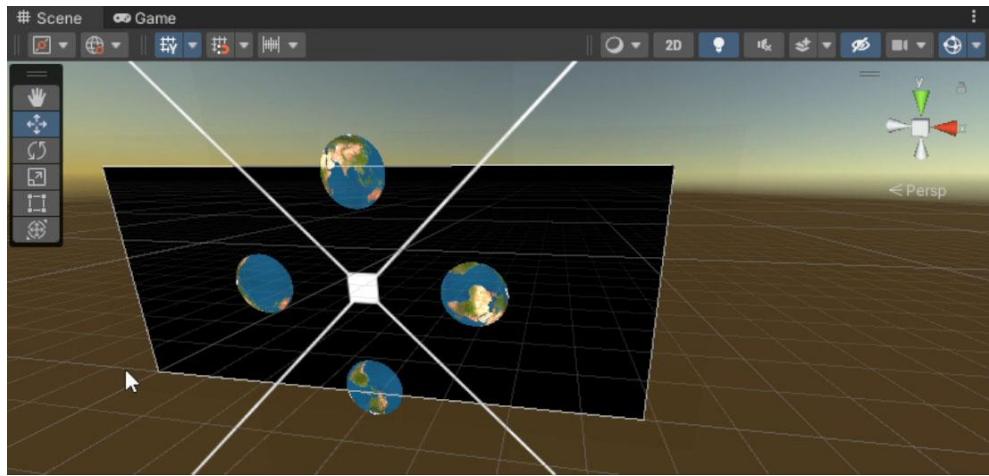
Le prospettive sono realizzate con quattro immagini che usano una texture come fonte.

Queste texture sono generate dalle videocamere che riprendono il modello 3D della Terra da quattro punti di vista diversi.

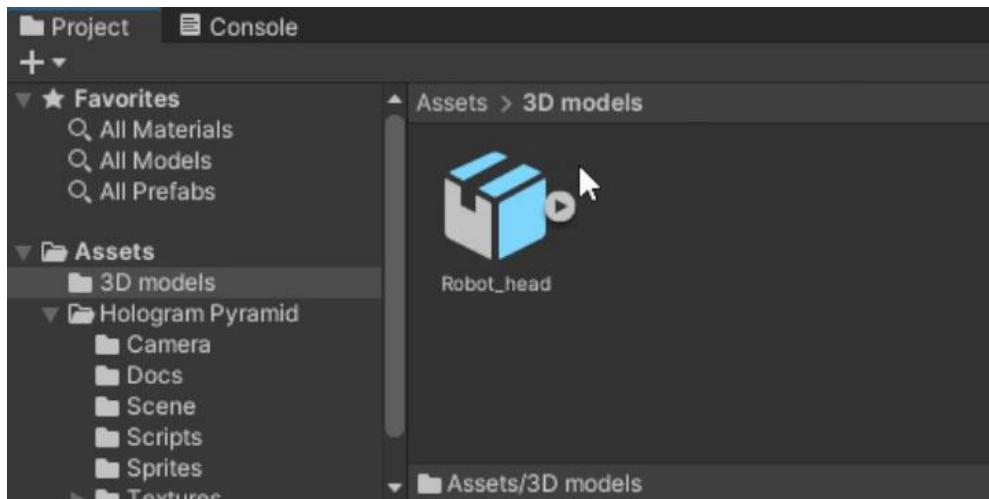


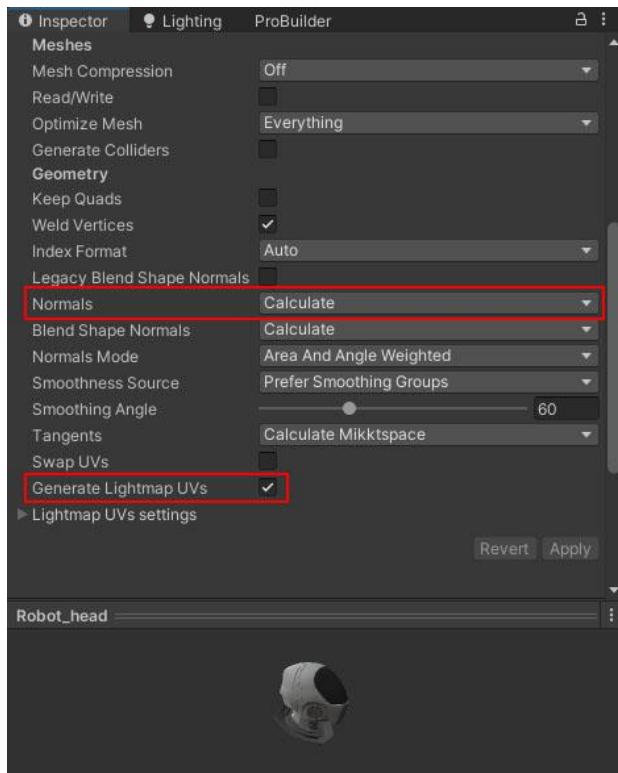
Tutte le parti dell’interfaccia sono contenute in un oggetto “Canvas” a cui è applicato un componente chiamato “Canvas scaler” che fa sì che le dimensioni di questo oggetto scalino con le dimensioni dello schermo. In seguito imposterò questo componente in base alle mie esigenze e fornirò una spiegazione dei suoi parametri.

La prima modifica da fare è rimuovere la scritta centrale perché l’applicazione non verrà usata sui dispositivi mobili, inoltre la sola croce è più discreta.



Ora devo sostituire la Terra con la testa del robot. Per farlo devo creare la cartella “3D models” all’interno del progetto ed importare al suo interno la testa del robot.



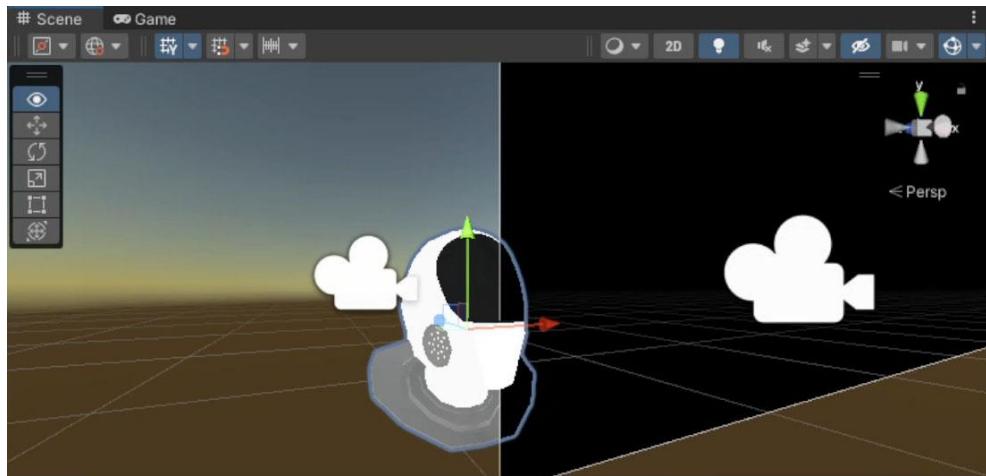


Selezionato il modello, nella finestra di ispezione c’è bisogno che Unity ricalcoli le normali e generi la mappa UV della testa.

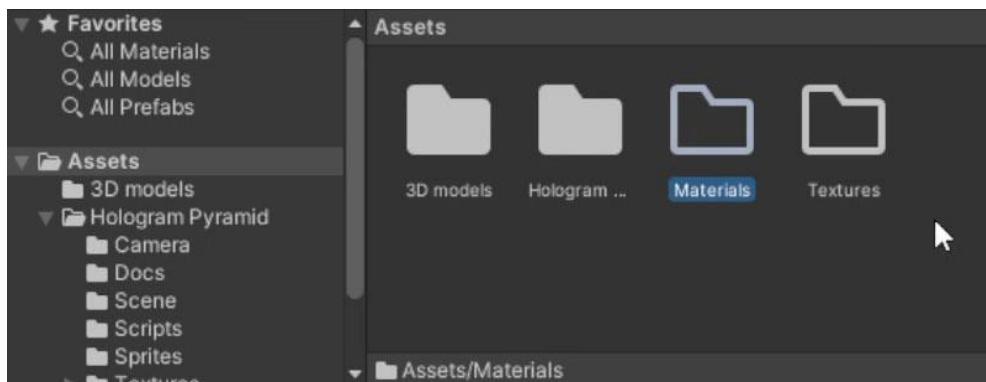
Anche se questi passaggi sono stati effettuati in Blender conviene ripeterli in Unity perché ogni programma che gestisce dei modelli 3D funziona

diversamente.

Dopo aver importato la testa del robot posso sostituirla alla Terra e rinominare l’oggetto che contiene l’ologramma in “Robot hologram”.

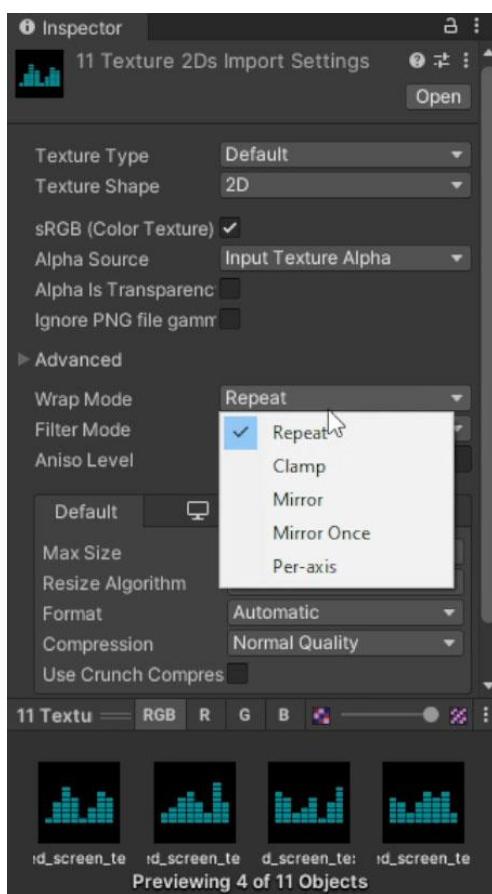
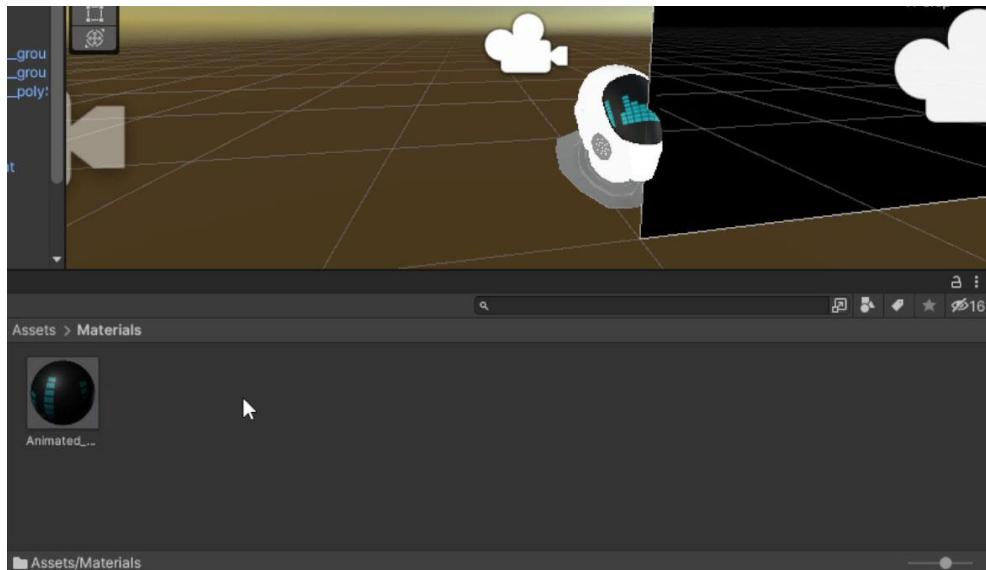


Creo le cartelle “Textures” e “Materials” e nella prima cartella importo le texture dello schermo.



Dopo aver applicato una texture allo schermo viene automaticamente creato l'apposito materiale nella cartella “Materials”.

Applicando tutte le texture allo schermo una ad una ottengo tutti i materiali che mi servono.



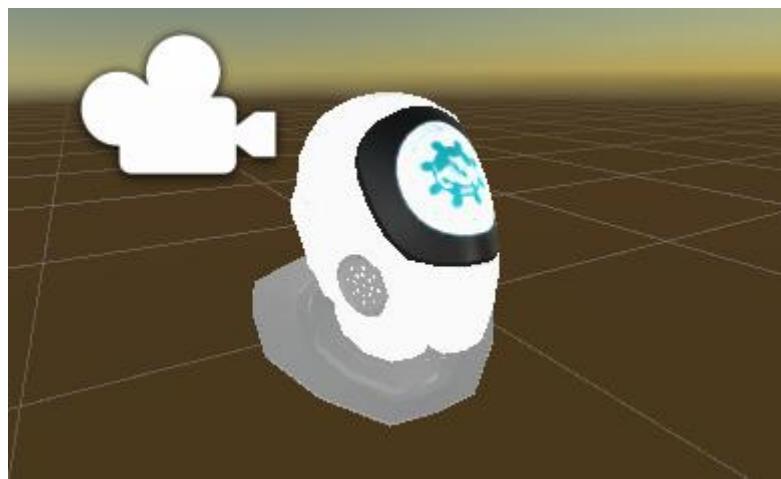
Osservando lo schermo si nota che le texture vengono ripetute. Per risolvere il problema devo selezionare tutte le texture e modificare l'impostazione “Wrap mode”.

Come suggerisce il nome, questa impostazione indica come la texture viene “avvolta” quando questa viene applicata ad un oggetto e le sue dimensioni non sono sufficienti a ricoprirlo. Di default la texture viene ripetuta per ricoprire le parti dell'oggetto che la texture principale non

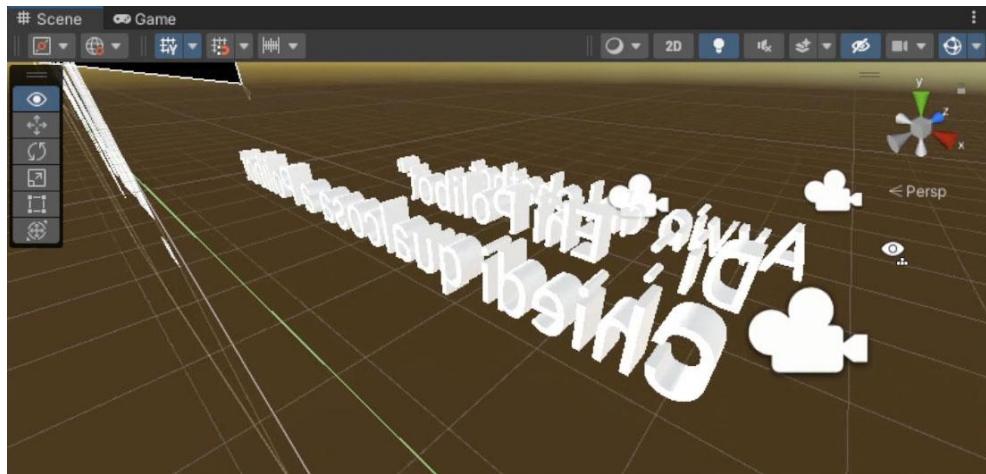
riesce a coprire e questo spiega la parte della texture visibile a sinistra di quella principale nell'immagine precedente.

Io voglio tenere solo la texture principale, perciò devo impostare la wrap mode a “Clamp”.

A seguito di questo cambiamento la texture non si ripete e l'effetto ottenuto è molto più gradevole.



Importo le scritte 3D nel progetto e copio il gruppo “Robot hologram” per realizzare il gruppo “Writings hologram”. Sposto il gruppo copiato dall’altro lato del rettangolo nero, cancello la testa del robot ed inserisco le scritte che ho sviluppato in Blender.

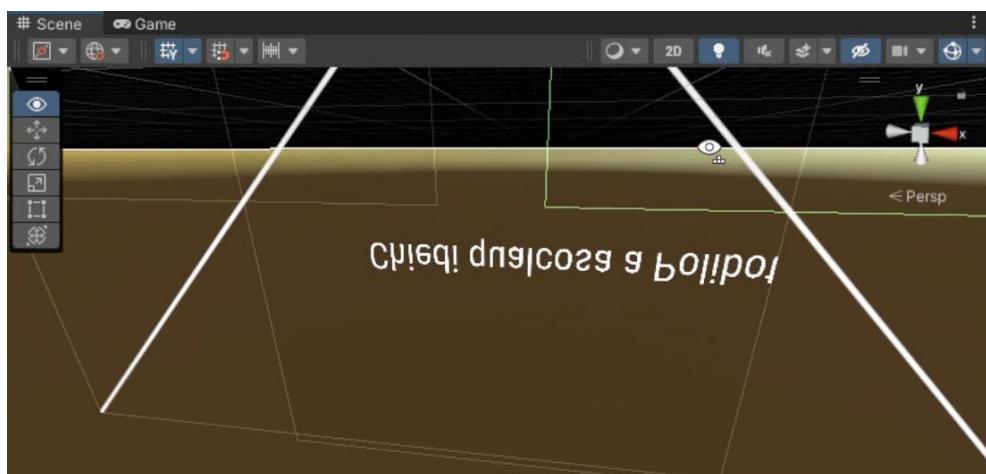


Se le telecamere che riprendono le scritte usassero la modalità di proiezione “Perspective” le scritte sull’interfaccia non sarebbero piatte, piuttosto si noterebbe il loro spessore come si può vedere nell’immagine precedente. Questo fa sì che le scritte siano meno leggibili e le rende sgradevoli.

Per rendere le scritte piatte bisogna modificare la modalità di proiezione ed impostarla ad “Orthographic”.



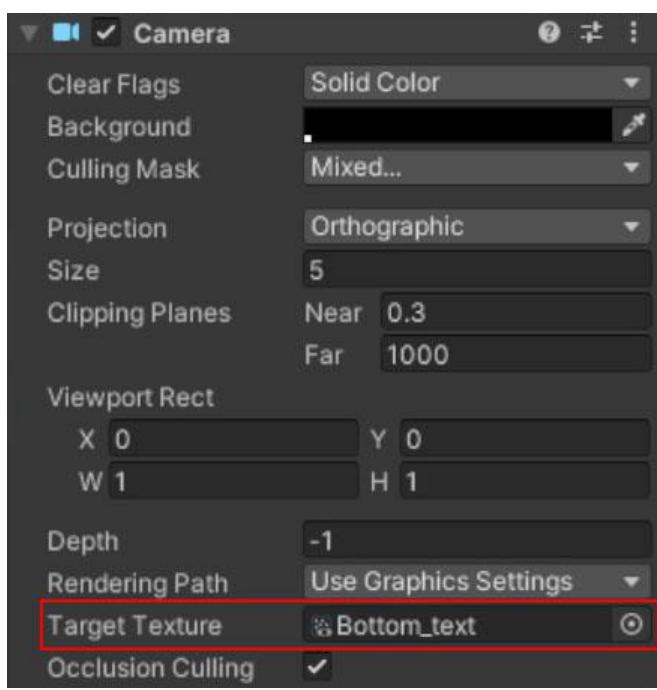
Ora le scritte sono molto più leggibili e gradevoli.



Ora devo impostare la posizione delle telecamere per fare sì che riprendano le scritte correttamente.

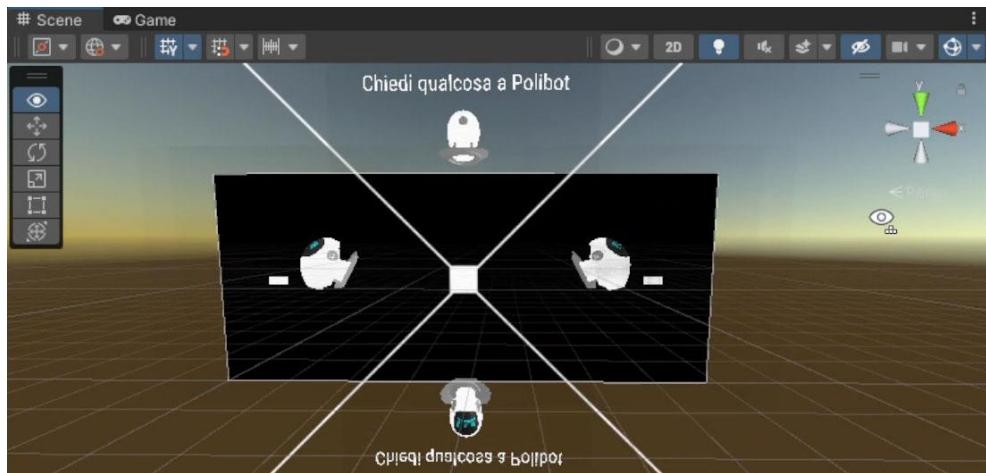
Rendo invisibili le due scritte più corte e tengo la più lunga. Se le telecamere sono posizionate correttamente per la scritta più lunga lo

sono anche per le altre. Dopo aver spostato le telecamere allineo ad occhio le scritte per farle apparire più o meno al centro dell'inquadratura. Dopo aver finito di disporre gli elementi sull'interfaccia farò un lavoro di messa a punto più accurato.



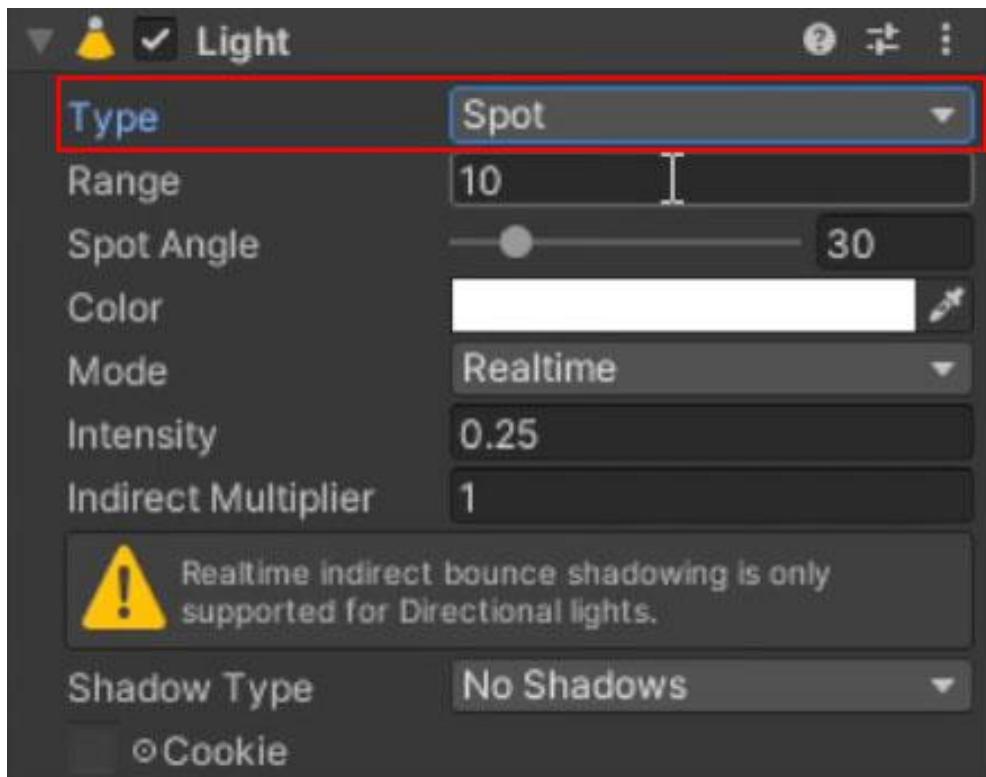
Aumento la risoluzione delle texture della testa da  $1024 \times 1024$  a  $2048 \times 2048$  e le duplico per realizzare le texture dei testi. Rinomino opportunamente le texture da dedicare ai testi e le assegno alle rispettive telecamere.

Al momento nell'interfaccia ci sono solo le immagini della testa del robot. Per inserire le immagini delle scritte duplico quelle della testa, assegno loro opportune texture e le sposto per renderle ben visibili.

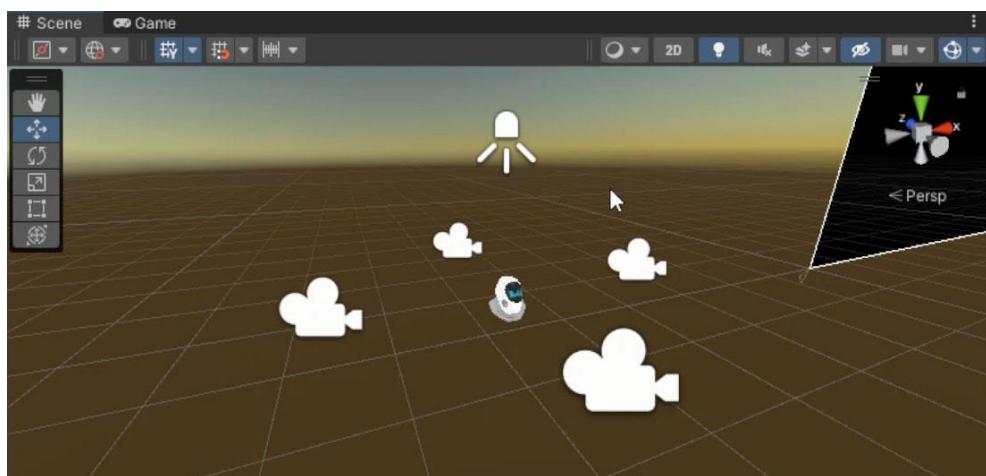


Ora che gli elementi sono disposti nell'interfaccia si nota che le immagini della testa sembrano piatte. Per aggiungere l'effetto della profondità bisogna modificare l'illuminazione della testa.

Modifico le luci delle videocamere che riprendono la testa da “Directional” a “Spot”.



In aggiunta alle luci delle videocamere aggiungo una luce di tipo “Spotlight” sopra la testa. Per ora non modifco i parametri delle luci perché ho intenzione di effettuare la loro messa a punto in seguito.



Per comprendere quanto segue bisogna conoscere le lightmap e le modalità di illuminazione.

Una lightmap è una struttura dati precalcolata che immagazzina la luminosità delle superfici di un modello 3D. In Unity le lightmap sono texture quadrate che vengono sovrapposte agli oggetti per alterarne la luminosità dando così l'impressione che questi siano illuminati.<sup>12</sup>

Le modalità di illuminazione sono tre: Realtime, Baked e Mixed.

- L'illuminazione Realtime è un'illuminazione calcolata per ogni frame. Questa modalità di illuminazione è indicata per gli oggetti dinamici la cui luminosità cambia nel tempo, quindi è necessario farla cambiare in tempo reale.<sup>13</sup>
- L'illuminazione Baked è un'illuminazione precalcolata e memorizzata nelle lightmap. Questa modalità di illuminazione è indicata per gli oggetti statici la cui luminosità non varia nel tempo. Calcolare la luminosità di questi oggetti per ogni frame produrrebbe sempre lo stesso risultato, pertanto è inutile.<sup>14</sup>

---

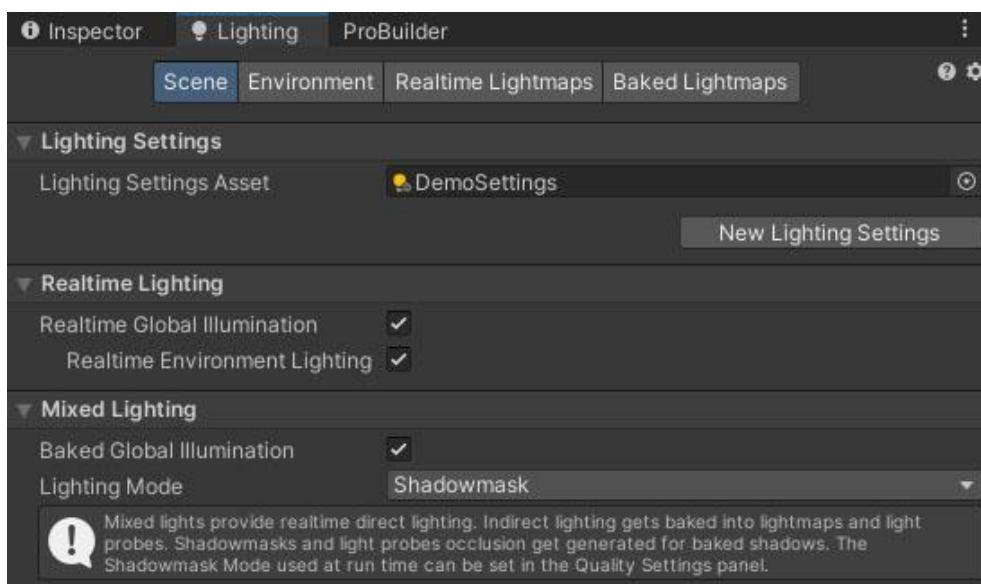
<sup>12</sup> Unity Documentation, Lightmapping,  
<https://docs.unity3d.com/Manual/Lightmappers.html>

<sup>13</sup> Unity Documentation, Real-time lighting,  
<https://docs.unity3d.com/560/Documentation/Manual/LightMode-Realtime.html>

<sup>14</sup> Unity Documentation, Light Mode: Baked, <https://docs.unity3d.com/Manual/LightMode-Baked.html>

- L’illuminazione Mixed è un mix delle due. Usare questa modalità serve a fare sì che la luminosità degli oggetti statici venga memorizzata nelle lightmap e quella degli oggetti dinamici venga calcolata per ogni frame.<sup>15</sup>

Accedo al menù dell’illuminazione tramite Window > Rendering > Lighting.



Tra i tanti parametri dell’illuminazione uno dei più importanti è la sua modalità.

---

<sup>15</sup> Unity Documentation, Light Mode: Mixed, <https://docs.unity3d.com/Manual/LightMode-Mixed.html>

Prima di parlare di questo parametro bisogna comprendere i concetti di luce diretta ed indiretta.

Nel mondo reale quando la luce entra in contatto con gli oggetti parte di viene riflessa ed il resto viene assorbita dal materiale e dissipata sotto forma di calore. La luce diretta è quella che un oggetto riceve direttamente dalla fonte di luce, mentre quella indiretta è quella che riceve dal riflesso di altri oggetti. Nel mondo reale una luce viene riflessa finché non viene completamente assorbita dai materiali e questo è un processo teoricamente infinito. In Unity è possibile scegliere il numero di volte in cui un raggio di luce può essere riflesso dagli oggetti. Anche in questo caso il compito dell'utilizzatore di Unity è impostare il numero minimo di rimbalzi tali da ottenere una illuminazione di qualità soddisfacente. Aumentare a dismisura il numero di rimbalzi rende i calcoli troppo onerosi e genera problemi di latenza!<sup>16</sup>

Si può scegliere tra tre modalità: Baked Indirect, Shadowmask e Subtractive.<sup>17</sup>

- La modalità Baked Indirect calcola la luce diretta per ogni frame e precalcola quella indiretta;

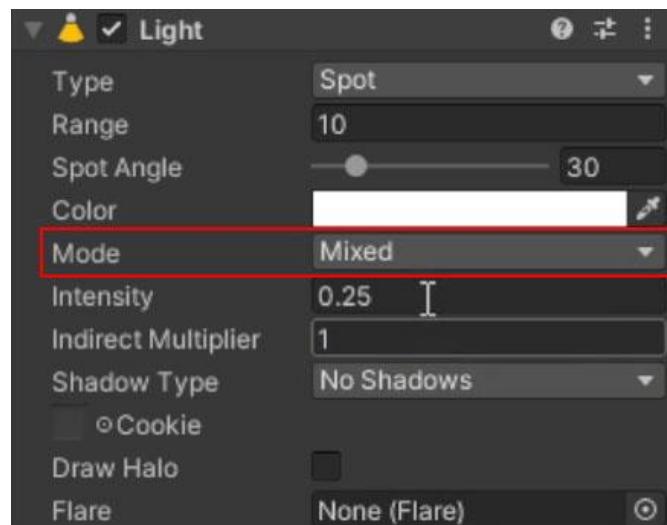
---

<sup>16</sup> Unity Documentation, Introduction to lighting,  
<https://docs.unity3d.com/Manual/LightingInUnity.html>

<sup>17</sup> Unity Documentation, Lighting Mode, <https://docs.unity3d.com/Manual/lighting-mode.html>

- La modalità Shadowmask è come la Baked Indirect ma ha una gestione migliore delle ombre;
- La modalità Subtractive precalcola sia la luce diretta che indiretta, perciò è adatta per gli hardware di fascia bassa.

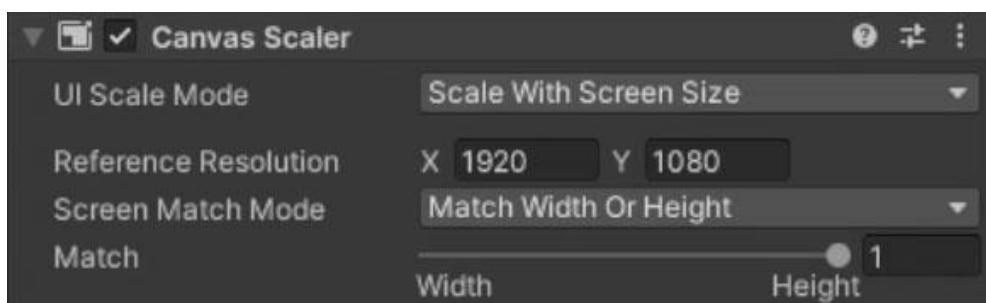
Imposto le modalità delle luci a mixed così da generare una lightmap per gli oggetti contrassegnati come statici ed usare l'illuminazione in tempo reale per gli oggetti dinamici.



A seguito della messa a punto dei parametri delle luci, dell'illuminazione e della posizione degli elementi dell'interfaccia ottengo questo risultato.



L’oggetto di Unity che contiene tutti gli elementi dell’interfaccia grafica si chiama “Canvas”. A questo oggetto si può applicare un componente chiamato “Canvas Scaler” che serve a fare scalare gli elementi dell’interfaccia in maniera solidale allo schermo.

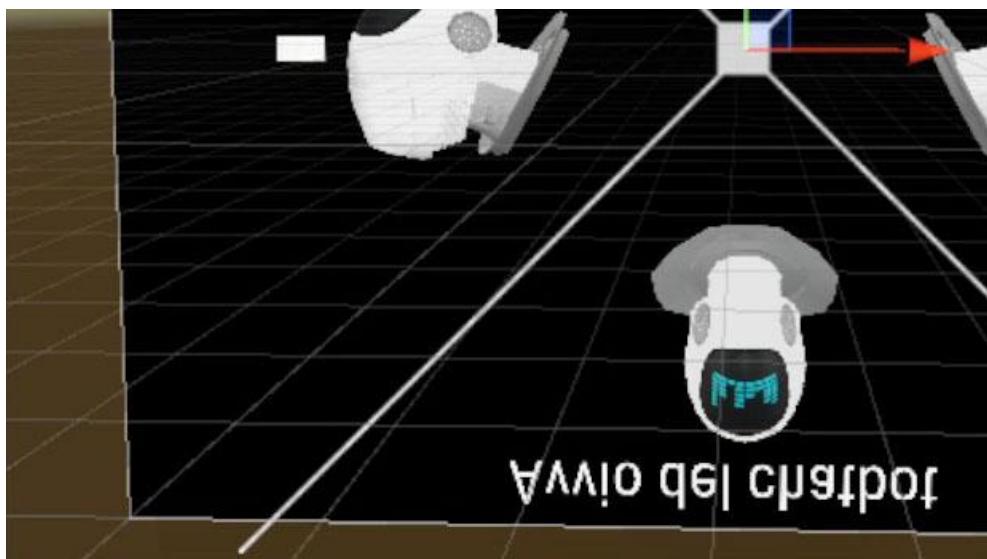


All’interno del componente si può scegliere una risoluzione di riferimento. Questo parametro imposta il modo in cui si vuole che l’interfaccia venga percepita dall’utente. In questo caso io voglio che

l'utente veda l'interfaccia come se la sua risoluzione fosse 1920 × 1080.

Imposto il modo in cui l'interfaccia combacia con lo schermo in modo che questa si riferisca o alla sua larghezza o alla sua altezza. Sotto questo parametro è presente uno slider in cui si può scegliere a quale dimensione dare la priorità. Io do la priorità completamente all'altezza per fare sì che la croce all'interno dell'interfaccia tocchi sempre i bordi dello schermo.

Per questo stesso scopo scalo anche la croce per fare sì che le sue punte escano leggermente dallo sfondo nero.



La testa del robot verrà visualizzata in una piramide olografica in cui sembrerà che la testa guardi verso l'utente. Se questo guarda la faccia della piramide alla sua destra vedrà la faccia del robot orientata verso

sinistra perché questo guarda in avanti. Allo stesso modo se guarda la faccia della piramide alla sua sinistra vedrà la faccia del robot orientata verso destra. Per questo motivo devo modificare l'orientazione delle texture nell'interfaccia per renderla realistica.

Per realizzare l'immagine del robot e della scritta da collocare nella parte destra dell'interfaccia devo spostare e ruotare le immagini dell'interfaccia che prima erano a sinistra. Allo stesso modo ricavo le immagini da usare nella parte sinistra. Dopo aver corretto l'interfaccia da un punto di vista grafico devo anche modificare i nomi delle texture e delle telecamere per fare sì che chi si occuperà del progetto dopo di me non si confonda. (Se non cambiassi i nomi della telecamera, della texture e delle immagini a sinistra queste avrebbero la parola “Right” nel nome e questo creerebbe confusione).



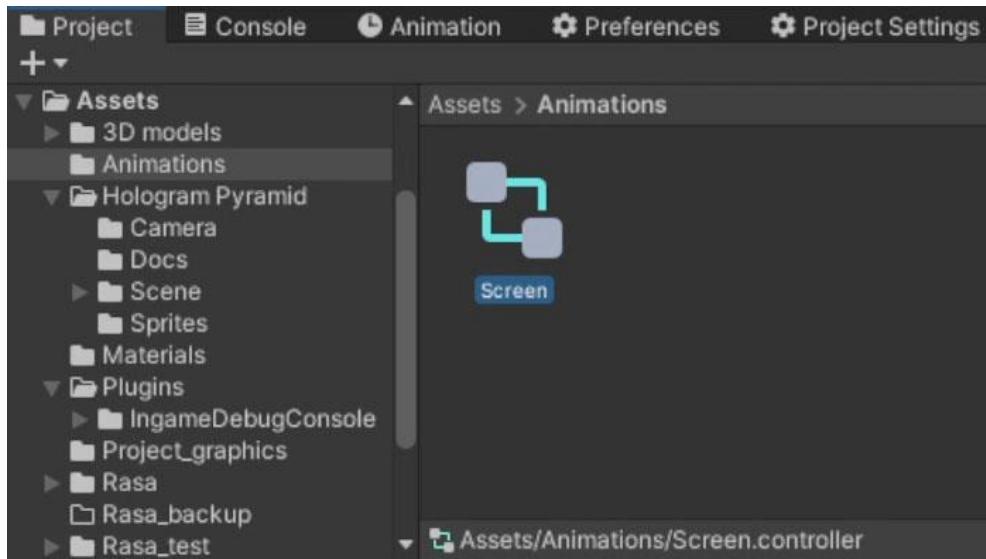
A questo punto nell'editor di Unity posso provare a modificare le dimensioni della scheda “Game” a mio piacimento ed osservare che l'interfaccia viene scalata di conseguenza.

## **Realizzazione delle animazioni**

La finestra delle animazioni si divide in due colonne. Nella colonna a sinistra si possono impostare l'animazione su cui lavorare, le proprietà da variare nel tempo ed il numero di campioni, cioè gli istanti in cui si possono impostare le variazioni delle proprietà.

Nella colonna a destra è presente una linea temporale per ogni proprietà. In ciascuna linea temporale si possono impostare dei punti chiave in corrispondenza dei campioni ed in ciascun punto chiave si può impostare una modifica della proprietà in questione.

Per realizzare le animazioni dello schermo devo selezionarlo poi devo aprire la finestra delle animazioni andando in Window > Animation > Animation. Nella finestra delle animazioni verrà richiesto di creare un controllore per l'animazione. Creandolo comparirà il controllore “Screen” nella cartella “Animations”.



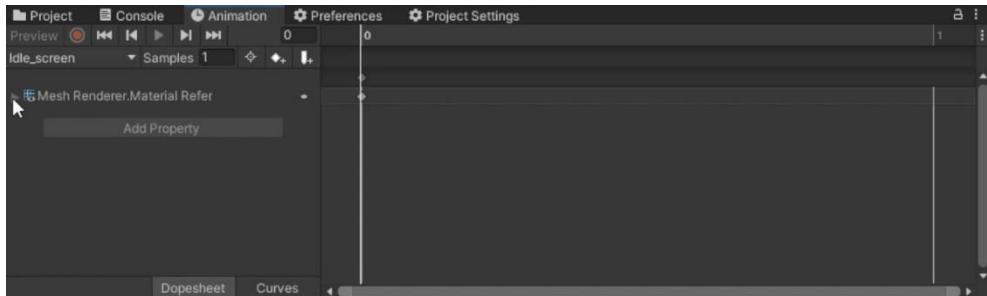
A questo punto posso cominciare a sviluppare le animazioni. Parto dall'animazione di idle, cioè l'animazione che verrà mostrata quando il robot non parla.

Nella colonna a sinistra imposto il numero di campioni ad uno e seleziono il materiale come proprietà da variare nel tempo.

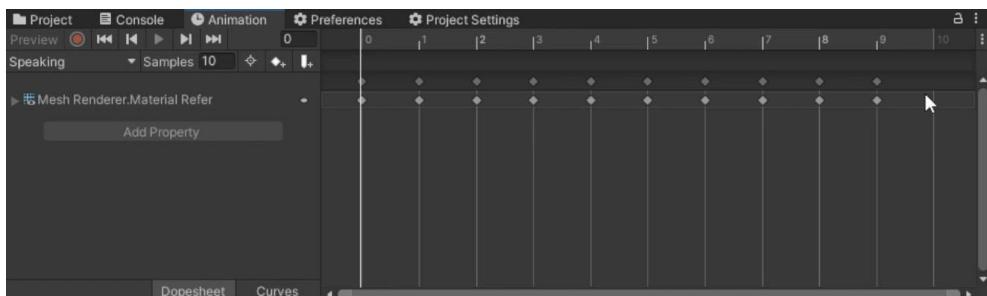
Nella colonna a destra imposto il materiale relativo alla texture di idle in corrispondenza dello zero.

Anche se il materiale usato è sempre lo stesso devo comunque usare un'animazione perché mi serve utilizzare i servizi forniti dal controllore che imposterò in seguito.

Facendo delle ricerche sul forum di Unity ho scoperto che un'animazione fatta in questo modo ha un impatto trascurabile sulle prestazioni del programma, quindi non è un grosso problema.<sup>18</sup>



Sviluppo l'animazione di speaking da usare quando il robot parla. Nella colonna a sinistra scelgo il materiale come proprietà da variare ed imposto dieci campioni. Nella colonna a destra imposto nei campioni da 0 a 9 i materiali relativi alle texture da 1 a 10 ricavate dalla gif.



Dopo aver sviluppato le animazioni devo impostare il controllore. La finestra del controllore si può aprire o da Windows > Animation >

---

<sup>18</sup> Unity Forum, How to create a static first state for an animation controller?, <https://forum.unity.com/threads/how-to-create-a-static-first-state-for-an-animation-controller.432741/>

Animator oppure facendo doppio click sul controllore nella cartella Animation visto in precedenza.

La finestra dell'animatore ha due colonne. Nella colonna a sinistra si possono impostare i livelli di animazione ed i parametri delle animazioni e nella colonna a destra si possono impostare le transizioni tra i vari stati.

I livelli delle animazioni servono a suddividere le animazioni complesse in parti. Ad esempio, se si dovesse animare un modello 3D di un umano si potrebbe usare un livello per le animazioni delle braccia, uno per le animazioni delle gambe ed uno per le animazioni del viso.<sup>19</sup>

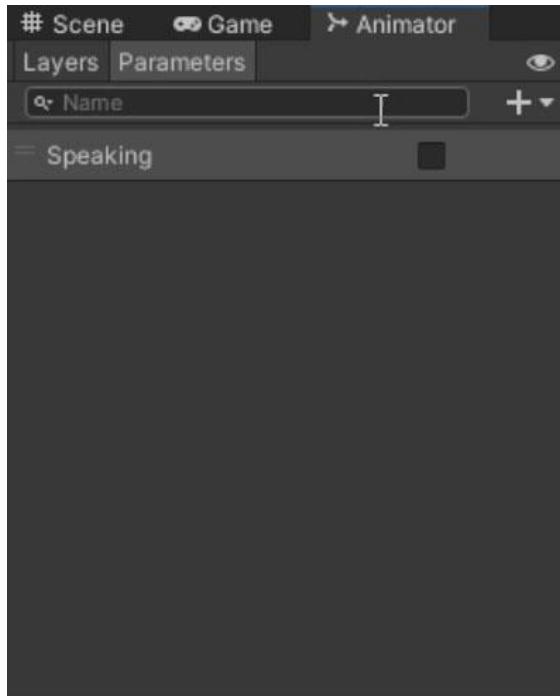
I parametri delle animazioni sono delle variabili usate per regolare le transizioni tra gli stati.

Gli stati sono associati alle animazioni oppure sono gli stati veri e propri in cui si può trovare il programma come gli stati di “Entry” ed “Exit” che rappresentano rispettivamente lo stato in cui il programma viene avviato e viene chiuso. È presente anche uno stato speciale chiamato “Any State”, questo stato serve a rappresentare tutti gli stati ed ha lo scopo di rendere più semplice la realizzazione di transizioni da tutti gli stati verso uno stato in particolare. Se non esistesse questo stato speciale per realizzare una transizione da tutti gli stati bisognerebbe realizzare

---

<sup>19</sup> Unity Documentation, Animation Layers,  
<https://docs.unity3d.com/Manual/AnimationLayers.html>

una transizione per ogni stato e se si dovesse aggiungere un ulteriore stato bisognerebbe sviluppare le transizioni anche per questo. Questo non sarebbe solo un lavoro lungo e certosino ma renderebbe anche la manutenzione del progetto più difficile.<sup>20</sup>



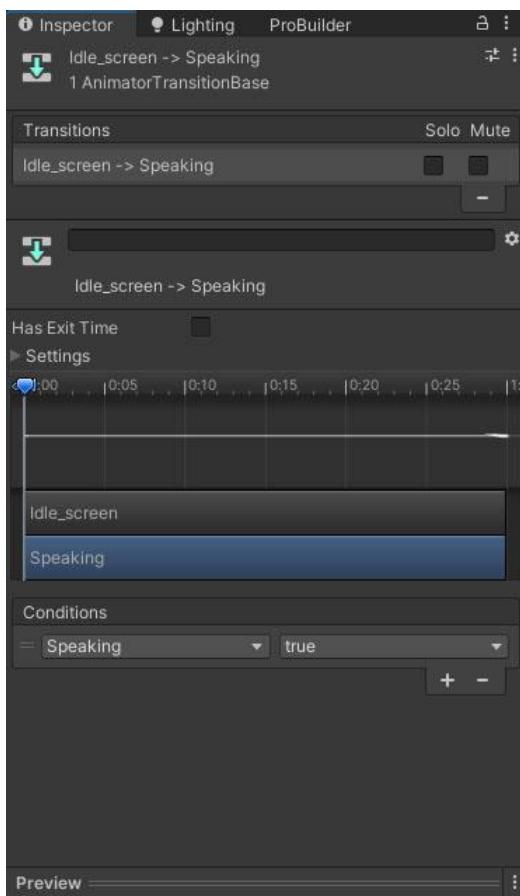
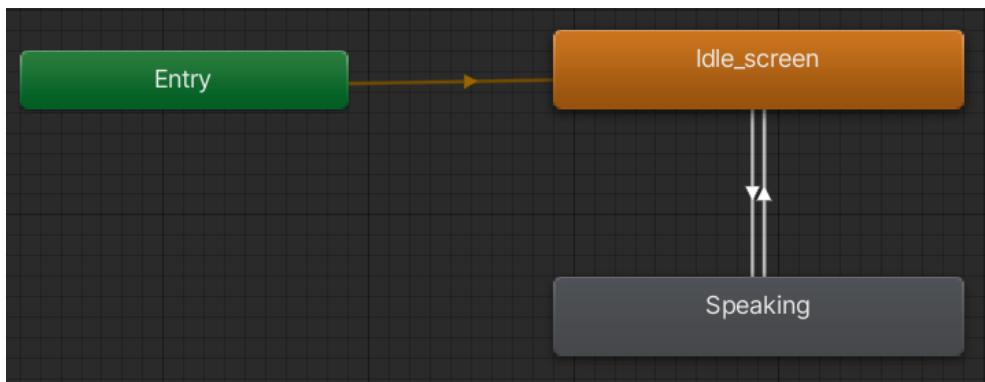
Le animazioni nel progetto sono semplici, perciò non necessito di suddividerle in più livelli. Ho solo bisogno di un parametro booleano che chiamo “Speaking” ed inizializzo a false. Questo parametro mi serve per indicare al controllore dell’animazione se il robot sta parlando o meno.

Il diagramma degli stati da implementare nella colonna a destra è molto semplice. Genero due stati associati alle animazioni create e collego lo stato di entry allo stato di idle per fare sì che l’applicazione passi in questo stato all’avvio del programma e poi inserisco due transizioni tra

---

<sup>20</sup> Unity Documentation, Animation States, <https://docs.unity3d.com/Manual/class-State.html>

gli stati di idle e di speaking per passare dal primo al secondo e viceversa.



Nella figura a lato c'è la transizione dallo stato di idle allo stato di speaking.

Nelle impostazioni si può impostare un periodo di transizione tra un'animazione e l'altra se si vuole che questa avvenga gradualmente. Io voglio che il cambiamento sia istantaneo quindi lo rimuovo.

La parte più importante dell'animazione è in fondo. Nella sezione delle condizioni si possono impostare i valori

che le variabili devono avere affinché avvenga la transizione. In questo caso la variabile “Speaking” deve essere vera.

La transizione dallo stato di speaking allo stato di idle è uguale ma la condizione è opposta.

## Realizzazione dell'icona

Voglio che l'icona del programma sia la testa del robot con il logo del politecnico sullo schermo. Per ottenere questo risultato devo estrarre l'immagine nella parte bassa dell'interfaccia.

Sul forum di Unity ho trovato uno script che aggiunto come componente ad una videocamera consente di esportare l'immagine ripresa in un dato momento.<sup>21</sup>

Lo script contiene due funzioni: una che serve ad esportare come file png la texture ripresa dalla videocamera al momento della propria chiamata e l'altra funzione è una funzione LateUpdate che fa sì che se l'utente preme il tasto F9 ad applicazione avviata si avvia la funzione che genera il file png.

Le funzioni LateUpdate sono funzioni eseguite in ogni frame in seguito alle funzioni Update.<sup>22</sup> Queste funzioni sono perfette per le telecamere perché in questo modo non possono effettuare delle operazioni con

---

<sup>21</sup> Unity Forum, How to save manually save a PNG of a camera view,  
<https://forum.unity.com/threads/how-to-save-manually-save-a-png-of-a-camera-view.506269/>

<sup>22</sup> Unity Documentation, MonoBehaviour.LateUpdate(),  
<https://docs.unity3d.com/ScriptReference/MonoBehaviour.LateUpdate.html>

l’immagine di un oggetto prima che un dato cambiamento sia avvenuto considerando che questo è generalmente nella funzione Update.<sup>23</sup>

Avvio l’applicazione, premo F9, ottengo l’immagine e la apro in Gimp. In questo programma la scalo per farle riempire la maggior parte dello spazio dell’immagine e la esporto. Al termine dell’esportazione il risultato ottenuto è il seguente.

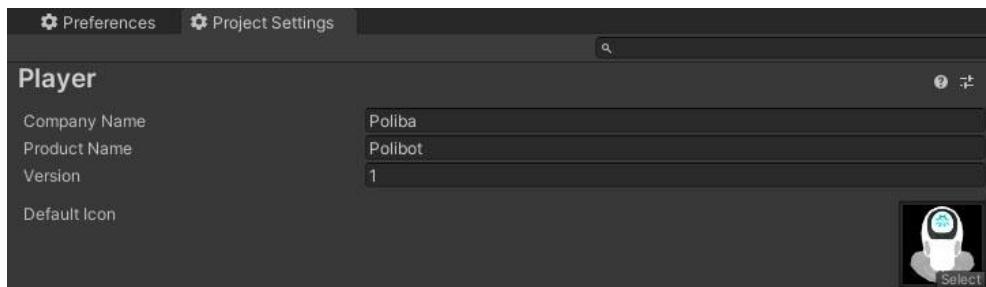


L’immagine è sgranata ai bordi per via del fatto che è stata scalata. Questo non è un problema perché quando diventerà l’icona dell’applicazione verrà visualizzata rimpicciolita e le sgranature non si noteranno.

---

<sup>23</sup> Unity Documentation, MonoBehaviour.Update(),  
<https://docs.unity3d.com/ScriptReference/MonoBehaviour.Update.html>

Ora che l'icona è pronta devo impostarla come icona dell'applicazione in Unity in Project Settings > Player > Default icon.



## **Configurazione di Rasa**

### **Spiegazione del funzionamento di Rasa**

Un chatbot è un software in grado di simulare la conversazione con un essere umano. I chatbot più comuni sono realizzati usando un grafo come quello di una macchina a stati e questo li rende facili da realizzare ma non li rende in grado di gestire input sconosciuti.

Un'evoluzione di un chatbot è un assistente contestuale come Rasa. Questi assistenti sono contestuali perché sono in grado di analizzare il contesto di una conversazione e ricavarne delle informazioni come farebbe un umano. Inoltre, usano l'intelligenza artificiale per analizzare gli input dell'utente consentendogli di conversare come vuole piuttosto che in maniera predeterminata.<sup>24</sup>

Dall'analisi degli input Rasa è in grado di evincere intenti, entità e valori.<sup>25</sup> Per comprendere meglio questi tre conviene fare degli esempi tratti dal progetto finito.

- 1) “Dove posso trovare l’aula A?”

---

<sup>24</sup> Sito ufficiale di Rasa, <https://rasa.com/>

<sup>25</sup> Rasa Docs, NLU Training Data, <https://rasa.com/docs/rasa/nlu-training-data/>

Da questa frase un umano evince che l'intento dell'utente è richiedere la posizione di una classe, nello specifico la classe A.

Da questa frase Rasa ricaverà:

- Intent = “richiesta\_aula”
  - Entity = “aula”
  - Value = “A”
- 2) “Come mi posso connettere ad Eduroam?”
- Intent = “richiesta\_rete”
  - Entity = “Eduroam”
  - Value = “Null”

In questo caso non serve che ci sia un valore perché non ci possono essere diverse istanze dell'entità dato che la rete del Politecnico è unica.

- 3) “Sto parlando con un robot?”
- Intent = “sfida\_bot”
  - Entity = “Null”
  - Value = “Null”

Nel caso in cui l'utente voglia sapere se sta conversando con un robot non serve che ci sia un'entità perché questa domanda non ha un oggetto di interesse, inoltre è una domanda a cui si può rispondere semplicemente in maniera affermativa o negativa.

Rasa, inoltre, consente l'utilizzo di azioni essenziali alla realizzazione del progetto.

Un'azione è un insieme di istruzioni, in questo caso in Python, che consentono allo sviluppatore di fare sì che Rasa possa eseguire dei compiti che vanno oltre la comune fornitura di una risposta.<sup>26</sup>

## Installazione di Rasa

Rasa si installa come un normale modulo di Python. Basta aggiornare pip e poi usarlo per installare la versione di Rasa che si desidera.<sup>27</sup>

```
python -m pip install --upgrade pip  
pip install rasa==2.6.1
```

Per fare sì che Rasa possa comprendere il linguaggio umano gli serve un file apposito relativo alla lingua desiderata. Questo file si può scaricare usando un modulo di Python chiamato “Spacy”.<sup>28</sup>

---

<sup>26</sup> Rasa Docs, Actions, <https://rasa.com/docs/rasa/actions/>

<sup>27</sup> Ibidem, Installation, <https://rasa.com/docs/rasa/installation>

<sup>28</sup> spaCy, Models & Languages, <https://spacy.io/usage/models>

```
pip install -U pip setuptools wheel  
pip install -U spacy  
python -m spacy download it_core_news_lg
```

Analizzando il nome del modello scaricato si capisce che è un modello italiano ricavato dalle notizie e di grosse dimensioni, quindi più dettagliato.

Nella cartella “Assets” del progetto creo la cartella “Rasa” ed apro al suo interno un prompt dei comandi.

In questo prompt digito il comando “Rasa init” e seleziono la cartella corrente come cartella di importazione dei file di Rasa.

Al termine dell’importazione nella cartella ci saranno i file necessari al funzionamento di un assistente molto semplice che chiede all’utente come si sente e se si sente triste gli fornisce un’immagine di un cucciolo di tigre.

La configurazione di Rasa consiste nella modifica di questi file per adattarli alle esigenze del progetto.

### **File “config.yml”**

Questo file contiene le impostazioni di Rasa. Molte di queste riguardano l’utilizzo dell’intelligenza artificiale e la modalità di analisi delle stringhe e vanno lasciate così come sono. Le uniche impostazioni da

modificare sono quelle sulla lingua, sul modello utilizzato e sulla distinzione tra lettere maiuscole e minuscole.<sup>29</sup>

## File “domain.yml”

Nel file in questione si possono impostare gli intenti, le entità, le azioni, le risposte ed il tempo massimo di inattività della sessione.<sup>30</sup>

Gli intenti del progetto sono:

- richiesta\_aula
- richiesta\_rete
- sfida\_bot
- nlu\_fallback

Questo intento riguarda i casi in cui Rasa non riesca a comprendere l'intento dell'utente, da qui il nome “Natural Language Understanding Fallback”.

- out\_of\_scope

Questo intento riguarda i casi in cui l'utente chieda cose non pertinenti o che il chatbot non conosce.

Le entità del progetto sono l'aula ed Eduroam come visto precedentemente negli esempi. L'unica azione del progetto serve a fare

---

<sup>29</sup> Rasa Docs, Model Configuration, <https://rasa.com/docs/rasa/model-configuration/>

<sup>30</sup> Idem, Domain, <https://rasa.com/docs/rasa/domain/>

sì che il robot fornisca informazioni riguardanti la posizione delle aule, perciò l'ho chiamata “action\_class\_answer”.

Le risposte in questo file sono quelle che non necessitano di elaborazione da parte delle azioni e sono relative a tutti gli intenti tranne le richieste dalla posizione delle aule.

Il tempo massimo di inattività della sessione è espresso in minuti ed impostato con un numero intero. Rasa ha bisogno di una sessione perché è un assistente contestuale e come tale deve tenere conto delle frasi precedenti, le quali costituiscono il contesto.

Nel progetto non c’è bisogno che Rasa usi le informazioni derivanti dal contesto perché le domande a cui può rispondere sono tutte indipendenti tra loro.

### **File “endpoints.yml”**

Il file “endpoints.yml” è un file contenente una serie di paragrafi commentati ciascuno relativo ad uno specifico servizio fornito da Rasa tramite un URL (da qui il nome “endpoints”).

Questo file è già compilato dagli sviluppatori di Rasa ed il suo utilizzatore non deve fare altro che decommentare i paragrafi relativi ad i servizi a cui è interessato ed eventualmente modificare i relativi URL.

Nel caso di questo progetto devo solo decommentare il paragrafo sulle azioni senza modificarne l’URL.

## File “rules.yml”

Questo file serve a stabilire le regole dell’andamento di una conversazione nei casi in cui lo si voglia imporre.<sup>31</sup> I casi in questione sono quelli in cui l’utente pronuncia frasi che Rasa non riesce a riconoscere o riconosce come non pertinenti ed i casi in cui l’utente chiede a Rasa se è un robot.

Ogni regola ha un nome ed un insieme di passaggi, ad esempio se si vuole che Rasa dichiari di essere un robot se l’utente lo richiede la regola da implementare è la seguente.

```
- rule: Dire di essere un bot quando l'utente lo chiede  
  steps:  
    - intent: sfida_bot  
    - action: utter_iamabot
```

Come si può vedere dall’esempio il nome della regola (specificato accanto alla parola chiave “rule”) può essere scelto arbitrariamente e contenere spazi. I passaggi indicano l’andamento della conversazione,

---

<sup>31</sup> Rasa Docs, Rules, <https://rasa.com/docs/rasa/rules/>

in questo caso se l’utente esprime l’intento “sfida\_bot” Rasa risponde con una delle frasi impostate per confermarlo.

Bisogna notare che in questo caso la parola chiave “action” non indica necessariamente un’operazione programmabile ma indica una generica operazione in senso lato, quindi anche la pronuncia di una frase preimpostata nel file “domain.yml”.

### **File “nlu.yml”**

Questo file serve per insegnare a Rasa a riconoscere gli intenti dell’utente fornendogli un insieme di esempi.<sup>32</sup> Per fare sì che ogni frase dell’utente venga compresa correttamente serve generare gruppi di esempi molto grandi contenenti tutte le frasi che si pensa che questo possa dire.

Ad esempio, per insegnare a Rasa a riconoscere l’intento dell’utente di sapere dove si trova un’aula bisogna compilare il file in questo modo:

---

<sup>32</sup> Rasa Docs, NLU Training Data, <https://rasa.com/docs/rasa/nlu-training-data/>

- *intent: richiesta\_aula*

*examples:* |

- *dove si trova l'aula [a]{“entity”: “aula”, “value”：“A”}?*

- *dov’è l’aula [a]{“entity”: “aula”, “value”：“A”}?*

- *dove posso trovare l’aula [a]{“entity”: “aula”, “value”：“A”}?*

- *come raggiungo l’aula [a]{“entity”: “aula”, “value”：“A”}?*

- *dove è l’aula [a]{“entity”: “aula”, “value”：“A”}?*

:

Questo estratto riguarda solo l’aula A, come si può vedere la richiesta della posizione di quest’ultima viene espressa in modi diversi ed in ogni caso la lettera che identifica l’aula viene racchiusa tra parentesi quadre. Accanto alle parentesi quadre, tra parentesi graffe, sono specificati il tipo di entità contenuta nella frase ed il suo valore. Dato che nel file “config.yml” ho specificato che Rasa non deve fare distinzione tra lettere maiuscole e minuscole è possibile specificare l’identificatore dell’aula sia in lettere maiuscole che in lettere minuscole.

Gli esempi riguardanti le altre aule riguardano le aule letterate dalla B alla S, l’aula AD, il centro linguistico e le aule numerate da 1 a 26.

Dato che la funzionalità di speech-to-text dell'applicazione potrebbe riconoscere le lettere ed i numeri in più modi e potrebbero esserci degli equivoci devo inserire degli esempi per poterli gestire. Ad esempio, la pronuncia della lettera Q potrebbe essere interpretata correttamente come “Q” oppure erroneamente come “qu” o “cu”. Questo vale anche per le pronunce dei numeri. La pronuncia del numero 1 potrebbe essere interpretata come “1” oppure come “uno”.

Per fare sì che Rasa riconosca correttamente gli intenti dell'utente c'è bisogno anche di indicare i possibili valori che può assumere un'entità ed i sinonimi di alcune parole usate.

I valori che può assumere un'entità vengono specificati nel seguente modo:

- *lookup: aula*

*examples: /*

- A

- B

- C

- D

- E

:

Il nome dell’entità è specificato nella prima riga accanto alla parola chiave “lookup” e dalla seconda riga in poi ci sono una serie di esempi.

I sinonimi invece vengono indicati nel seguente modo:

- *synonym*: *poliba*

*examples*: /

- *poliba*

- *politecnico*

- *politecnico di Bari*

- *università*

:

Il nome del sinonimo viene specificato nella prima riga accanto alla parola chiave “synonym” e dalla seconda riga in poi ci sono una serie di esempi. I sinonimi servono a rendere più flessibile il riconoscimento del linguaggio naturale per farlo adattare ai diversi modi in cui un utente può esprimere una frase.

## File “stories.yml”

Il file “stories.yml” serve ad indicare a Rasa come si vuole che venga gestito il flusso di una conversazione creando delle storie con una serie di passaggi.<sup>33</sup> Un estratto renderà tutto più chiaro e comprensibile.

```
stories:  
  - story: richiesta aula A  
  
steps:  
  - intent: richiesta_aula  
  
entities:  
  - aula: "A"  
  
  - action: action_class_answer  
  
  ...
```

Quella nell’estratto è la storia che spiega a Rasa il flusso della conversazione nel caso in cui l’utente richieda la posizione dell’aula A.

---

<sup>33</sup> Rasa Docs, Stories, <https://rasa.com/docs/rasa/stories/>

Il primo passaggio consiste nella richiesta dell’aula da parte dell’utente. Dato che il primo passaggio è un intento si può anche specificare un’entità ed il relativo valore.

Il secondo passaggio è un’azione che deve compiere Rasa, in questo caso è un’operazione programmabile ma potrebbe anche non esserlo. Indicando a Rasa che l’azione che deve compiere è “action\_class\_answer” questo passa allo script delle azioni in Python la coppia entità-valore.

La compilazione del file “stories.yml” richiede molto tempo perché bisogna inserire ogni singola richiesta che può fare l’utente per fare sì che le sue richieste ricevano la risposta desiderata.

### **File “test\_stories.yml”**

Questo file è simile al precedente. Anche qui si descrive il flusso di una conversazione indicandone i passaggi ma in questo caso piuttosto che indicare soltanto intento, entità, valore ed azione si inseriscono anche degli esempi di frasi in linguaggio naturale come fatto nel file “nlu.yml”.<sup>34</sup>

---

<sup>34</sup> Rasa Docs, Training Data Format, <https://rasa.com/docs/rasa/2.x/training-data-format/#test-stories>

- *story: Esempio richiesta aula A*

*steps:*

- *user: |*

*dove si trova l'aula [a]{"entity": "aula", "value": "A"}?*

*dove si trova la classe [a]{"entity": "aula", "value": "A"}?*

*intent: richiesta\_aula*

- *action: action\_class\_answer*

⋮

Le frasi in linguaggio naturale che può usare l’utente per esprimere il suo intento sono specificate nella voce “user” e l’estrazione dell’entità e del suo valore sono effettuate come nel file per la comprensione del linguaggio naturale.

## File “actions.py”

Questo file contiene lo script che esegue l’azione “action\_class\_answer”. Nella prima parte del file vengono importate le librerie necessarie al suo funzionamento. Quelle riguardanti Rasa vengono inserite nel file dal momento della sua creazione e non vanno toccate. Sotto queste un utente può inserire quelle che gli occorrono, io ho inserito la libreria “random” per selezionare a caso una risposta ad una domanda da un insieme di risposte.

Nella seconda parte del file ho inserito una tupla di risposte per ogni domanda. Per ora le risposte sono solo due ma in futuro se ne potrebbero inserire altre se non si vuole che sembri che il robot dica sempre le stesse frasi in maniera monotona.

```
responses_class_A =  
("L'aula A si trova al secondo piano dell'edificio che si  
affaccia sull'atrio Cherubini, lato ovest.", "L'aula A è al  
secondo piano dell'edificio che si affaccia sull'atrio  
Cherubini, lato ovest.")
```

Questo estratto riguarda la definizione della tupla delle risposte alla domanda “Dove si trova l’aula A?”.

Le tuple delle aule da 1 a 20 contengono risposte che dicono all’utente di raggiungere il secondo piano dell’edificio sovrastante l’atrio con le sculture e poi seguire le indicazioni perché al piano in questione si può accedere da più ingressi. Non sapendo da quale ingresso entrerà l’utente

non posso dirgli di andare in una determinata direzione ma devo limitarmi a dirgli di seguire le indicazioni.

Dopo aver definito tutte le tuple di tutte le risposte a tutte le domande devo generare un dizionario che associa i valori dell'entità “classe” alle tuple delle risposte.

```
responses = {"A":responses_class_A,  
             "B":responses_class_B,  
             "C":responses_class_C,  
             "D":responses_class_D,  
             :}
```

Infine il file contiene la classe “ActionClassAnswer“ in cui dato il valore dell'entità “classe” si seleziona una tupla dal dizionario delle risposte e si sceglie in maniera casuale un suo elemento. Scelto l'elemento lo si fornisce in output. Se il valore dell'entità non è una delle chiavi del dizionario il messaggio di output è "La classe richiesta non esiste.".

## Generazione del modello

Il modello di Rasa è un file generato a seguito dell'elaborazione di tutti i file compilati in precedenza.

Per generarlo bisogna aprire un prompt dei comandi nella cartella in cui è stato installato Rasa e digitare il comando “rasa train”. Ad elaborazione conclusa comparirà un nuovo file nella cartella “models”. Dato che Rasa usa il file del modello più recente se dovessero esserci altri file questi si possono anche cancellare.<sup>35</sup>

## Prova del modello

Il modello appena generato si può provare aprendo due shell e digitando in una il comando “rasa shell” e nell’altra il comando “rasa run actions”.

Il primo comando consente di dialogare con Rasa da linea di comando ed il secondo comando avvia il server delle azioni che usa il file “actions.py” per fornire le risposte alle richieste riguardanti le classi.

Dopo che entrambi i server sono stati avviati, nella shell contenente la shell di Rasa è possibile chiedere informazioni e verificare che tutto funziona correttamente.

---

<sup>35</sup> Rasa Docs, Command Line Interface, <https://rasa.com/docs/rasa/command-line-interface>

## Attivazione della funzione speech-to-text

Per utilizzare la funzione speech-to-text nell'applicazione occorre attivare il riconoscimento vocale su Windows. In Windows 10 per raggiungere l'impostazione bisogna andare in Impostazioni > Privacy > Comandi vocali. In questo menù bisogna Attivare il riconoscimento vocale online.

## Comandi vocali

### Riconoscimento vocale online

Usa la voce per la dettatura e altre app tramite la tecnologia di riconoscimento vocale online di Microsoft.

Quando questa impostazione è disattivata, puoi comunque usare l'app Riconoscimento vocale di Windows o altre app per i comandi vocali che non dipendono da questa impostazione.



Attivato

Attivare il riconoscimento vocale in questo modo va bene in fase di sviluppo dell'applicazione ma chiedere all'utente finale di effettuare lo stesso procedimento sarebbe inadeguato, sarebbe più comodo se questa impostazione venisse modificata in automatico.

Per farlo bisogna sapere che questa impostazione è legata ad una variabile del registro di sistema, nello specifico la variabile “HasAccepted” collocata in:

HKEY\_CURRENT\_USER\Software\Microsoft\Speech\_OneCore\Settings\OnlineSpeechPrivacy

Questa informazione sarà essenziale nella fase di sviluppo dell’installer perché mi consentirà di scrivere il file che modifica la voce di registro in questione.<sup>36</sup>

---

<sup>36</sup> Appuals, How to Enable / Disable Online Speech Recognition in Windows 10?,  
<https://appuals.com/enable-disable-online-speech-recognition-in-windows-10/>

## **Attivazione di una voce per la funzionalità text-to-speech**

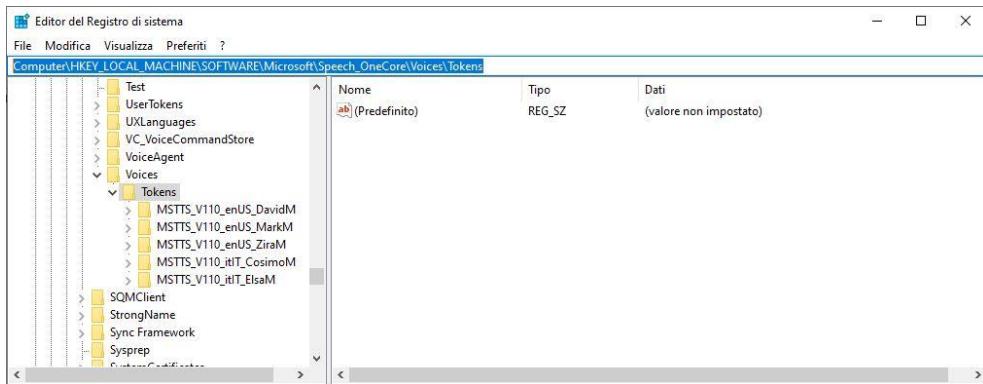
Windows possiede un insieme di voci però solo alcune di queste possono essere usate dai programmi di terze parti. La voce dell'applicazione si chiama “Microsoft – Cosimo” e purtroppo fa parte delle voci utilizzabili solo da programmi interni a Windows.

È possibile utilizzare la voce in questione, così come le altre riservate ad uso interno, per i programmi di terze parti modificando il registro di sistema.<sup>37</sup>

Aprendo l'editor del registro di sistema ed andando nel percorso “Computer\HKEY\_LOCAL\_MACHINE\SOFTWARE\Microsoft\Speech\_OneCore\Voices\Tokens” si noterà che la cartella “Tokens” ha un insieme di sottocartelle. Ciascuna di queste è relativa ad una voce e tra di esse si vede anche quella relativa a Cosimo.

---

<sup>37</sup> Ghacks.net, Unlock all Windows 10 TTS voices system-wide to get more of them, <https://www.ghacks.net/2018/08/11/unlock-all-windows-10-tts-voices-system-wide-to-get-more-of-them/>



Per utilizzare la voce “Microsoft – Cosimo” per programmi di terze parti devo esportare la sua cartella come file del registro del sistema e modificarlo. Chiamo il file in questione “voce\_Cosimo.reg”.

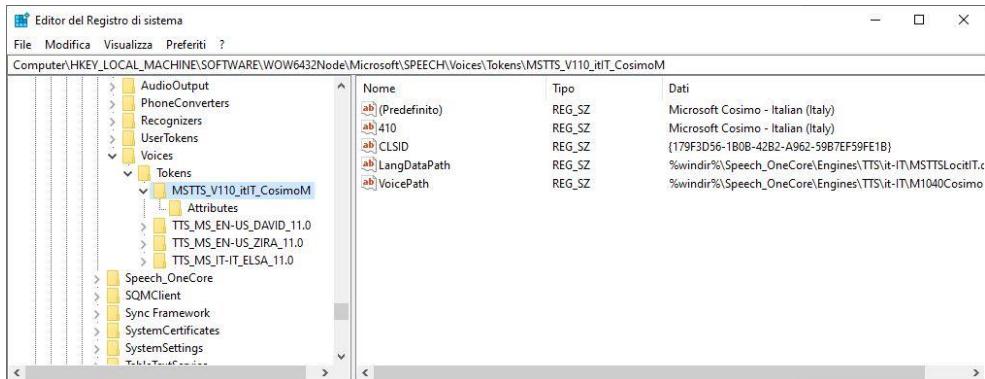
Il file esportato è costituito da una prima riga contenente la versione del registro di sistema, una riga vuota e due paragrafi: il primo relativo alla cartella principale della voce ed il secondo alla sua sottocartella degli attributi. Questi due paragrafi servono a dare a Windows le informazioni su come usare la voce con i programmi interni. Per fargli usare la voce anche con i programmi di terze parti bisogna copiare i due paragrafi ed incollarli in fondo al file. Fatto ciò, bisogna modificare la prima riga di ciascun paragrafo per aggiungere delle cartelle al registro di sistema.

I percorsi da assegnare rispettivamente a ciascun paragrafo sono:

1. HKEY\_LOCAL\_MACHINE\SOFTWARE\Microsoft\Speech\Voice\Tokens\MSTTS\_V110\_itIT\_CosimoM

2. HKEY\_LOCAL\_MACHINE\SOFTWARE\Microsoft\Speech\Voices\Tokens\MSTTS\_V110\_itIT\_CosimoM\Attributes
3. HKEY\_LOCAL\_MACHINE\SOFTWARE\WOW6432Node\Microsoft\SPEECH\Voices\Tokens\MSTTS\_V110\_itIT\_CosimoM
4. HKEY\_LOCAL\_MACHINE\SOFTWARE\WOW6432Node\Microsoft\SPEECH\Voices\Tokens\MSTTS\_V110\_itIT\_CosimoM\Attributes

Compilato il file basta eseguirlo per inserire le cartelle nel registro di sistema ed utilizzare la voce di Cosimo per i programmi di terze parti.



Il procedimento appena effettuato si basa su un'assunzione implicita, cioè quella di usare una versione di Windows in italiano con le voci italiane preinstallate al suo interno. Questa assunzione è valida nel caso di questa applicazione perché è interna al Politecnico di Bari. Se si dovesse installare l'applicazione su una versione di Windows in un'altra lingua bisognerebbe scaricare la voce di Cosimo tramite l'apposito menù in Impostazioni > Accessibilità > Assistente vocale.

## Personalizza la voce dell'Assistente vocale

Scegli una voce

Microsoft Cosimo - Italian (Italy) ▾

[Aggiungi altre voci](#)

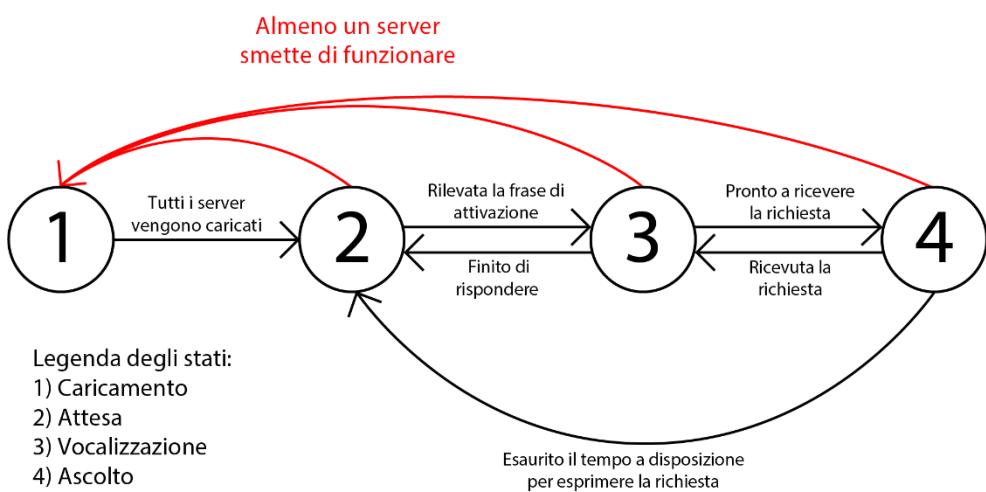
Installare Polibot su una versione di Windows senza la voce di Cosimo creerebbe correttamente le cartelle nel registro di sistema ma la voce in questione sarebbe inutilizzabile data la mancanza degli appositi file.

Infatti, se si osserva tra le variabili della cartella principale della voce si noterà la presenza della variabile “VoicePath” che indica il percorso del file di sintesi vocale utilizzato dalla voce.

## Codice

### Stati dell'applicazione

Il funzionamento dell'applicazione è rappresentato nel seguente automa a stati finiti.



Gli stati dell'applicazione sono:

#### 1. Caricamento:

In questo stato l'applicazione aspetta che vengano avviati i server di Rasa e di sintetizzazione vocale. Durante l'attesa l'utente non può comunicare con Polibot e viene mostrato un messaggio di caricamento.

#### 2. Attesa:

Quando l'applicazione si trova in questo stato attende che l'utente dica la frase di attivazione.

3. Vocalizzazione:

Questo stato è relativo ai momenti in cui Polibot parla all'utente.

4. Ascolto:

In questo stato Polibot ascolta la richiesta dell'utente.

Le transizioni tra gli stati sono:

- Da 1 a 2:

Questa transizione avviene quando vengono avviati tutti i server necessari al funzionamento dell'applicazione.

- Da 2 a 3:

Questo passaggio di stato avviene quando l'applicazione rileva che l'utente ha detto la frase di attivazione.

- Da 3 a 4:

Polibot ha comunicato all'utente di essere in ascolto e si prepara ad ascoltare la sua richiesta.

- Da 4 a 3:

Polibot ha ricevuto la richiesta dell'utente, ha elaborato una risposta e si prepara per vocalizzarla.

- Da 3 a 2:

Polibot ha finito di vocalizzare la risposta e torna nello stato di ascolto per ricevere un'altra richiesta.

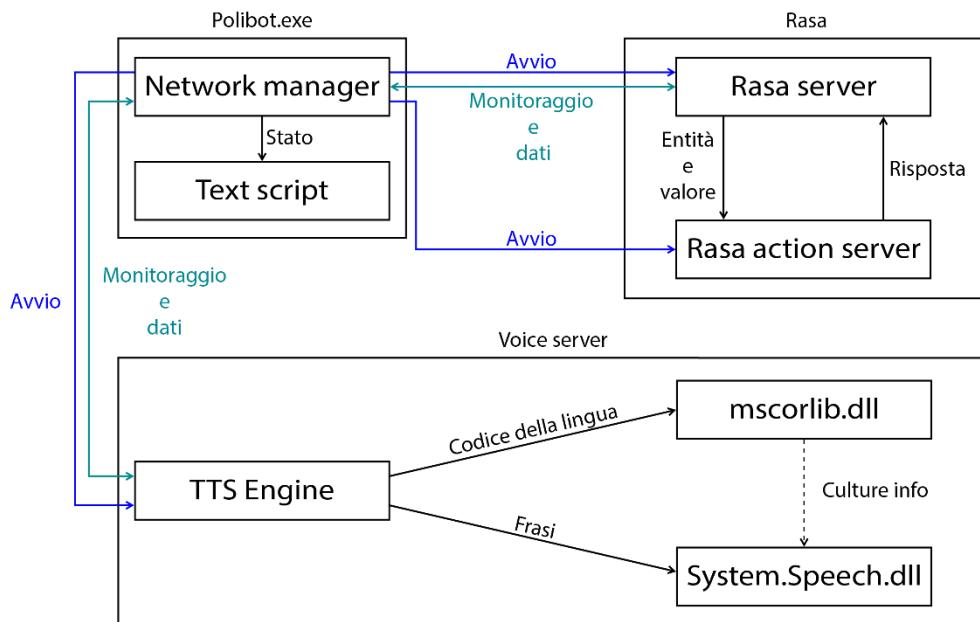
- Da 4 a 2:

Questa transizione avviene se l’utente non comunica la sua domanda entro un tempo prestabilito. Sebbene questo passaggio di stato possa sembrare fastidioso per gli utenti che non pronunciano in tempo la loro domanda questa transizione è essenziale per il corretto funzionamento dell’applicazione. Senza di questa Polibot potrebbe essere lasciato in stato di ascolto indefinitamente e per farlo uscire da questo stato bisognerebbe chiudere l’applicazione!

- Transizioni forzate allo stato 1:

Queste avvengono se almeno uno dei server a cui si affida l’applicazione smettono di funzionare. Una volta che questa torna nello stato 1 avviene il riavvio dei server non funzionanti.

## Schema di funzionamento dell'applicazione



## Legenda



All'interno del programma “Polibot.exe” sono presenti uno script principale ed uno subordinato. La suddivisione dei compiti tra i diversi script è essenziale per mantenere un progetto in ordine.

Lo script principale è “NetworkManager.cs”. Questo script non solo si occupa dello scambio di dati tramite la rete ma si occupa anche di attivare e monitorare lo stato dei server. I server vengono avviati usando il prompt dei comandi e quando sono operativi interagiscono con lo

script principale tramite delle richieste web. Lo scambio di dati avviene con il metodo POST ed il monitoraggio è realizzato con richieste periodiche con il metodo GET.

Affinché sia possibile monitorare lo stato di un server serve che questo fornisca un servizio di hello. Il server principale di Rasa lo fornisce ma quello delle azioni no, pertanto, non è possibile monitorare lo stato di quest'ultimo.<sup>38</sup>

Ho implementato un servizio di ricezione dei messaggi con il metodo POST ed un servizio di hello con il metodo GET nel file “TTS\_Engine.py” per rendere l’interazione tra questo e lo script principale analoga all’interazione con Rasa.

Lo script per il servizio di vocalizzazione interagisce con due librerie di Windows: System.Speech.dll e mscorelib.dll. L’utilizzo della prima libreria consiste nella creazione di un oggetto per la sintesi vocale e nella scelta di una voce da usare per la vocalizzazione dei messaggi. La selezione della voce avviene in più tentativi. Il primo è il più diretto perché si prova a selezionare la voce tramite il suo nome. Il secondo tentativo consiste nel selezionare una voce maschile italiana e se questo dovesse fallire lo script prova a selezionare una voce italiana senza

---

<sup>38</sup> Medium, Integrating Rasa Open Source Chatbot Into Unity [Part 1] : The Connection, Divyang Pradeep Pal, <https://medium.com/analytics-vidhya/integrating-rasa-open-source-chatbot-into-unity-part-1-the-connection-9ba582c804cd>

specificarne il genere. Nel caso in cui tutti questi tentativi dovessero fallire la vocalizzazione avviene con la voce di default di Windows.

Per il secondo e terzo tentativo è necessario indicare la nazionalità della voce. Per questo compito è necessaria la seconda libreria che è in grado di generare un oggetto appartenente alla classe “CultureInfo” istanziato usando il codice della lingua italiana. Dato che è ragionevole assumere che l’applicazione verrà installata su computer che utilizzano la lingua italiana e che l’utilizzo della voce di Cosimo verrà reso possibile in fase di installazione, il primo tentativo dovrebbe avere successo, pertanto, l’interazione tra gli oggetti creati usando la seconda libreria e la prima libreria è solo una possibilità.

Lo script subordinato “TextScript.cs” riceve dallo script principale lo stato in cui si trova il programma e gestisce il testo sull’interfaccia di conseguenza.

### **File “TTS\_Engine.py”**

Questo file serve per usare la libreria di Windows per la vocalizzazione. L’utilizzo di Python in realtà è uno stratagemma per aggirare un problema di Unity. Questo purtroppo non consente di utilizzare tutte le librerie del .NET Framework ma solo alcune e purtroppo System.Speech.dll non è tra queste. L’utilizzo di Python comporta un utilizzo maggiore di RAM e CPU di quello che si avrebbe usando la libreria all’interno di uno script in C#, pertanto, uno dei possibili

sviluppi futuri del progetto consiste nell'implementazione della funzionalità di vocalizzazione all'interno di Unity.

Il file comincia con l'importazione delle librerie necessarie alla realizzazione ed all'utilizzo di un mutex, cioè un oggetto detenuto da Windows a cui è assegnata un'etichetta che lo identifica univocamente. Windows fa sì che possa esistere una ed una sola istanza di questo oggetto e se si prova a crearne un'altra si riceve un errore.<sup>39</sup> Lo script gestisce l'eccezione relativa a questo errore terminando la sua esecuzione. In questo modo ho sviluppato uno script in grado di avere una sola istanza in esecuzione.

A seguito del controllo sull'unicità dell'istanza importo le altre librerie necessarie al funzionamento dello script. Se avessi importato tutte le librerie all'inizio lo script avrebbe impiegato più tempo a caricare ed a verificare l'unicità dell'istanza ed in caso di istanze multiple avrei occupato più memoria RAM inutilmente.

Importato tutto l'occorrente procedo ad istanziare gli oggetti che mi occorrono per la sintetizzazione vocale ed a selezionare la voce da utilizzare come descritto in precedenza.

---

<sup>39</sup> Microsoft Docs, Mutex Objects, <https://docs.microsoft.com/en-us/windows/win32/sync/mutex-objects>

A questo punto devo solo implementare i metodi per l’implementazione del server ed avviarlo.

Il metodo per la risposta alle richieste effettuate con il metodo POST è il seguente:

```
def do_POST(self):
    # Response to the sender
    self.send_response(200)
    self.send_header('Content-type','text/html')
    self.end_headers()
    message = str("OK")
    self.wfile.write(bytes(message, "utf8"))

    # Read the message and make the voice speak
    content_length = int(self.headers['Content-Length'])
    post_data = self.rfile.read(content_length)
    # post_data is a string of bytes and has to be decoded
    post_data = post_data.decode('utf-8')
    post_dict = json.loads(post_data)
    message = post_dict['message']
    synth.SpeakAsync(message)

return
```

La prima parte del metodo serve a fornire allo script “NetworkManager.exe” un feedback positivo inviando la stringa “OK”. La seconda parte del metodo serve ad estrarre il contenuto del messaggio e vocalizzarlo. È importante notare la penultima riga dell’estratto in cui si legge che l’oggetto per la sintesi vocale vocalizza

il messaggio ricevuto in maniera asincrona per evitare di fermare l'esecuzione dello script nel processo.

Il metodo per il monitoraggio è il seguente:

```
def do_GET(self):
    # Specifichiamo il codice di risposta
    self.send_response(200)
    # Specifichiamo uno o più header
    self.send_header('Content-type','text/html')
    self.end_headers()

    # Specifichiamo il messaggio che costituirà il corpo della risposta
    message = str(synth.State == SynthesizerState.Speaking)
    self.wfile.write(bytes(message, "utf8"))

    return
```

In questo metodo si confronta lo stato attuale del sintetizzatore con il valore dello stato enumerato che indica che questo sta parlando e poi si converte il risultato del confronto in stringa. Questa stringa poi verrà inviata al Network Manager che la userà non solo per verificare l'operatività del server ma anche per conoscere lo stato della voce.

L'ultimo metodo da definire è quello per avviare il server:

```
def run():
    print('Starting voice server...')
    server_address = ('127.0.0.1', 8081)
    httpd = HTTPServer(server_address, voice_RequestHandler)
    print('Running voice server...')
    httpd.serve_forever()
```

Questo metodo avvia il server in locale e lo mette in ascolto indefinitamente sulla porta 8081.

### File “NetworkManager.cs”

All’inizio di questo file vengono importare le librerie necessarie al suo funzionamento e vengono dichiarate le classi per agevolare lo svolgimento delle operazioni. Ad esempio, una di queste classi serve a contenere il messaggio da inviare a Rasa ed in seguito verrà convertita prima in formato JSON e poi in sequenza di bytes per poi essere inserita in un pacchetto ed inviata tramite la rete.

La classe principale dello script è la classe NetworkManager. La classe principale di ogni script in Unity deve avere lo stesso nome del file altrimenti nella console dell’editor si riceve un messaggio di errore.

La classe in questione è una sottoclasse di MonoBehaviour. Questa classe è la classe di base di tutti gli script di Unity.<sup>40</sup>

All'inizio di questa classe, nella sezione dichiarativa, vengono dichiarate le stringhe contenenti gli URL dei server, il programma da usare per i prompt, la frase di attivazione e la sua risposta e le stringhe contenenti le stringhe dei comandi.

Ad esempio, la stringa per l'avvio del server di Rasa è la seguente:

```
private const string server_command =  
    "Set-ExecutionPolicy Unrestricted -Scope Process -force;  
    Polibot_venv/Scripts/activate.ps1; cd Assets/Rasa; rasa run";
```

Innanzitutto, la stringa viene dichiarata come privata e costante perché non serve che sia pubblica e non va modificata in nessun momento. La stringa vera e propria comincia con un comando di impostazione delle politiche di esecuzione che serve a consentire al prompt di eseguire lo script “activate.ps1” autonomamente. Senza questo comando verrebbe mostrata la finestra in cui si chiede all’utente di eseguire lo script con i privilegi di amministratore. La stessa finestra che viene mostrata quando si avvia un programma di installazione. Lo script “activate.ps1” serve ad attivare l’ambiente virtuale all’interno

---

<sup>40</sup> Unity Documentation, MonoBehaviour,  
<https://docs.unity3d.com/ScriptReference/MonoBehaviour.html>

della shell. Eseguendo questo script in un’istanza di Powershell avviata normalmente si nota che il nome dell’ambiente virtuale racchiuso tra parentesi tonde viene anteposto alla normale riga in cui si inseriscono i comandi. Il terzo ed il quarto comando servono rispettivamente a spostarsi nella cartella di Rasa ed avviare il server.

Tra le altre variabili dichiarate nella sezione dichiarativa ci sono le variabili booleane relative allo stato dei server e della voce. Queste variabili sono inizializzate a false perché quando lo script viene avviato presumibilmente il resto dei componenti deve essere ancora avviato.

Per lo stato dell’applicazione uso un numero intero e lo inizializzo ad 1 perché all’avvio dello script l’applicazione si trova probabilmente in questo stato.

Le condizioni iniziali dell’applicazione non sono garantite perché i server potrebbero essere già avviati al momento dell’avvio dello script o manualmente dall’utente oppure da un’istanza precedente dell’applicazione.

Dato che questo script interagisce con il suo script subordinato c’è bisogno di un suo riferimento. È buona norma ottenere il riferimento ad

un altro script nella funzione Awake che è la prima ad essere eseguita una volta caricato lo script in memoria.<sup>41</sup>

```
//Reference to the UI script
private TextScript textscript;
[SerializeField] GameObject TextManager;

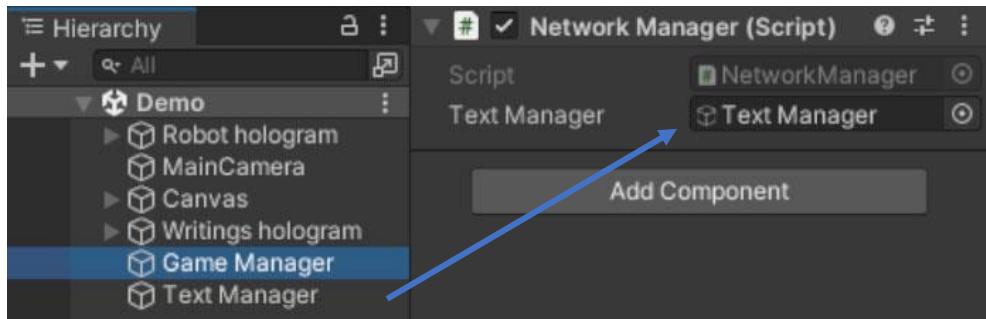
void Awake()
{
    textscript =
    TextManager.GetComponent<TextScript>();
}
```

Lo script è un componente dell’oggetto TextManager. Per fare sì che lo script NetworkManager.cs abbia un riferimento all’oggetto in questione è necessario che la variabile relativa ad esso venga dichiarata come “[SerializeField]”.<sup>42</sup> Così facendo è possibile selezionare un oggetto dalla gerarchia e trascinarlo su un campo dello script pur mantenendo la variabile privata.

---

<sup>41</sup> Unity Documentation, MonoBehaviour.Awake(),  
<https://docs.unity3d.com/ScriptReference/MonoBehaviour.Awake.html>

<sup>42</sup> Unity Documentation, SerializeField,  
<https://docs.unity3d.com/ScriptReference/SerializeField.html>



Nell'immagine viene mostrato che l'oggetto Text Manager a cui è assegnato lo script subordinato viene passato ad un campo dello script principale assegnato all'oggetto Game Manager.

Nella funzione start vengono avviati i server ed istanziati gli oggetti per il riconoscimento vocale.

```

void Start()
{
    StartCoroutine(startRasa());
    StartCoroutine(startVoiceServer());

    actions.Add(activationPhrase.ToLower(),
    detectedActivationPhrase); //The text is recognized in lower case
    keywordRecognizer = new KeywordRecognizer(actions.Keys.ToArray());
    keywordRecognizer.OnPhraseRecognized +=
        KeywordRecognizer_OnPhraseRecognized;

    dictationRecognizer = new DictationRecognizer();
    dictationRecognizer.DictationResult +=
        DictationRecognizer_DictationResult;
    dictationRecognizer.DictationComplete +=
        DictationRecognizer_DictationComplete;
}

}

```

Le prime due istruzioni della funzione servono ad avviare i server usati dall'applicazione.

Le istruzioni contenute in una funzione vengono eseguite nella finestra temporale di un frame. Questa finestra temporale è insufficiente per l'avvio di un server e l'attesa che questo sia operativo, perciò serve utilizzare una coroutine, cioè una funzione che consente di suddividere

un insieme di operazioni tra diversi frame.<sup>43</sup> Per scandire le operazioni da eseguire all'interno di un frame e per restituire il controllo a Unity bisogna usare l'istruzione “yield return” seguita da un oggetto. Se si vuole solo restituire il controllo a Unity questo oggetto deve essere “null”, ma si possono anche effettuare altre operazioni. Ad esempio, per fare aspettare una coroutine che monitora sullo stato di un server uso la seguente istruzione:

```
yield return new WaitForSeconds(5);
```

In questo modo rendo lo script più performante perché nel tempo che intercorre tra una richiesta e la sua successiva la coroutine non viene eseguita.

Nella funzione “Start” è presente solo una coroutine per l'avvio di Rasa perché la coroutine che avvia il server principale avvia anche quello delle azioni.

Gli oggetti “keyword recognizer” e “dictation recognizer” sono entrambi relativi alla funzione speech-to-text. Il primo può rimanere in ascolto indefinitamente in attesa che l'utente pronunci l'activation phrase<sup>44</sup> mentre l'altro rimane attivo per un tempo limitato ed ascolta

---

<sup>43</sup> Unity Documentation, Coroutines, <https://docs.unity3d.com/Manual/Coroutines.html>

<sup>44</sup> Ibidem, KeywordRecognizer,  
<https://docs.unity3d.com/ScriptReference/Windows.Speech.KeywordRecognizer.html>

un input vocale non predeterminato.<sup>45</sup> Questi due oggetti non possono essere attivi simultaneamente perché entrambi usano il microfono ed hanno scopi diversi. Data una frase pronunciata dall’utente la libreria per il riconoscimento vocale deve sapere da quale oggetto farla analizzare.

Il keyword recognizer si basa su una mappa (o dizionario) in cui le chiavi sono le frasi da riconoscere ed i valori sono le funzioni da eseguire per ciascuna frase.

La mappa in questione si chiama “actions”. A questa viene aggiunta la coppia (activation phrase, detectedActivationPhrase) e poi viene istanziato l’oggetto keyword recognizer passandogli le chiavi della mappa. La pronuncia di una frase da parte dell’utente è a tutti gli effetti un evento e va trattato come tale, per questo motivo l’ultima istruzione del gruppo relativo al keyword recognizer serve a mettere questo oggetto in ascolto di (iscrivere a) questo evento.

La sezione del dictation recognizer lo istanzia e lo iscrive agli eventi relativi al completamento della dettatura ed all’elaborazione del suo risultato.

L’interazione tra questi due oggetti consente la realizzazione di un assistente vocale simile a quello installato sui telefoni Android. Questo

---

<sup>45</sup> Unity Documentation, DictationRecognizer,  
<https://docs.unity3d.com/ScriptReference/Windows.Speech.DictationRecognizer.html>

è sempre in ascolto e pronunciando la activation phrase “OK Google” si attiva ed è possibile dare un comando vocale.

In maniera simile, per utilizzare Polibot bisogna dire l’activation phrase “Ehi Polibot”. Quando il keyword recognizer rileva questa frase Polibot conferma di aver ricevuto il messaggio dicendo “Ti ascolto”. A questo punto l’utente deve formulare una richiesta che verrà ascoltata dal dictation recognizer. Questa richiesta verrà inviata a Rasa, elaborata, restituita allo script in questione e vocalizzata.

L’interazione con i server avviene tramite due coroutine: una per l’avvio ed una per il monitoraggio.

La coroutine per l’avvio apre il prompt dei comandi relativo al server desiderato ed effettua delle richieste al servizio di hello finché non ne riceve una che conferma l’operatività del server. Dopo aver verificato l’operatività del server avvia la coroutine di monitoraggio che effettua periodicamente delle richieste al servizio di hello e se una di queste fallisce chiude il prompt dei comandi del server in questione ed avvia la sua coroutine di avvio. In questo modo ho sviluppato un sistema di avvio, controllo e ripristino dei server.

Uno dei problemi principali che ho dovuto risolvere in questo script è il cambiamento di stato. Dato che questo cambia non solo in base all’operatività dei server ma anche in base allo stato di vocalizzazione del sintetizzatore vocale. Questo script aveva il problema di cambiare stato prima che il sintetizzatore vocale finisse di vocalizzare le frasi. Per

risolvere il problema ho utilizzato due coroutine una per la vocalizzazione e l'altra per il cambiamento di stato.

La coroutine per la vocalizzazione invia la frase al server per la vocalizzazione e finché questo non comincia a vocalizzare rimane in attesa. Quando parte la voce la coroutine avvia la coroutine per il cambio di stato. Per evitare che quest'ultima possa modificare lo stato prima che la frase finisca rimane in attesa che la variabile “speaking” venga impostata a “false” poi procede a modificare lo stato dell'applicazione apportando le relative modifiche all'utilizzo degli oggetti per il riconoscimento vocale ed infine trasmette il cambio di stato allo script subordinato.

### **File “TextScript.cs”**

Questo script contiene solo la classe principale derivata da MonoBehaviour e nella sezione dichiarativa contiene la lista degli elementi delle varie scritte dichiarati come [SerializeField] ed il riferimento al controllore dell'animazione.

Ha una funzione Awake in cui si collega allo script principale in maniera analoga a quella vista in precedenza e una funzione “setState” che viene usata dallo script principale per modificare lo stato nello script in questione.

La gestione delle scritte sull'interfaccia e l'interazione con il controllore dell'animazione avvengono nella funzione “Update”. Questa contiene

uno switch che usa il numero dello stato come parametro e decide quale scritta deve essere visibile e quale no. Il caso più interessante è il primo perché nello stato 1 la scritta non è fissa.

*case 1:*

```
    animator.SetBool("Speaking", false);
    loading_text.SetActive(true);
    idle_instruction_text.SetActive(false);
    listening_instruction_text.SetActive(false);

    timeDiff = Time.time - startTime;
    if (timeDiff < timeStep)
    {
        Dot_1.SetActive(false);
        Dot_2.SetActive(false);
        Dot_3.SetActive(false);
    }
    else if (timeDiff >= timeStep && timeDiff < 2 * timeStep)
    {
        Dot_1.SetActive(true);
    }
    else if (timeDiff >= 2 * timeStep && timeDiff < 3 * timeStep)
    {
        Dot_2.SetActive(true);
    }
    else if (timeDiff >= 3 * timeStep && timeDiff < 4 * timeStep)
    {
        Dot_3.SetActive(true);
    }
    else
    {
        startTime = Time.time;
    }
    break;
```

La prima istruzione serve a comunicare all'animatore che Polibot non sta parlando. Le istruzioni dalla seconda alla quarta servono a stabilire quale scritta deve essere visibile e quale no, in questo caso la scritta attiva è quella relativa al caricamento.

Le istruzioni che seguono servono a stabilire quali puntini sono visibili e per farlo usa il tempo trascorso dal momento in cui non è visibile alcun puntino. Il tempo iniziale dello script viene impostato nella funzione start e viene reimpostato nel caso 1 dopo che tutti i puntini sono diventati visibili.

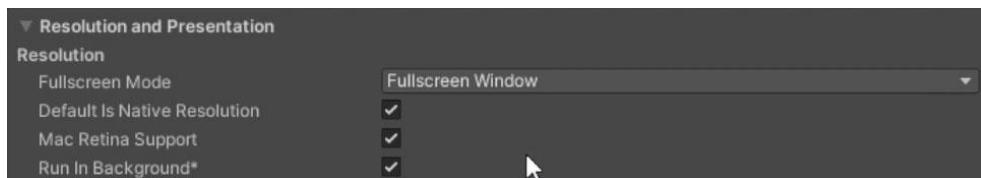
Affinché questo algoritmo funzioni correttamente serve stabilire un intervallo temporale che scandisca la comparsa e la scomparsa dei puntini. Questo intervallo temporale è contenuto nella variabile “timeStep” ed è inizializzato ad 1 secondo nella sezione dichiarativa dello script.

Dato che i tre puntini sono elementi figli della scritta di caricamento, all'interno degli altri casi basta impostare quest'ultima come inattiva per disattivare anche i puntini.

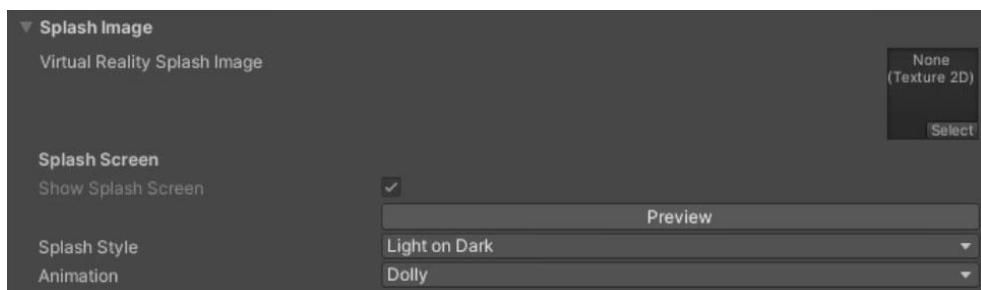
## Esportazione dell'applicazione in Unity

Prima di esportare il progetto bisogna impostare dei parametri in Project Settings > Player (Stesso menù dell'icona).

Bisogna impostare il nome della compagnia, il nome del prodotto e la sua versione come visto nell'immagine precedente, inoltre bisogna impostare la modalità a schermo intero nella sezione “Resolution and Presentation”.

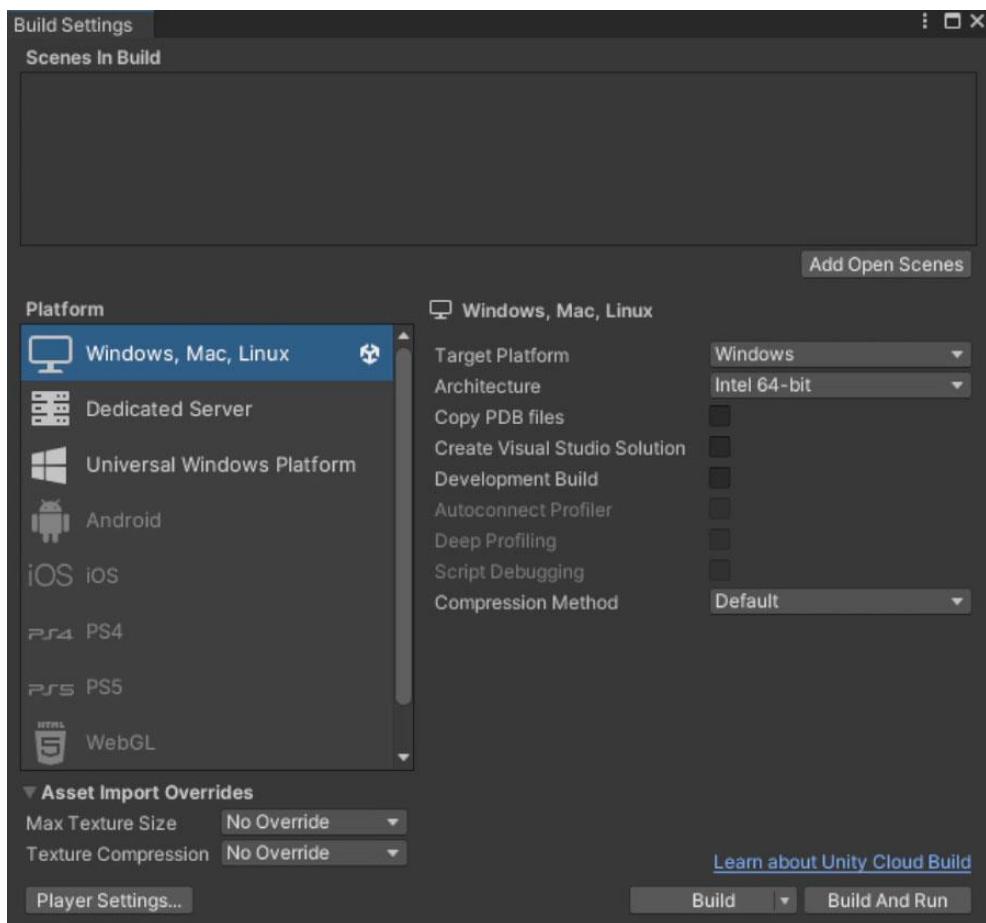


Sotto la sezione corrente c’è la sezione “Splash Image” in cui si può impostare l’immagine mostrata all’avvio dell’applicazione. In questo menù scelgo lo stile “Light on Dark” e l’animazione “Dolly”. È possibile avere un’anteprima dell’effetto che si otterrà nell’applicazione con l’apposito pulsante.



Lascio il resto dei parametri così come sono perché non mi serve modificarli.

Adesso sono pronto per esportare l'applicazione. Per farlo bisogna andare in File > Build Settings e si aprirà la finestra seguente.



Esporto l'applicazione solo per Windows perché l'applicazione usa i servizi di riconoscimento vocale e di sintesi vocale di questo sistema operativo. È importante che il sistema operativo del computer che

esegue l'applicazione sia a 64 bit perché Rasa non funziona sui sistemi a 32 bit per via di un suo modulo chiamato Tensorflow.

A questo punto posso procedere con l'esportazione premendo il pulsante "Build".

Nella finestra di esplorazione delle cartelle che si apre creo un insieme di cartelle interne a quella del progetto per realizzare il percorso:

[Cartella del progetto] > Builds > Windows > x64 > [versione dell'applicazione]

A seguito dell'esportazione ottengo i seguenti file.

Nome	Ultima modifica	Tipo	Dimensione
MonoBleedingEdge	14/07/2022 15:41	Cartella di file	
Polibot_BurstDebugInformation_DoNotShip	14/07/2022 15:41	Cartella di file	
Polibot_Data		Cartella di file	
Polibot.exe		Applicazione	639 KB
UnityCrashHandler64.exe		Applicazione	1.099 KB
UnityPlayer.dll	14/07/2022 15:41	Estensione dell'ap...	28.159 KB

Tra i file esportati è presente anche una cartella relativa al debug. Questa cartella non va consegnata all'utente finale e va cancellata.

A questo punto bisogna tenere presente che l'applicazione necessita della cartella "Assets". Devo creare questa cartella, inserirvi la cartella di Rasa e creare una sottocartella di nome "Scripts". In quest'ultima devo inserire i file del server della vocalizzazione.

Nome	Ultima modifica	Tipo	Dimensione
Assets	14/07/2022 17:18	Cartella di file	
MonoBleedingEdge	14/07/2022 15:47	Cartella di file	
Polibot_Data	14/07/2022 15:47	Cartella di file	
Polibot.exe	14/07/2022 15:41	Applicazione	639 KB
UnityCrashHandler64.exe	14/07/2022 15:41	Applicazione	1.099 KB
UnityPlayer.dll	14/07/2022 15:41	Estensione dell'applicazione	28.159 KB

## Cartella dell'installer

Per la realizzazione dell'installer creo delle cartelle per creare il percorso:

Installer > x64 > [versione dell'installer]

All'interno di ciascuna versione dell'installer relativa a ciascun tentativo effettuato ci sono due cartelle: App e Dependencies.

La cartella App contiene i file per il funzionamento dell'applicazione visti nell'immagine precedente e la cartella Dependencies contiene i file necessari all'installazione dell'applicazione.

Nome	Ultima modifica	Tipo	Dimensione
Attivazione_riconoscimento_vocale.reg	24/06/2022 16:14	Voci di registrazione	1 KB
Configurazione.bat	29/06/2022 14:36	File batch Windows	1 KB
Modifica_registro.bat	25/06/2022 09:14	File batch Windows	1 KB
python-3.8.10-amd64.exe	09/05/2022 14:26	Applicazione	27.634 KB
requirements.txt	01/07/2022 09:42	Documento di testo	4 KB
VC_redist.x64.exe	09/05/2022 18:20	Applicazione	24.722 KB

I file contenuti nella cartella Dependencies sono:

- Installer di Python 3.8.10

Ho cominciato a sviluppare il progetto usando questa versione ed ho messo tutto nelle condizioni di funzionare con la medesima. Pertanto, è bene installare la versione in

questione anche se teoricamente ogni versione di Python 3.8 andrebbe bene.

- Installer di Microsoft Visual C+++

Microsoft Visual C++ 2015 – 2022 Redistributable x64 serve a fare funzionare un modulo di Python chiamato Tensorflow che viene usato da Rasa.

- requirements.txt

Questo file contiene tutti i moduli dell'ambiente virtuale di Python utilizzato dall'applicazione e verrà usato per installare i moduli in questione sul computer su cui verrà installata.

- Attivazione\_riconoscimento\_vocale.reg e Modifica\_registro.bat

Il file Attivazione\_riconoscimento\_vocale.reg contiene la voce del registro di sistema da modificare per attivare il riconoscimento vocale. Se non si attiva il riconoscimento vocale l'applicazione non sarà in grado di ascoltare la voce dell'utente.

Il file Modifica\_registro.bat ha il solo scopo di usare il file precedente.

L'utilizzo del solo file .reg non è consentito dal programma per la realizzazione dell'installer e questo mi ha costretto allo sviluppo di uno stratagemma per raggiarlo (workaround).

- Configurazione.bat

Questo file serve ad assicurarsi che i comandi vengano eseguiti nella cartella “Dependencies”, generare l’ambiente virtuale, attivarlo ed installare i moduli di Python necessari al funzionamento dell’applicazione.

### Sviluppo del file Configurazione.bat

Qui riporto il contenuto del file Configurazione.bat.

```
:: Uso una variabile contenente il percorso dello script  
SET my_path="%~dp0"  
cd %my_path%  
  
:: Adesso sono nella cartella dependencies  
cd ..  
  
:: Genero l'ambiente virtuale e lo attivo  
py -3.8 -m venv .\App\Polibot_venv  
call .\App\Polibot_venv\Scripts\activate.bat  
  
:: Adesso che ho attivato l'ambiente virtuale devo installare i moduli di Python  
python -m pip install --upgrade pip  
python -m pip install -U pip setuptools wheel  
pip install --no-cache-dir -r .\Dependencies\requirements.txt
```

Prima di cominciare a commentare lo script è necessaria una premessa.

Quando l’installer usa la shell per eseguire un comando lo fa aprendo cmd.exe che è collocato all’interno della cartella System32 di Windows e di default i comandi inseriti nel prompt vengono eseguiti in questa

cartella. In primo luogo, non mi serve che lo script agisca in questa cartella ed in secondo luogo lo script genererebbe un errore e si bloccherebbe perché non troverebbe le cartelle giuste.

Per risolvere questo problema devo fare sì che il prompt si sposti nella cartella Dependencies usando la variabile %~dp0.<sup>46</sup>

Dopo essermi spostato nella cartella Dependencies vado nella cartella di installazione del programma e creo l’ambiente virtuale all’interno della cartella App.

Per creare l’ambiente virtuale uso il Python launcher così da forzare l’utilizzo della versione 3.8 per la creazione dell’ambiente virtuale. Così facendo anche l’ambiente virtuale userà tale versione.

Purtroppo, non è possibile creare un ambiente virtuale ed usarlo dopo averlo spostato in un’altra cartella perché i file all’interno della cartella del medesimo contengono il percorso assoluto in cui questo è stato creato dall’istruzione apposita. Se avessi potuto creare l’ambiente virtuale, installare i moduli di Python ed inserirlo nella cartella App già pronto lo avrei fatto per risparmiare all’utente finale l’attesa dell’installazione dei moduli necessari al funzionamento dell’applicazione.

---

<sup>46</sup> Stack Overflow, What does %~dp0 mean, and how does it work?, <https://stackoverflow.com/questions/5034076/what-does-dp0-mean-and-how-does-it-work>

Dopo aver creato l’ambiente virtuale devo attivarlo con lo script apposito per poter usare dei comandi al suo interno.

Nell’ambiente virtuale devo aggiornare i moduli pip e wheel perché questi moduli vengono utilizzati per l’installazione degli altri moduli ed aggiornarli vuol dire installare questi ultimi al meglio.

A questo punto posso installare i moduli richiesti dall’applicazione usando il comando “pip install”. Il flag “-r” serve ad indicare che il file specificato dopo di esso sia il file dei requisiti convenzionalmente chiamato “requirements.txt”. Per spiegare il flag “--no-cache-dir” serve spiegare cosa succede quando Python scarica un modulo da internet. I moduli di Python, come altri file trasferiti via internet, al loro interno hanno un file per il controllo dell’integrità ricavato elaborando il contenuto del modulo tramite una funzione di hashing. Per evitare di ripetere questi calcoli se si dovesse scaricare lo stesso modulo, Python memorizza l’hash calcolato nella sua cache. Nel caso in cui l’utente riscarichi un modulo di Python dopo averlo disinstallato c’è la possibilità che l’hash ricevuto all’interno del modulo riscaricato non coincida con quello memorizzato localmente generando così un errore bloccante. Durante i tentativi effettuati per lo sviluppo dell’installer sono capitate situazioni di questo tipo quindi ho usato questo flag per imporre a Python di non usare il contenuto della sua cartella “cache”.

## Sviluppo dei file per la modifica del registro di sistema

Prima di riportare il contenuto del file .reg e commentarne il contenuto è necessario fare una premessa. Il programma per la realizzazione dell'installer consente di inserire delle variabili nel registro di sistema ma solo di tipo stringa. In questo caso la variabile da inserire nel registro è di tipo double word e questo inserimento necessita di un workaround.

Qui riporto il contenuto del file reg.

```
Windows Registry Editor Version 5.00

[HKEY_CURRENT_USER\Software\Microsoft\
Speech_OneCore\Settings\OnlineSpeechPrivacy]

"HasAccepted"=dword:00000001
```

La prima riga del file è relativa alla versione del registro di sistema.

La seconda riga del file (spezzata in due per motivi di paginazione) è la posizione della variabile all'interno del database gerarchico e la terza riga è costituita dalla variabile, il suo tipo ed il suo valore.

Dopo aver scritto questo file ho effettuato dei tentativi per fare sì che l'installer riesca ad usarlo ma non hanno avuto successo. Facendo delle ricerche su internet ho scoperto che la soluzione consiste nel creare un file batch che utilizza il file reg commentato poc'anzi.

Qui riporto il contenuto del file Modifica\_registro.bat.

```
reg import Attivazione_riconoscimento_vocale.reg
```

Questo file contiene solo il comando “reg import” che serve a modificare il registro di sistema in base alle istruzioni all’interno del file specificato.

## **Realizzazione dell’installer**

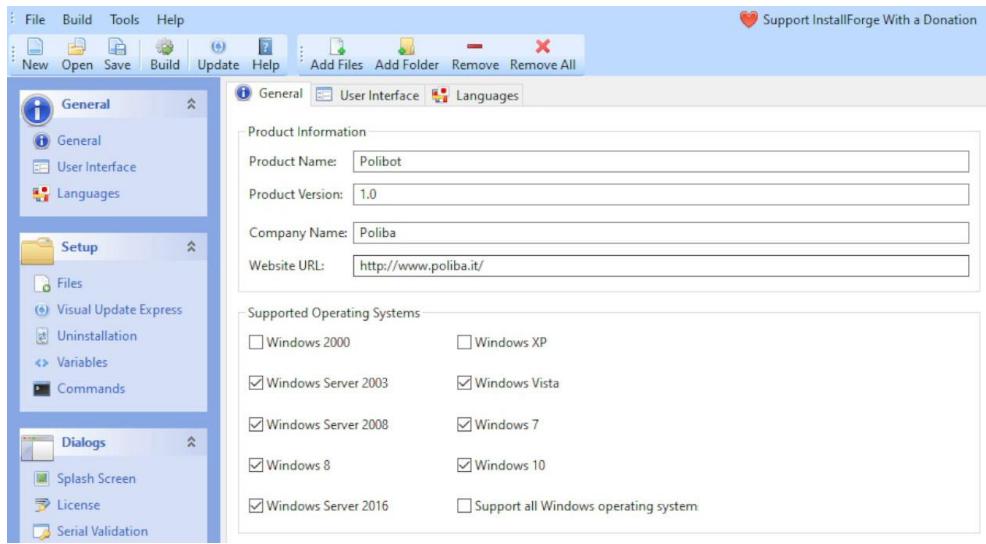
Per la realizzazione dell’installer uso un programma gratuito chiamato “InstallForge”. In questo programma bisogna compilare una serie di schede organizzate in sezioni ed infine esportare l’installer.

InstallForge, nei campi delle sue schede, usa delle variabili proprie i cui nomi sono racchiusi in parentesi angolari. Ad esempio, le variabili <Company> e <AppName> servono ad indicare i contenuti dei rispettivi campi compilabili nella sezione “General”.

### **Sezione “General”**

In questa sezione si impostano le generalità sull’installer come le informazioni sull’applicazione e la lingua.

La prima scheda di questa sezione è la seguente:



Nella parte superiore si inseriscono le informazioni del programma e dell’azienda a cui appartiene. Nella parte inferiore si scelgono le versioni di Windows su cui si può installare il programma.

Dall’insieme di sistemi operativi selezionati di default devo rimuovere Windows XP perché questo non è compatibile con Python 3.<sup>47</sup>

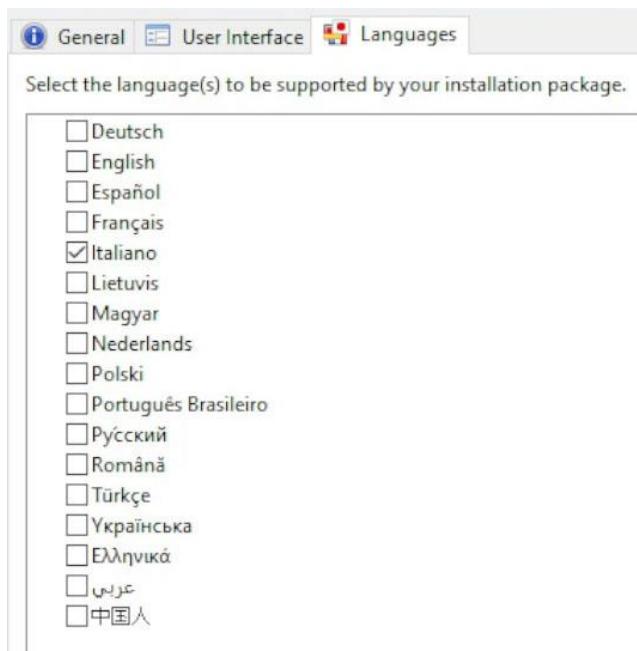


Nella scheda “User Interface” si possono scegliere la “Wizard Image” e la “Header Image” che sono rispettivamente l’immagine mostrata a

---

<sup>47</sup> Python, Python Releases for Windows, <https://www.python.org/downloads/windows/>

sinistra del testo nella prima schermata dell'installer e l'immagine mostrata in alto a destra nelle altre schermate del medesimo.

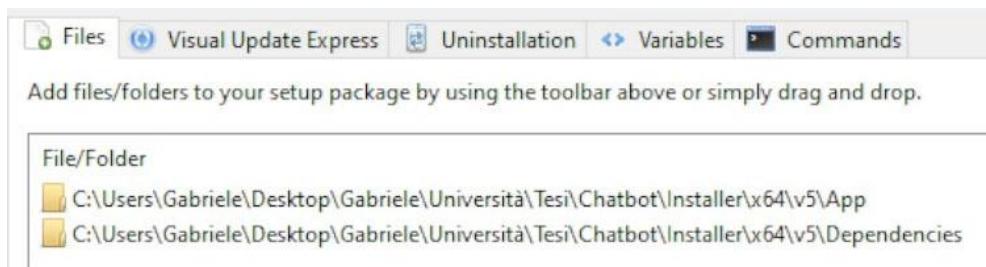


Nella scheda “Languages” si possono scegliere le lingue dell’installer. Scelgo di lasciare solo l’italiano perché questo programma verrà usato solo dagli impiegati del politecnico.

## **Sezione “Setup”**

La sezione “Setup” è relativa alle impostazioni per l’installazione, l’aggiornamento e la disinstallazione del programma.

La scheda “Files” è divisa in due parti. Nella parte superiore si stabilisce quali sono i file da trasferire nella cartella di installazione del programma.



Nella parte inferiore si stabilisce il percorso della cartella di installazione e se si vuole che l'utente possa cambiarla.

Default Installation Path

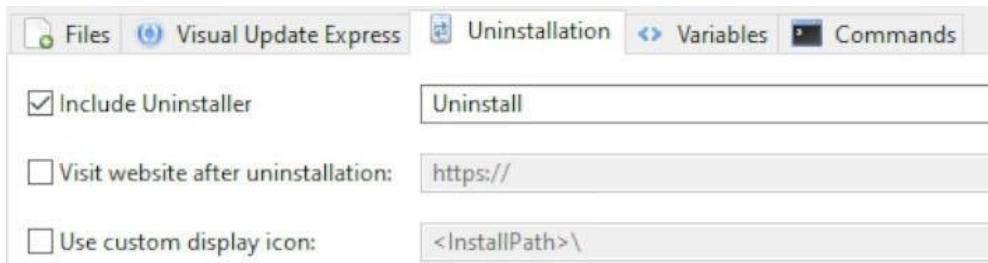
C:\Program Files\<Company>\<AppName>\

Allow user to change installation path

Ho modificato il percorso di default perché quello fornito dal programma è quello in cui vengono memorizzati i file dei programmi a 32 bit. Il percorso che ho indicato è quello per i programmi a 64 bit.

La scheda “Visual Update Express” non va compilata perché il programma non verrà aggiornato tramite internet.

Nella scheda “Uninstallation” si può stabilire se generare un programma per la disinstallazione dell’applicazione.



Nella scheda “Variables” si possono aggiungere altre variabili prendendone i valori dal registro di sistema. Le variabili aggiunte in questo modo possono essere utilizzate come se fossero variabili proprie di InstallForge racchiudendole tra parentesi angolari.

Nella scheda “Commands” si possono stabilire i comandi da usare per l’installazione dell’applicazione dopo che i file sono stati trasferiti.

Type	Command	Parameters	Options
Execute Application	<InstallPath>\Dependencies\python-3.8.10-amd64.exe	PrependPath=1	-wait
Execute Application	<InstallPath>\Dependencies\VC_redist.x64.exe		-wait
Execute Application	<InstallPath>\Dependencies\Configurazione.bat		-wait
Execute Application	<InstallPath>\Dependencies\Modifica_registro.bat		-wait -hide

I comandi possono essere di due tipi: “Execute Application” e “Shell Execute”.

I primi sono comandi eseguiti come se un utente li avviasse con un doppio click e sono indicati per i programmi con interfaccia grafica; i secondi sono comandi eseguiti come se fossero avviati all’interno della shell (cmd.exe).

I comandi sono dotati di parametri ed opzioni. I parametri sono relativi al particolare comando mentre le opzioni possono essere usate per tutti

i comandi. L’opzione “-wait” serve ad imporre che si aspetti la fine del comando corrente prima dell’esecuzione del successivo. Questa opzione rende la sequenza di esecuzione dei comandi più ordinata, inoltre l’esecuzione contemporanea di più comandi potrebbe confondere l’utente finale. L’opzione “-hide” serve a nascondere la finestra di esecuzione del comando. Se venisse mostrata la finestra di un comando che richiede poco tempo l’utente vedrebbe una finestra aprirsi, vedrebbe qualche riga di testo al suo interno e poi vedrebbe la finestra chiudersi prima di riuscire a leggere il suo contenuto. L’utente potrebbe avere voglia di conoscere il contenuto della shell, oppure potrebbe pensare che qualcosa non sia andato per il verso giusto e riprovare ad installare il programma (ottenendo lo stesso risultato), oppure potrebbe anche trovarlo sgradevole senza farsi troppe domande a riguardo.

I comandi cominciano con la variabile <InstallPath> che si riferisce alla cartella contenente le cartelle App e Dependencies.

Il primo comando serve ad avviare l’installer di Python 3.8.10. Il parametro “PrependPath=1” serve a fare sì che la casella che l’utente usa per scegliere se aggiungere il percorso di Python alla variabile di ambiente “PATH” sia già spuntata.

Il secondo comando installa Visual C++. Faccio eseguire questo comando per secondo così da avere tutto il necessario per l’installazione dei moduli di Python.

Il terzo comando esegue il file batch per la configurazione dell’ambiente virtuale. A questo punto è lecito domandarsi perché io esegua un file batch usando il tipo “Execute Application” piuttosto che “Shell Execute”. Innanzitutto, i comandi di tipo “Shell Execute” non possono usare la variabile <InstallPath>, inoltre se usassi un comando di questo tipo per l’esecuzione di ogni comando contenuto nel file batch avrei il problema della discontinuità della sequenza dei comandi perché quando si esegue un comando di tipo “Shell Execute” si apre una shell, viene eseguito il comando poi questa si chiude. Quindi, il risultato dell’operazione va perduto ed il comando successivo non può utilizzarlo. Questo vuol dire che non si può usare il comando per l’attivazione dell’ambiente virtuale e poi installare i moduli in esso nel comando successivo.

La configurazione dell’ambiente virtuale usa solo l’opzione “-wait” perché è un processo molto lungo e se venisse nascosto l’utente vedrebbe la barra di progresso dell’installer ferma per molto tempo e comincerebbe a chiedersi se stia andando tutto bene.

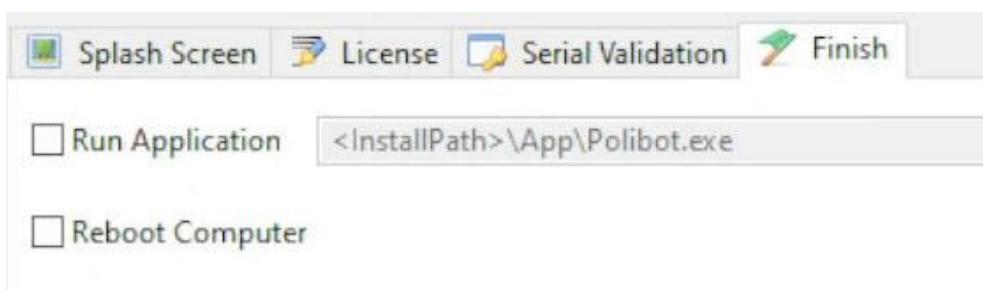
La quarta istruzione serve ad attivare il riconoscimento vocale. Trattandosi di un’istruzione che richiede poco tempo per essere eseguita uso l’opzione “-hide” oltre all’opzione “-wait”.

Purtroppo, i comandi di tipo “Shell Execute” non possono usare la variabile contenente il percorso di installazione altrimenti avrei usato un comando di questo tipo.

## **Sezione “Dialogs”**

Questa sezione serve per la compilazione delle possibili schermate di dialogo dell’installer. Le schede di questa sezione servono a mostrare lo splash screen, mostrare l’accordo di licenza, richiedere all’utente un codice per la validazione (in maniera simile alla validazione di Windows) e decidere cosa mostrare nella schermata finale.

Dato che l’applicazione verrà usata solo internamente al politecnico non serve compilare le prime tre schede ed al massimo si può decidere di compilare la quarta.



Nella schermata finale si può mostrare all’utente una casella in cui questo può decidere cosa fare al termine dell’installazione. Si può o riavviare il sistema o avviare l’applicazione.

Al termine dell’installazione non è necessario il riavvio del sistema, quindi, al più si può decidere se avviare l’applicazione o no.

## Sezione “System”

La sezione “System” contiene le modalità con cui l’applicazione interagirà con il sistema. Nella scheda “Registry” si possono impostare le variabili da aggiungere al registro di sistema e nella scheda “Shortcuts” si possono impostare le scorciatoie per accedere all’applicazione.

Le variabili del file voce\_Cosimo.reg sono tutte di tipo stringa quindi ho potuto riportarle una ad una nella scheda “Registry”.

Root Key	Sub Key	Value Name	Value Data
HKEY_LOCAL_MACHINE	SOFTWARE\Microsoft\Speech\Voices\Tokens\MS..	Microsoft Cosimo - Italian (Italy)	
HKEY_LOCAL_MACHINE	SOFTWARE\Microsoft\Speech\Voices\Tokens\MS..	410	Microsoft Cosimo - Italian (Italy)
HKEY_LOCAL_MACHINE	SOFTWARE\Microsoft\Speech\Voices\Tokens\MS..	CLSID	{179F3D56-1B0B-42B2-A962-59B7EF59FE1B}
HKEY_LOCAL_MACHINE	SOFTWARE\Microsoft\Speech\Voices\Tokens\MS..	LangDataPath	%windir%\Speech_OneCore\Engines\TTS\it-IT\MS..
HKEY_LOCAL_MACHINE	SOFTWARE\Microsoft\Speech\Voices\Tokens\MS..	VoicePath	%windir%\Speech_OneCore\Engines\TTS\it-IT\MS..
HKEY_LOCAL_MACHINE	SOFTWARE\Microsoft\Speech\Voices\Tokens\MS..	Age	Adult
HKEY_LOCAL_MACHINE	SOFTWARE\Microsoft\Speech\Voices\Tokens\MS..	DataVersion	11.0.2013.1022
HKEY_LOCAL_MACHINE	SOFTWARE\Microsoft\Speech\Voices\Tokens\MS..	Gender	Male
HKEY_LOCAL_MACHINE	SOFTWARE\Microsoft\Speech\Voices\Tokens\MS..	Language	410
HKEY_LOCAL_MACHINE	SOFTWARE\Microsoft\Speech\Voices\Tokens\MS..	Name	Microsoft Cosimo
HKEY_LOCAL_MACHINE	SOFTWARE\Microsoft\Speech\Voices\Tokens\MS..	SayAsSupport	spell=NativeSupported; cardinal=GlobalSupported;
HKEY_LOCAL_MACHINE	SOFTWARE\Microsoft\Speech\Voices\Tokens\MS..	SharedPronunciation	
HKEY_LOCAL_MACHINE	SOFTWARE\Microsoft\Speech\Voices\Tokens\MS..	Vendor	Microsoft
HKEY_LOCAL_MACHINE	SOFTWARE\WOW6432Node\Microsoft\SPEECH\V...	11.0	Microsoft Cosimo - Italian (Italy)
HKEY_LOCAL_MACHINE	SOFTWARE\WOW6432Node\Microsoft\SPEECH\V...	410	Microsoft Cosimo - Italian (Italy)
HKEY_LOCAL_MACHINE	SOFTWARE\WOW6432Node\Microsoft\SPEECH\V...	CLSID	{179F3D56-1B0B-42B2-A962-59B7EF59FE1B}
HKEY_LOCAL_MACHINE	SOFTWARE\WOW6432Node\Microsoft\SPEECH\V...	LangDataPath	%windir%\Speech_OneCore\Engines\TTS\it-IT\MS..
HKEY_LOCAL_MACHINE	SOFTWARE\WOW6432Node\Microsoft\SPEECH\V...	VoicePath	%windir%\Speech_OneCore\Engines\TTS\it-IT\MS..
HKEY_LOCAL_MACHINE	SOFTWARE\WOW6432Node\Microsoft\SPEECH\V...	Age	Adult
HKEY_LOCAL_MACHINE	SOFTWARE\WOW6432Node\Microsoft\SPEECH\V...	DataVersion	11.0.2013.1022
HKEY_LOCAL_MACHINE	SOFTWARE\WOW6432Node\Microsoft\SPEECH\V...	Gender	Male
HKEY_LOCAL_MACHINE	SOFTWARE\WOW6432Node\Microsoft\SPEECH\V...	Language	410
HKEY_LOCAL_MACHINE	SOFTWARE\WOW6432Node\Microsoft\SPEECH\V...	Name	Microsoft Cosimo
HKEY_LOCAL_MACHINE	SOFTWARE\WOW6432Node\Microsoft\SPEECH\V...	SayAsSupport	spell=NativeSupported; cardinal=GlobalSupported;
HKEY LOCAL MACHINE	SOFTWARE\WOW6432Node\Microsoft\SPEECH\V...	SharedPronunciation	

La scheda “Shortcuts” è suddivisa in due parti. Nella parte superiore si possono impostare le scorciatoie che l’installer deve generare a seguito dell’installazione.

Destination	Shortcut Name	Target File
Desktop	Polibot	<InstallPath>\App\Polibot.exe
Startmenu	Polibot	<InstallPath>\App\Polibot.exe

Nella parte inferiore si possono modificare il percorso di default, la possibilità dell’utente di modificarlo e la possibilità di generare le scorciatoie per tutti gli utenti.

Default path for start menu shortcuts: `<Company>\<AppName>\`

- Allow user to change the start menu shortcut path
- Create start menu shortcuts for all users
- Create desktop shortcuts for all users

## Sezione “Build”

Nella sezione build si possono modificare le impostazioni con cui l’installer verrà esportato.

Setup File:	<code>C:\Users\Gabriele\Desktop\Gabriele Università\Tool\Charles\Installer\x64\v5\Polibot_installer_x64.exe</code>
Icons	
Setup Icon:	<code>C:\Program Files (x86)\solicus\InstallForge\icons\modern.ico</code>
Uninstaller Icon:	<code>C:\Program Files (x86)\solicus\InstallForge\icons\modern.ico</code>
Compression	
Compression Method:	Lempel-Ziv-Markov Chain Algorithm (LZMA)
Compression Level:	Max

La prima impostazione riguarda il percorso in cui l'installer verrà esportato.

La seconda parte delle impostazioni riguarda l'icona da assegnare al programma di installazione e di disinstallazione. Le icone fornite dal programma sono due una più vecchia ed una moderna.

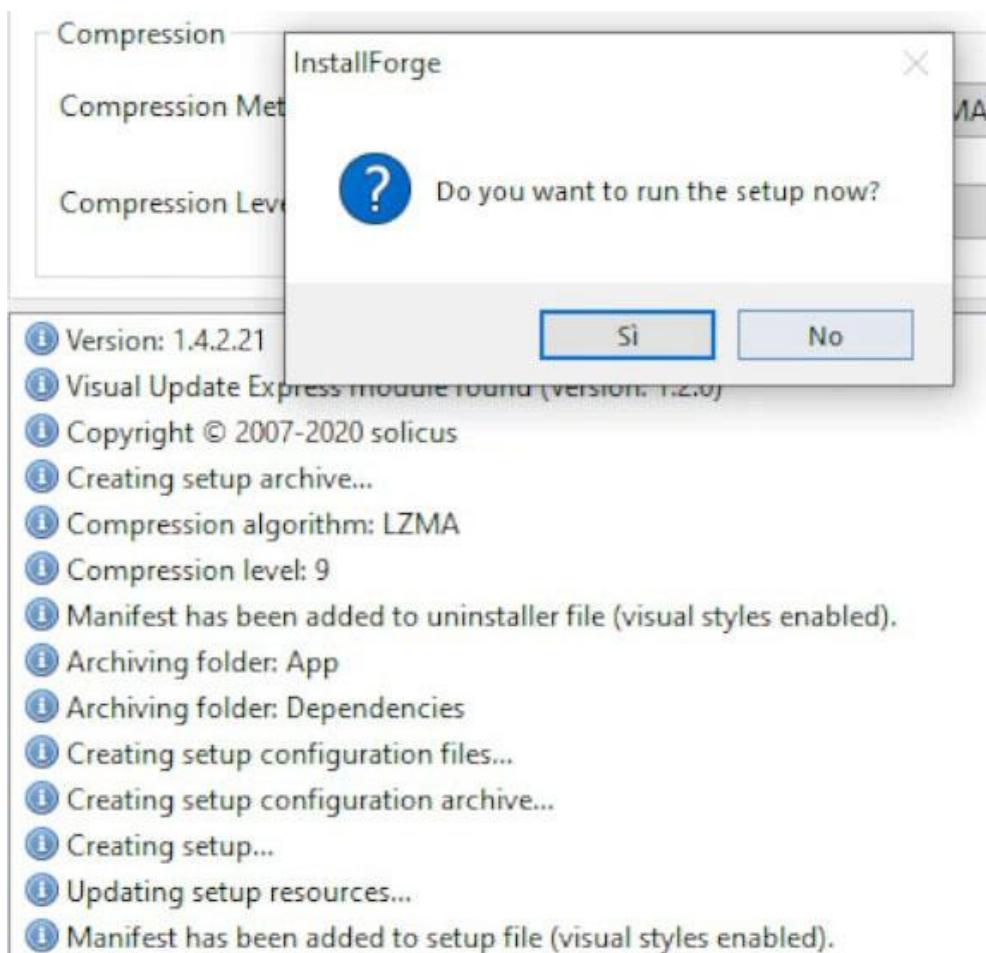


La terza parte delle impostazioni riguarda l'algoritmo ed il livello di compressione dei file da includere nell'installer, cioè quelli specificati nella sezione “Setup”.

A seguito della compilazione di questa scheda si può esportare l'installer cliccando sull'icona “Build” nella barra degli strumenti in alto a sinistra.

Al termine del processo di esportazione viene mostrato un messaggio che indica la fine del processo e richiede se si vuole avviare l'installer.

Inoltre, nella echo sottostante la parte da compilare della scheda sono elencate le operazioni svolte da InstallForge.

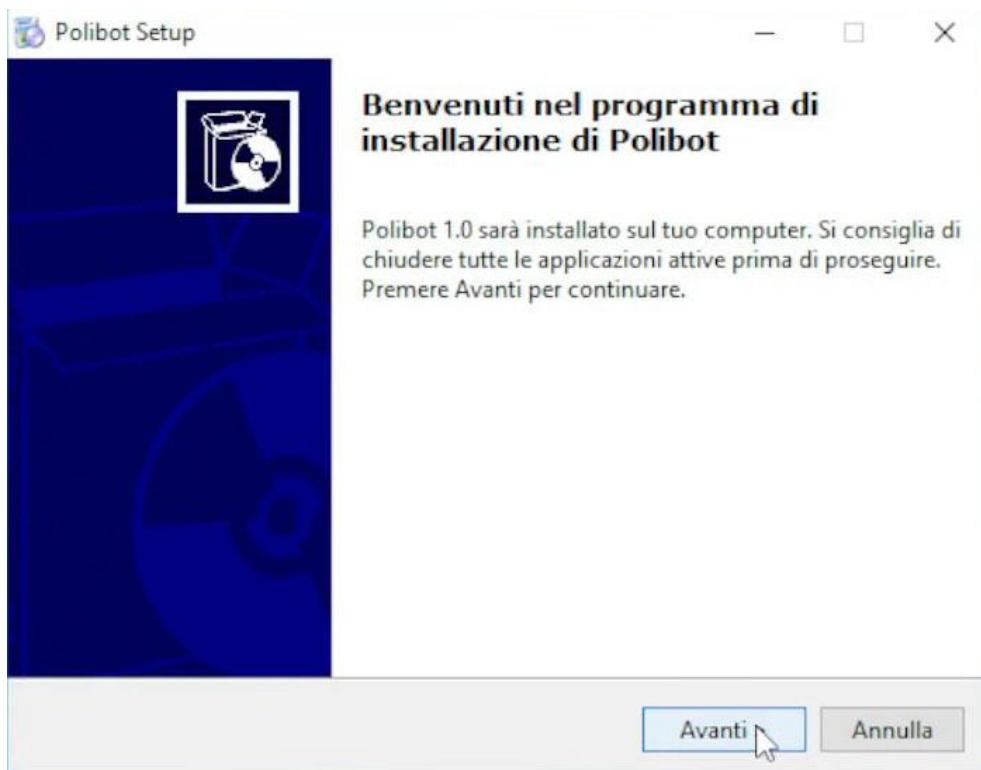


A questo punto nella cartella dell'installer è comparso il file eseguibile per l'installazione. Questo file può essere inviato ad un altro computer e si occuperà dell'installazione del programma in maniera autosufficiente.

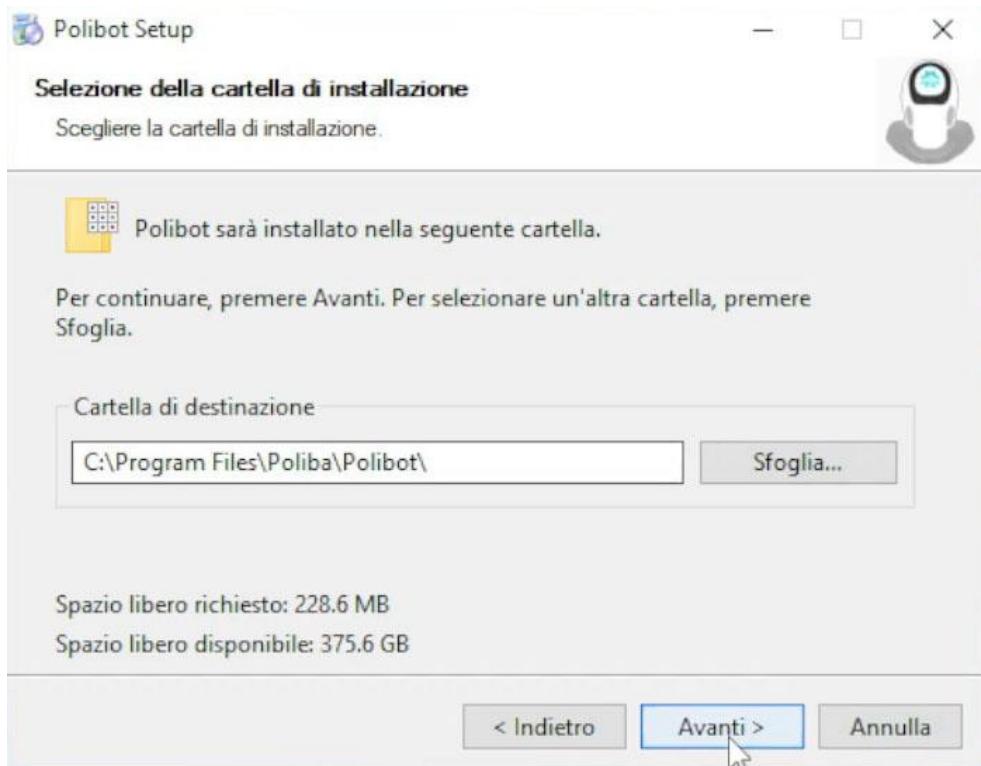
## Installazione

Ora è giunto il momento di vedere in che modo i passaggi svolti nel capitolo precedente vengono messi in atto dall'installer.

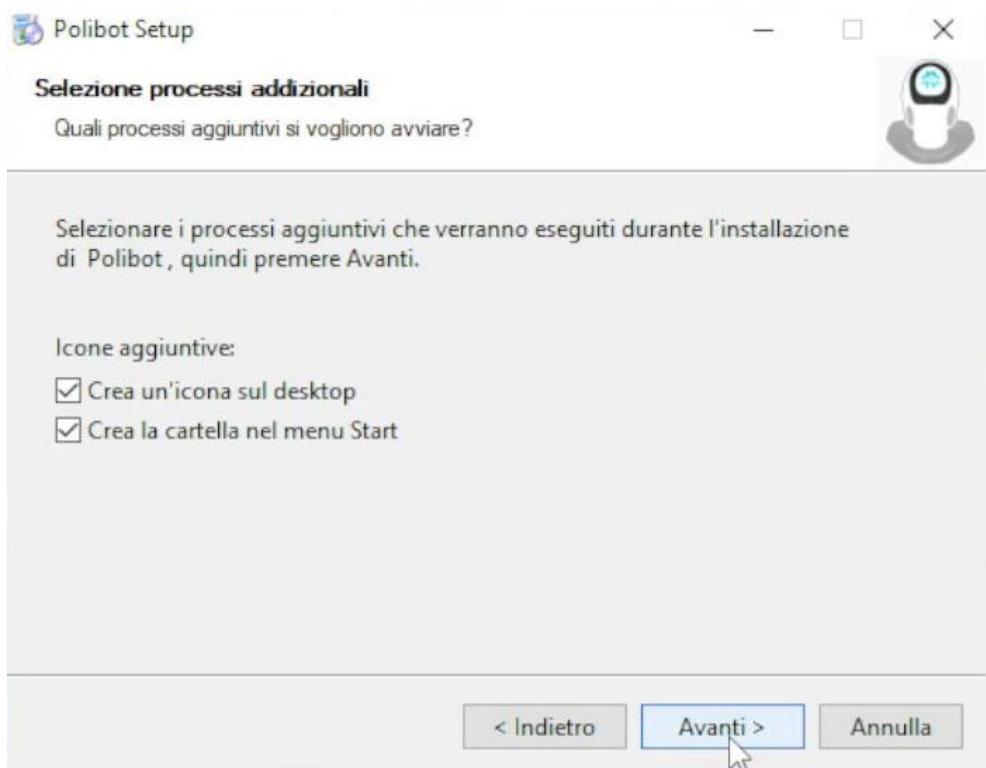
Nella prima schermata vengono mostrati il nome dell'applicazione e la sua versione, entrambi impostati nella scheda “General”.



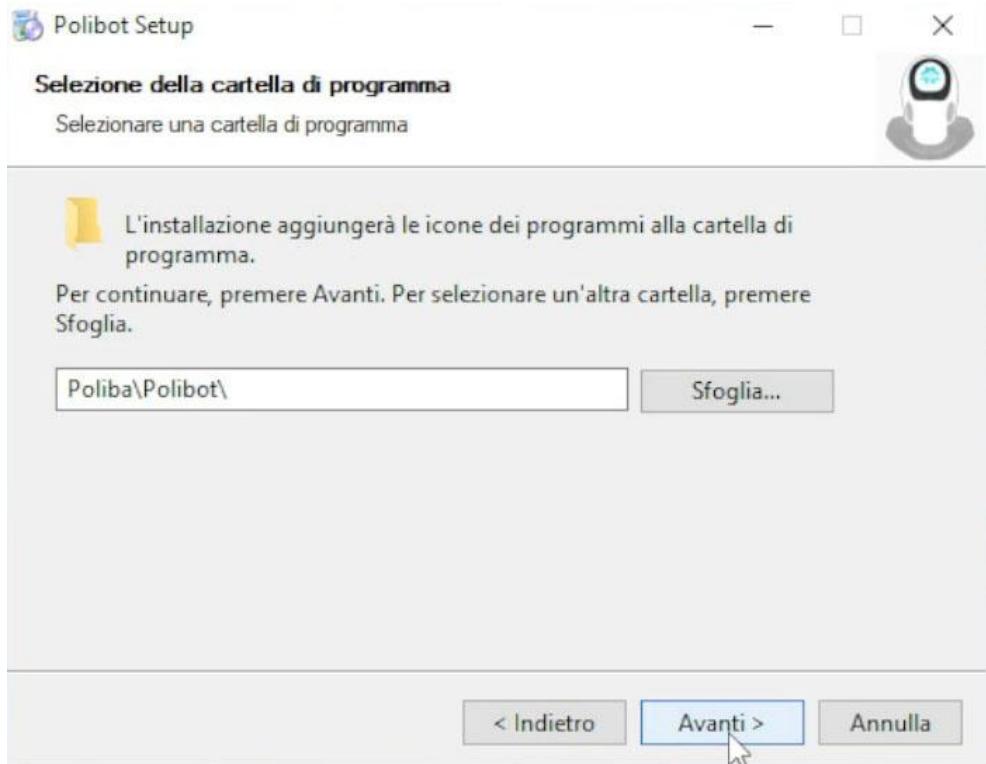
Nella seconda schermata si chiede all'utente di impostare una cartella di installazione e ne viene mostrata una di default. Queste impostazioni sono nella scheda “Files”.



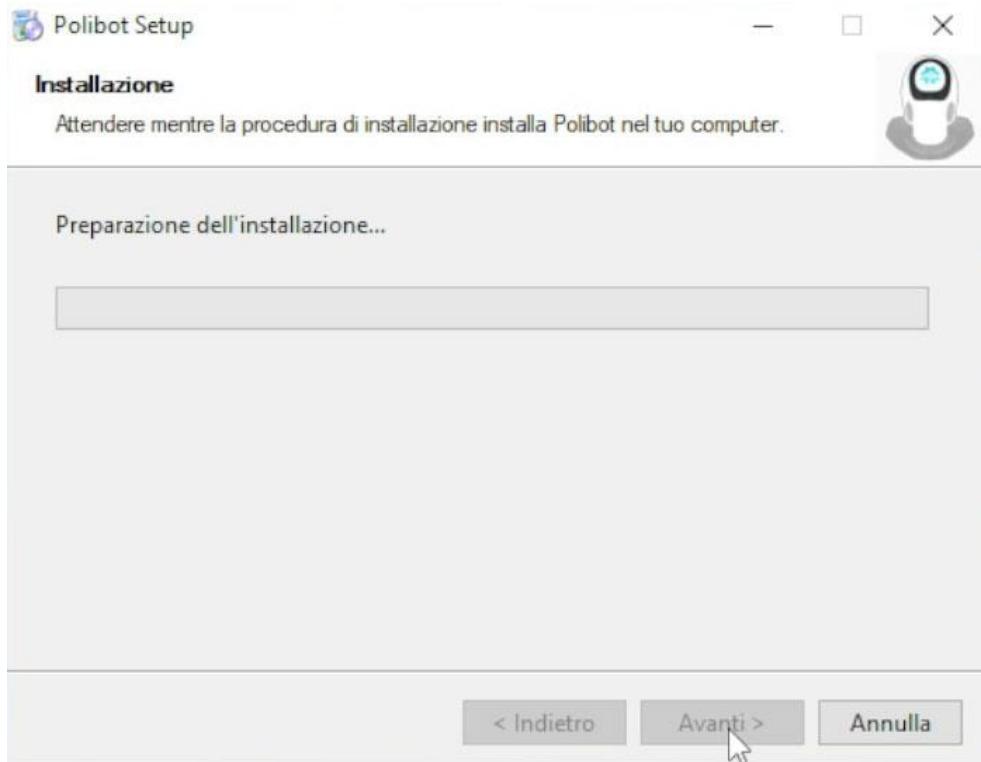
Nella terza schermata l'utente ha la possibilità di scegliere di creare icone aggiuntive come stabilito nella scheda “Shortcuts”.



La quarta schermata chiede all'utente il percorso in cui inserire il collegamento all'interno del menu di avvio. Anche questa impostazione è contenuta nella scheda “Shortcuts”.

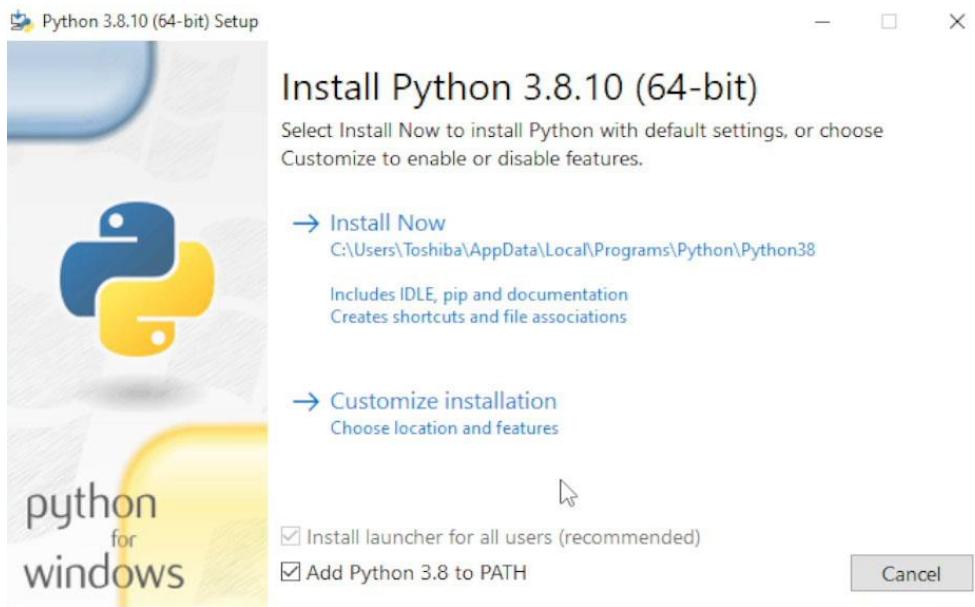


Nella quinta schermata è presente una barra di progresso relativa al trasferimento dei file.



Al termine del trasferimento dei file l'installer comincia ad eseguire i comandi impostati nell'apposita schermata. Dal flusso di esecuzione di questi si può notare l'effetto delle opzioni “-wait” e “-hide” spiegate in precedenza.

Il primo comando eseguito riguarda l'installazione di Python. Nella parte bassa della schermata si può notare che la spunta per aggiungere il percorso di Python alla variabile di sistema “PATH” è già spuntata perché ho usato il parametro “PrependPath=1”.



Al termine dell'installazione di Python si apre la finestra di installazione di Visual C++. In questa finestra bisogna solo accettare le condizioni di licenza e procedere con l'installazione. Non è necessario riavviare il computer al termine dell'installazione dell'applicazione perché la versione Redistributable di Visual C++ non necessita il riavvio del sistema per essere utilizzata.



Dopo aver installato Visual C++ viene eseguito il file Configurazione.bat.

```
C:\Program Files\Poliba\Polibot\Dependencies>SET my_path="C:\Program Files\Poliba\Polibot\Dependencies\"  
C:\Program Files\Poliba\Polibot\Dependencies>cd "C:\Program Files\Poliba\Polibot\Dependencies\"  
C:\Program Files\Poliba\Polibot\Dependencies>cd ..  
C:\Program Files\Poliba\Polibot>py -3.8 -m venv .\App\Polibot_venv  
C:\Program Files\Poliba\Polibot>call .\App\Polibot_venv\Scripts\activate.bat  
Requirement already satisfied: pip in c:\program files\poliba\polibot\app\polibot_venv\lib\site-packages (21.1.1)  
Collecting pip  
  Using cached pip-22.1.2-py3-none-any.whl (2.1 MB)  
Installing collected packages: pip  
  Attempting uninstall: pip  
    Found existing installation: pip 21.1.1  
    Uninstalling pip-21.1.1:  
      Successfully uninstalled pip-21.1.1
```

Al termine dell'esecuzione di questo file viene eseguito il file batch per la modifica del registro del sistema ma l'utente non lo vede perché

l'esecuzione di questo comando è nascosta. Inoltre, essendo un comando eseguito in poco tempo non causa alcun problema all'utente che può procedere con l'installazione agnostico dell'accaduto.

L'ultima schermata è quella in cui si termina l'installer. Non vengono mostrate le spunte per il riavvio del sistema o per l'avvio dell'applicazione come impostato nella scheda "Finish".



A questo punto andando nella cartella di installazione del programma, all'interno della cartella App, si nota che non sono presenti solo i file impostati all'interno dell'installer ma anche la cartella "Polibot\_venv" che dimostra il corretto funzionamento del file Configurazione.bat.

Nome	Ultima modifica	Tipo	Dimensione
Assets	14/07/2022 19:02	Cartella di file	
MonoBleedingEdge	14/07/2022 19:02	Cartella di file	
Polibot_Data	14/07/2022 19:02	Cartella di file	
Polibot_venv	14/07/2022 19:17	Cartella di file	
Polibot.exe	14/07/2022 19:02	Applicazione	639 KB
UnityCrashHandler64.exe	14/07/2022 19:02	Applicazione	1.099 KB
UnityPlayer.dll	14/07/2022 19:02	Estensione dell'applicazione	28.159 KB

# **Conclusioni**

## **Risultati ottenuti**

Il risultato del lavoro compiuto è un programma per Windows a 64 bit che verrà usato per la realizzazione di un punto informativo.

Nel corso dello svolgimento dei lavori il progetto non è stato modificato solo per le esigenze tecniche che esporrò in seguito ma anche per una modifica di requisiti suggeritami dal mio relatore e correlatore.

Il progetto iniziale prevedeva che io realizzassi una stanza virtuale con Probuilder e Snaps al cui interno è presente una scrivania su cui poggia la testa del robot. In seguito, il mio relatore e correlatore mi hanno chiesto di adattare il lavoro svolto ad uno schermo con piramide olografica.

Inizialmente l'applicazione avrebbe dovuto funzionare su un Raspberry ma le funzioni di speech-to-text e text-to-speech sono implementate usando delle librerie di Windows, quindi, un sistema operativo basato su Linux non avrebbe potuto utilizzarle. A questo punto ho considerato due alternative: la prima era installare un sistema operativo della famiglia di Windows sul Raspberry e l'altra era trovare delle librerie alternative per Linux.

Per quanto riguarda la prima alternativa ho installato Windows IoT sul Raspberry e poi ho provato ad installare Python. Purtroppo, l’installazione di Python su Windows IoT non è semplice e per alcuni dispositivi, come il mio, la procedura di installazione fornita da Microsoft non funziona!<sup>48</sup> Inoltre, anche se fossi riuscito ad installare Python non avrei potuto installare Rasa dato che questo non è disponibile per i dispositivi con architettura ARM64.

Per quanto riguarda la seconda alternativa i servizi per l’accessibilità offerti da Linux sono di qualità inferiore rispetto a quelli di Microsoft e sono anche più difficili da gestire. In fondo era ovvio che Windows disponesse di servizi di accessibilità migliori dato che è un sistema operativo più diffuso.

Rassegnatomi allo sviluppo per Windows ho notato che l’utilizzo della libreria System.Speech.dll funziona eccellentemente nell’editor di Unity ma non funziona nella versione esportata dell’applicazione. Infatti, non appena questa si apre si verifica un crash dovuto alla mancanza della libreria. Individuare la causa del crash non è stato semplice. L’errore fornитomi dal debugger dell’applicazione non era chiaro e mi ci sono voluti giorni per capire che non si possono usare librerie non supportate dal backend di Unity. La cosa mi è sembrata strana dato che Unity si basa sul .NET Framework e che la libreria in

---

<sup>48</sup> Microsoft Docs, Python, <https://docs.microsoft.com/en-us/windows/iot-core/developer-tools/python>

questione appartiene ad esso. L'impossibilità di usare System.Speech.dll nel codice in C# nonostante tutti i tentativi di importazione mi ha messo davanti a due alternative: la prima era quella di provare a manipolare Mono (il backend di Unity) per fargli supportare la libreria che mi serve e l'altra era di trovare un'altra maniera di utilizzarla. Dato che nella fase di svolgimento del progetto ho realizzato un workaround in Python al lettore sarà ovvio che ho optato per la seconda ma non prima di provare la prima alternativa. Infatti, se fossi riuscito nell'intendo di manipolare Mono avrei sviluppato un'applicazione con un codice più semplice ed anche più leggera dato che avrei potuto evitare di usare uno script in Python. Spero vivamente che chi si occuperà dell'applicazione dopo di me riesca in questo intento o semplicemente che Unity aggiunga il supporto a questa libreria ufficialmente.

Dopo aver sviluppato un'applicazione funzionante per Windows a 64 bit mi è venuto in mente che magari avrei potuto usare Docker per poterla usare sul Raspberry. Dopo aver imparato ad usare Docker ed essere riuscito ad inserire l'applicazione in un container ho provato ad utilizzarla in locale ma non è stato possibile perché Docker non può essere usato per applicazioni con un'interfaccia grafica. Ho anche provato ad installare un XServer per Windows come quello presente nei sistemi operativi della famiglia di Linux ma l'applicazione non ha funzionato. Facendo delle ricerche su internet ho scoperto che la ricerca scientifica si sta prodigando per sviluppare un sistema per ottenere

applicazioni “system agnostic”, cioè che funzionano indipendentemente dal sistema operativo su cui sono installate.

## Sviluppi futuri

### 1. Utilizzo di System.Speech.dll interno a Unity:

Ormai penso che il lettore abbia capito che questo sviluppo futuro è, a mio avviso, il più importante perché consentirebbe di alleggerire l'applicazione e renderla più performante. Infatti se il codice in C# potesse usare questa libreria non ci sarebbe bisogno di utilizzarla in Python ed interagire con essa tramite richieste web periodiche. Lo si potrebbe semplicemente fare tramite delle funzioni. Ovviamente quanto detto per questa libreria vale anche per mscorelib.dll.

### 2. Realizzazione di servizi di sistema che gestiscano i server utilizzati all'applicazione:

Al momento il codice dell'applicazione si occupa di avviare, monitorare e riavviare i server. Un servizio di sistema potrebbe fare la stessa cosa. La delegazione della gestione dei server al sistema operativo consentirebbe di ridurre le dimensioni del codice.

3. Inclusione dei file per utilizzare la voce “Microsoft – Cosimo” nell’installer:

Al momento l’installer crea le voci del registro di sistema necessarie all’utilizzo della voce in questione ma se questa voce non è presente nel sistema l’applicazione non può utilizzarla.

4. Aggiunta di informazioni e di risposte alternative:

Questo sviluppo futuro è ovvio e di facile realizzazione. Basta lavorare sui file usati da Rasa.

## Sitografia

### Rasa

- Sito ufficiale di Rasa, <https://rasa.com/>
- Rasa Docs, Installation, <https://rasa.com/docs/rasa/installation>
- Rasa Docs, Model Configuration,  
<https://rasa.com/docs/rasa/model-configuration/>
- spaCy, Models & Languages, <https://spacy.io/usage/models>
- Rasa Docs, NLU Training Data, <https://rasa.com/docs/rasa/nlu-training-data/>
- Rasa Docs, Domain, <https://rasa.com/docs/rasa/domain/>
- Rasa Docs, Rules, <https://rasa.com/docs/rasa/rules/>
- Rasa Docs, Stories, <https://rasa.com/docs/rasa/stories/>
- Rasa Docs, Test Stories,  
<https://rasa.com/docs/rasa/2.x/training-data-format/#test-stories>
- Rasa Docs, Actions, <https://rasa.com/docs/rasa/actions/>
- Rasa Docs, Rasa SDK, Actions, <https://rasa.com/docs/action-server/sdk-actions/>
- Rasa Docs, Command Line Interface,  
<https://rasa.com/docs/rasa/command-line-interface>
- Rasa Docs, Rasa Playground,  
<https://rasa.com/docs/rasa/playground/>

## Unity

- Unity (motore grafico),  
[https://it.wikipedia.org/wiki/Unity\\_\(motore\\_grafico\)](https://it.wikipedia.org/wiki/Unity_(motore_grafico))
- Unity Documentation, Unity Manual,  
<https://docs.unity3d.com/Manual/index.html>
- Unity Documentation, ProBuilder,  
<https://docs.unity3d.com/Manual/com.unity.probuilder.html>
- YouTube, Bliz Studio, Unity 3d - ProBuilder Series - The "UV Editor" tool allows you to manipulate and unwrap your UVs,  
<https://www.youtube.com/watch?v=7XSPwH7dOIw>
- Unity Asset Store, Hologram Pyramid,  
<https://assetstore.unity.com/packages/tools/hologram-pyramid-61735>
- Unity, Canvas,  
<https://docs.unity3d.com/Packages/com.unity.ugui@1.0/manual/UICanvas.html>
- Unity, Canvas Scaler,  
<https://docs.unity3d.com/Packages/com.unity.ugui@1.0/manual/script-CanvasScaler.html>
- Unity Documentation, Camera component,  
<https://docs.unity3d.com/Manual/class-Camera.html>
- Unity Documentation, Introduction to lighting,  
<https://docs.unity3d.com/Manual/LightingInUnity.html>

- YouTube, Brackeys, LIGHTING in Unity,  
<https://www.youtube.com/watch?v=VnG2gOKV9dw>
- Unity Documentation, Types of light,  
<https://docs.unity3d.com/Manual/Lighting.html>
- Unity Documentation, Light Modes,  
<https://docs.unity3d.com/Manual/LightModes.html>
- Unity Documentation, Lighting Mode,  
<https://docs.unity3d.com/Manual/lighting-mode.html>
- Unity Documentation, Lightmapping,  
<https://docs.unity3d.com/Manual/Lightmappers.html>
- Unity Documentation, Real-time lighting,  
<https://docs.unity3d.com/560/Documentation/Manual/LightMode-Realtime.html>
- Unity Documentation, Light Mode: Baked,  
<https://docs.unity3d.com/Manual/LightMode-Baked.html>
- Unity Documentation, Light Mode: Mixed,  
<https://docs.unity3d.com/Manual/LightMode-Mixed.html>
- YouTube, Brackeys, 2D Animation in Unity (Tutorial),  
<https://www.youtube.com/watch?v=hkaysu1Z-N8>
- Unity Documentation, Animation Layers,  
<https://docs.unity3d.com/Manual/AnimationLayers.html>
- Unity Forum, How to create a static first state for an animation controller?, <https://forum.unity.com/threads/how-to-create-a-static-first-state-for-an-animation-controller.432741/>

- YouTube, Brackeys, How to BUILD / EXPORT your Game in Unity (Windows | Mac | WebGL),  
<https://www.youtube.com/watch?v=7nxKAtxGSn8>

## Codice

- Medium, Integrating Rasa Open Source Chatbot Into Unity [Part 1] : The Connection, Divyang Pradeep Pal,  
<https://medium.com/analytics-vidhya/integrating-rasa-open-source-chatbot-into-unity-part-1-the-connection-9ba582c804cd>
- Unity Documentation, MonoBehaviour,  
<https://docs.unity3d.com/ScriptReference/MonoBehaviour.html>
- Unity Documentation, MonoBehaviour.Awake(),  
<https://docs.unity3d.com/ScriptReference/MonoBehaviour.Awake.html>
- Unity Documentation, MonoBehaviour.Start(),  
<https://docs.unity3d.com/ScriptReference/MonoBehaviour.Start.html>
- Unity Documentation, MonoBehaviour.Update(),  
<https://docs.unity3d.com/ScriptReference/MonoBehaviour.Update.html>
- Unity Documentation, MonoBehaviour.LateUpdate(),  
<https://docs.unity3d.com/ScriptReference/MonoBehaviour.LateUpdate.html>

- Unity Forum, How to save manually save a PNG of a camera view, <https://forum.unity.com/threads/how-to-save-manually-save-a-png-of-a-camera-view.506269/>
- Microsoft Docs, Voice input in Unity, <https://docs.microsoft.com/en-us/windows/mixed-reality/develop/unity/voice-input-in-unity>
- YouTube, Dapper Dino, How to Add Voice Recognition to Your Game - Unity Tutorial, <https://www.youtube.com/watch?v=29vyEOgsW8s>
- Unity Documentation, KeywordRecognizer, <https://docs.unity3d.com/ScriptReference/Windows.Speech.KeywordRecognizer.html>
- Unity Documentation, DictationRecognizer, <https://docs.unity3d.com/ScriptReference/Windows.Speech.DictationRecognizer.html>
- Unity Documentation, Coroutines, <https://docs.unity3d.com/Manual/Coroutines.html>
- Unity Documentation, UnityWebRequest, <https://docs.unity3d.com/ScriptReference/Networking.UnityWebRequest.html>
- Unity Documentation, UnityWebRequest.error, <https://docs.unity3d.com/2019.2/Documentation/ScriptReference/Networking.UnityWebRequest-error.html>
- Unity Documentation, SerializeField, <https://docs.unity3d.com/ScriptReference/SerializeField.html>

- YouTube, Dani Krossing, HOW TO ACCESS DATA FROM ANOTHER SCRIPT 🎮 | Get Data From Other Scripts In Unity | Unity Tutorial,  
<https://www.youtube.com/watch?v=Y7pp2gzCzUI>
- YouTube, Lost Relic Games, Breaking up Code in Unity (Important game dev tips for beginners),  
[https://www.youtube.com/watch?v=\\_vj1GASSO9U](https://www.youtube.com/watch?v=_vj1GASSO9U)
- Microsoft Docs, Process Class, <https://docs.microsoft.com/en-us/dotnet/api/system.diagnostics.process?view=net-6.0>
- Stack Overflow, Kill process tree programmatically in C#,  
<https://stackoverflow.com/questions/5901679/kill-process-tree-programmatically-in-c-sharp>

## Blender

- Blender (programma),  
[https://it.wikipedia.org/wiki/Blender\\_\(programma\)](https://it.wikipedia.org/wiki/Blender_(programma))
- Manuale di Blender,  
<https://docs.blender.org/manual/en/latest/index.html>
- CGTrader, Robot head Free 3D model,  
<https://www.cgtrader.com/free-3d-models/character/sci-fi-character/robot-head-5051d388-5f77-41c0-a5d1-14e9e96af41d>
- YouTube, Blender Base Camp, Use Face Orientation To Locate Incorrect Normals And How To Fix Them - Blender 2.8 Tutorial, <https://www.youtube.com/watch?v=clzstqtN6YQ>

- YouTube, Ryan King Art, How to Use Text in Blender (Tutorial), [https://www.youtube.com/watch?v=supbk\\_5MamY](https://www.youtube.com/watch?v=supbk_5MamY)

## Python

- Python, <https://it.wikipedia.org/wiki/Python>
- Stack Overflow, How to run multiple Python versions on Windows, <https://stackoverflow.com/questions/4583367/how-to-run-multiple-python-versions-on-windows>
- YouTube, PyMike, Tutorial Python 3 - 20 - Ambienti Virtuali in Python 3 con VENV (Virtual Environments) – ITALIANO, <https://www.youtube.com/watch?v=fEPilaUGH6U>
- O'Reilly, Bill Bell, Determining if Another Instance of a Script Is Already Running in Windows,  
<https://www.oreilly.com/library/view/python-cookbook/0596001673/ch06s09.html>
- Microsoft Docs, Mutex Objects, <https://docs.microsoft.com/en-us/windows/win32/sync/mutex-objects>
- CreateMutex documentation, [http://timgolden.me.uk/pywin32-docs/win32event\\_CreateMutex\\_meth.html](http://timgolden.me.uk/pywin32-docs/win32event_CreateMutex_meth.html)
- YouTube, SMK TECH, How to use a C# dll in python project | Using Function from C# dll | Dynamic Link Library | Python, <https://www.youtube.com/watch?v=lnsTJRY1jPU>

- Microsoft Docs, SynthesizerState Enum,  
<https://docs.microsoft.com/en-us/dotnet/api/system.speech.synthesis.synthesizerstate?view=netframework-4.8>
- HTML.it, Vito Gentile, Creare un server HTTP con Python,  
<https://www.html.it/articoli/creare-un-server-http-con-python/>

## **Realizzazione delle immagini da applicare allo schermo**

- GIMP, <https://it.wikipedia.org/wiki/GIMP>
- Manuale di GIMP, <https://docs.gimp.org/2.10/en/>
- GIFER, <https://gifer.com/en/Z23b>
- GIFER, Terms of service, <https://gifer.com/en/p/tos>

## **Registro di sistema**

- Registro di sistema,  
[https://it.wikipedia.org/wiki/Registro\\_di\\_sistema](https://it.wikipedia.org/wiki/Registro_di_sistema)
- Microsoft Support, How to add, modify, or delete registry subkeys and values by using a .reg file,  
<https://support.microsoft.com/en-us/topic/how-to-add-modify-or-delete-registry-subkeys-and-values-by-using-a-reg-file-9c7f37cf-a5e9-e1cd-c4fa-2a26218a1a23>
- Appuals, How to Enable / Disable Online Speech Recognition in Windows 10?, <https://appuals.com/enable-disable-online-speech-recognition-in-windows-10/>

- Ghacks.net, Unlock all Windows 10 TTS voices system-wide to get more of them,  
<https://www.ghacks.net/2018/08/11/unlock-all-windows-10-tts-voices-system-wide-to-get-more-of-them/>

## Realizzazione dell'installer

- InstallForge, <https://installforge.net/>
- InstallForge Support, <https://installforge.net/support1/>
- YouTube, Habesha Tech Tips, How to make an Installer (Setup.exe) for your Application Software,  
<https://www.youtube.com/watch?v=cVN62zhiNH0>
- Stack Exchange, Is there a list of Windows special directories/shortcuts (like %TEMP%)?,  
<https://superuser.com/questions/217504/is-there-a-list-of-windows-special-directories-shortcuts-like-temp>
- Stack Overflow, What does %~dp0 mean, and how does it work?, <https://stackoverflow.com/questions/5034076/what-does-dp0-mean-and-how-does-it-work>
- Python, Python Releases for Windows,  
<https://www.python.org/downloads/windows/>
- Microsoft Download Center, Visual C++ Redistributable for Visual Studio 2015, <https://www.microsoft.com/en-us/download/details.aspx?id=48145>

## Altri tentativi

- To use the Microsoft Speech SDK with a C# program,  
[https://www.utm.edu/staff/bbradley/200320/cs490/adding\\_speech.html](https://www.utm.edu/staff/bbradley/200320/cs490/adding_speech.html)
- Microsoft Docs, Microsoft Speech Platform  
Use SSML to Create Prompts and Control TTS,  
[https://docs.microsoft.com/en-us/previous-versions/office/developer/speech-technologies/jj127898\(v=msdn.10\)](https://docs.microsoft.com/en-us/previous-versions/office/developer/speech-technologies/jj127898(v=msdn.10))
- Microsoft Docs, Create Grammars Using SRGS XML  
(Microsoft.Speech), [https://docs.microsoft.com/en-us/previous-versions/office/developer/speech-technologies/hh378349\(v=office.14\)](https://docs.microsoft.com/en-us/previous-versions/office/developer/speech-technologies/hh378349(v=office.14))
- Microsoft Docs, Microsoft Speech API (SAPI) 5.3,  
[https://docs.microsoft.com/en-us/previous-versions/windows/desktop/ms723627\(v=vs.85\)](https://docs.microsoft.com/en-us/previous-versions/windows/desktop/ms723627(v=vs.85))
- Microsoft Docs, SpVoice GetVoices method (SAPI 5.3),  
[https://docs.microsoft.com/en-us/previous-versions/windows/desktop/ms723601\(v=vs.85\)](https://docs.microsoft.com/en-us/previous-versions/windows/desktop/ms723601(v=vs.85))
- Microsoft Docs, SPVOICESTATUS (SAPI 5.3),  
[https://docs.microsoft.com/en-us/previous-versions/windows/desktop/ms720498\(v=vs.85\)](https://docs.microsoft.com/en-us/previous-versions/windows/desktop/ms720498(v=vs.85))
- Microsoft Docs, ISpeechVoiceStatus Interface (SAPI 5.3),  
[https://docs.microsoft.com/en-us/previous-versions/windows/desktop/ms722539\(v=vs.85\)](https://docs.microsoft.com/en-us/previous-versions/windows/desktop/ms722539(v=vs.85))

- Microsoft Docs, ISpeechVoiceStatus RunningState Property (SAPI 5.3), [https://docs.microsoft.com/en-us/previous-versions/windows/desktop/ms722545\(v=vs.85\)](https://docs.microsoft.com/en-us/previous-versions/windows/desktop/ms722545(v=vs.85))
- Microsoft Docs, SpVoice Interface (SAPI 5.3), [https://docs.microsoft.com/en-us/previous-versions/windows/desktop/ms723602\(v=vs.85\)](https://docs.microsoft.com/en-us/previous-versions/windows/desktop/ms723602(v=vs.85))
- Microsoft Docs, SpVoice Status property (SAPI 5.3), [https://docs.microsoft.com/en-us/previous-versions/windows/desktop/ms723612\(v=vs.85\)](https://docs.microsoft.com/en-us/previous-versions/windows/desktop/ms723612(v=vs.85))
- Blender, Primitives, <https://docs.blender.org/manual/en/latest/modeling/meshes/primitives.html>
- StackExchange, What is the difference between a UV Sphere and an Icosphere?, <https://blender.stackexchange.com/questions/72/what-is-the-difference-between-a-uv-sphere-and-an-icosphere>
- YouTube, Olav3D Tutorials, [2.8] Blender Tutorial: Quick Glass in EEVEE, <https://www.youtube.com/watch?v=SN9lkS7K04k>
- YouTube, UGuruz, Realistic Glass Material in Unity HDRP, <https://www.youtube.com/watch?v=XTnobfLSTV0>
- YouTube, Brackeys, BUILD LEVELS QUICKLY in Unity with SNAPS!, <https://www.youtube.com/watch?v=b4oqOdBCy3c>

- Unity Documentation, SelectionMode.Editable,  
<https://docs.unity3d.com/ScriptReference/SelectionMode.Editable.html>
- Unity Documentation, Material,  
<https://docs.unity3d.com/ScriptReference/Material.html>
- Unity Documentation, Color Constructor,  
<https://docs.unity3d.com/ScriptReference/Color-ctor.html>
- Unity Manual, Fixing lightmap UV overlap,  
<https://docs.unity3d.com/Manual/ProgressiveLightmapper-UVOverlap.html>
- YouTube, Coco Code, Get started with UI Builder (UI toolkit)  
- time to ditch old UI system?,  
<https://www.youtube.com/watch?v=NQYHIH0BJbs>
- Unity Documentation, VisualElement,  
<https://docs.unity3d.com/ScriptReference/UIElements.VisualElement.html>
- Unity Documentation, GUI.Label,  
<https://docs.unity3d.com/ScriptReference/GUI.Label.html>
- Unity Documentation, Selectable.OnPointerUp,  
<https://docs.unity3d.com/2017.3/Documentation/ScriptReference/UI.Selectable.OnPointerUp.html>
- Microsoft Developer Support, Getting Started with Windows  
10 IoT Core & Raspberry Pi 3B+,  
<https://devblogs.microsoft.com/premier-developer/getting-started-with-windows-10-iot-core-raspberry-pi-3b/>

- YouTube, Microsoft IoT Developers, Getting Started - Windows 10 IoT Core + Raspberry Pi 3,  
<https://www.youtube.com/watch?v=JPRUbGIyODY>
- Microsoft Docs, Using PowerShell for Windows IoT,  
<https://docs.microsoft.com/en-us/windows/iot-core/connect-your-device/powershell>
- Microsoft Docs, Python, <https://docs.microsoft.com/en-us/windows/iot-core/developer-tools/python>
- Rasa Community Forum, Running RASA on the RPi 4 with Raspbian Buster!, <https://forum.rasa.com/t/running-rasa-on-the-rpi-4-with-raspbian-buster/20805>
- Shell Scripting Tutorial, <https://www.shellscript.sh/>
- YouTube, gotbletu, VOSK Offline Speech Recognition, Speech To Text for Linux Android iOS Mac OSX Windows,  
<https://www.youtube.com/watch?v=xAdqA1prU5s>
- Microsoft Docs, Using .NET 4.x in Unity,  
<https://docs.microsoft.com/en-us/visualstudio/gamedev/unity/unity-scripting-upgrade>
- Unity Documentation, The Unity linker,  
<https://docs.unity3d.com/Manual/unity-linker.html>
- Unity Documentation, Managed code stripping,  
<https://docs.unity3d.com/Manual/ManagedCodeStripping.html>
- GitHub, Debugging Unity Games,  
<https://github.com/dnSpy/dnSpy/wiki/Debugging-Unity-Games>

- BobH's Blog, How to use dnspy to debug a unity game dynamically, <https://bobh.mkaliez.com/archives/use-dnspy-to-debug-unity-code-en.html>
- Repository dnSpy-Unity-mono, <https://github.com/wh0am15533/dnSpy-Unity-mono>
- Microsoft Docs, Use Visual Studio Tools for Unity, <https://docs.microsoft.com/en-us/visualstudio/gamedev/unity/get-started/using-visual-studio-tools-for-unity?pivots=windows>
- Stack Overflow, Building a DLL for Unity with Mono, <https://stackoverflow.com/questions/59689081/building-a-dll-for-unity-with-mono>
- pyttsx3, pyttsx3 - Text-to-speech x-platform, <https://pyttsx3.readthedocs.io/en/latest/>
- pypi.org, pyttsx3, <https://pypi.org/project/pyttsx3/>
- YouTube, TechWorld with Nana, Dockerfile Tutorial - Docker in Practice || Docker Tutorial 10, <https://www.youtube.com/watch?v=WmcDMiyqfZs>
- Microsoft Docs, Dockerfile on Windows, <https://docs.microsoft.com/en-us/virtualization/windowscontainers/manage-docker/manage-windows-dockerfile>
- Elton Stoneman, How to Dockerize Windows Applications: The 5 Steps, <https://blog.sixeyed.com/how-to-dockerize-windows-applications/>