Alma Mater Studiorum
Università di Bologna
Campus di Cesena

SCUOLA DI INGEGNERIA E ARCHITETTURA

Corso di Laurea Magistrale in Ingegneria e Scienze Informatiche

MecWilly 3D: supporto all'orientamento topografico nei bambini in età scolare attraverso un serious game per la raccolta differenziata

Relatore:
Dott.ssa Silvia Mirri

Presentata da:
Matteo Corradin

Sessione III
Anno Accademico 2015-2016

Introduzione

I robot e gli applicativi per smartphone e tablet assumono un ruolo sempre più attivo a supporto dello sviluppo e potenziamento delle abilità nei bambini.
A partire dagli ultimi anni della scuola dell'infanzia (4-5 anni) una delle abilità più importanti che affrontano i bambini è l'orientamento topografico, con particolare riferimento alla relatività delle posizioni (destra e sinistra) degli oggetti rispetto ad un determinato punto. Tali posizioni, infatti, non sono assolute e nemmeno sono caratteristiche intrinseche degli oggetti, bensì sono relative rispetto ad una determinata posizione. L'egocentrismo tipico di quell'età, fra l'altro, rende difficile al bambino il pensare a punti di vista alternativi rispetto ai propri, nonché il mettersi "nei panni" dell'altro. L'utilizzo delle ICT e della robotica, associate al conflitto socio-cognitivo offrono molteplici soluzioni per supportare al meglio l'acquisizione e il potenziamento di tali abilità di orientamento spaziale a partire già dai 5 anni.
Una delle procedure più comuni per indurre un conflitto socio-cognitivo è quella di far svolgere un compito a due persone che devono mettersi d'accordo e discutere i loro punti di vista per raggiungere una soluzione comune.
In questo contesto entra in gioco MecWilly, il quale funge da mezzo scatenante di questo processo.
MecWilly è un robot umanoide alto 1 metro e 20 cm, realizzato con materiali economici e di recupero, grazie alla collaborazione con il Dipartimento di Psicologia dell'Università di Bologna e il Comune di Rimini. Grazie ad apposite fotocamere (posizionate in corrispondenza degli occhi) è in grado di

riconoscere le persone dai tratti del viso, seguirle con lo sguardo, capire se gli si sta sorridendo, concentrare l'attenzione su particolari zone dell'ambiente circostante.
Il robot MecWilly viene utilizzato per studiare il conflitto socio-cognitivo nei bambini in età pre-scolare e scolare.
I bambini devono interagire con MecWilly impartendogli degli ordini pre-concordati fra loro, per raggiungere un obiettivo comune. In particolare, MecWilly è stato utilizzato nell'ambito di progetti che hanno lo scopo di supportare i bambini nel miglioramento delle loro abilità di orientamento topografico e al tempo stesso nell'educazione alla raccolta differenziata e al riciclo dei rifiuti.
In questo progetto MecWilly si muove in una griglia di gioco all'interno del quale sono presenti dei bidoni per la raccolta differenziata. Lo scopo del gioco è condurre MecWilly, che ha in mano un oggetto, verso il bidone corretto. L'esperimento consiste nel valutare il tempo impiegato per raggiungere l'obiettivo e il numero totale di mosse.
Un videogioco bidimensionale per tablet è stato realizzato con lo scopo di verificare se esiste una differenza in termini di risultati utilizzando un ambiente virtuale, anzichè reale. La versione in 2D però non riproduceva fedelmente l'esperimento originale, svolto in un ambiente reale (una classe della scuola primaria). Nello specifico, essendo la app sviluppata su un piano bidimensionale, il punto di osservazione del bambino era posto sopra il labirinto e non di fronte al robot, facendo venire meno alcuni aspetti fondamentali del conflitto (creato dal posizionamento frontale del bambini rispetto al robot).
L'obettivo di questa tesi è realizzare una applicazione 3D che ha la finalità di sviluppare e migliorare le abilità nell'orientamento topografico (mappe) e nell'orientamento spaziale (contesto) dei bambini della scuola primaria.
L'applicazione 3D, che riproduce fedelmente l'esperimento originale, offre la possibilità di poter mettere a confronto l'efficacia dell'applicazione utilizzata su tablet con quella in ambiente reale, in quanto il bambino è posizionato frontalmente rispetto al robot e, dunque, ha lo stesso punto di vista creato

nella situazione reale.
Il progetto è la realizzazione di un videogioco 3D che simuli il funzionamento e le dinamiche dell'esperimento con il robot reale.
Il presente documento di tesi è strutturato nel seguente modo: nel Capitolo 1 viene analizzato il rapporto fra videogiochi e educazione; nel Capitolo 2 viene illustrato il progetto, da dove nasce l'idea, le motivazioni che portano alla necessità del progetto e gli strumenti software utilizzati; nel Capitolo 3 viene trattato tutto ciò che riguarda l'implementazione del progetto, vengono dapprima trattati i modelli 3D che servono per il videogioco e successivamente il progetto in Unity, nel Capitolo 4 viene presentato un caso d'uso reale in cui il videogioco è stato utilizzato.

Indice

Elenco delle figure

Capitolo 1

Videogiochi e educazione

> *Coloro che fanno distinzione fra intrattenimento e educazione forse non sanno che l'educazione deve essere divertente e il divertimento deve essere educativo*
>
> Marshall McLuhan

Nel 2014 ha fatto il giro del mondo uno speech, durante uno dei tanti Ted Talks tenuti in giro per il mondo, di Cordelle Steiner, uno studente di terza elementare (presso la Matoska International Elementary in White Bear Lake), che quotidianamente in classe utilizza videogiochi come strumento frontale educativo sotto la guida del professore Ananth Pai [22].
La sempre maggiore accessibilità economica verso gli educational games, l'arrivo in età scolare di una generazione ormai nata con i videogiochi come forma primaria di intrattenimento, la proliferazione di startup attive nel settore e l'intraprendenza di alcune scuole rappresentano dei fattori determinanti per la crescita di questo settore.
Dal miliardo e mezzo di dollari generato nel 2012 dai prodotti game based learning, è previsto un passaggio a 2.3 miliardi nell'arco del 2017 con una crescita dell'8.3% anno su anno. Si prevede che paesi come Cina, Stati Uniti, India, Indonesia, Brasile, Corea del Sud, Giappone domineranno il mercato con una grandissima richiesta di questi nuovi supporti digitali per la formazione, soprattutto mobile-oriented. Ormai quasi tutte le learning apps

dedicate a studenti della scuola primaria includono e implementano elementi di gaming o di gamification [2].

1.1 Edutainment

L'edutainment (a volte tradotto come "intrattenimento educativo") è una forma di intrattenimento finalizzata sia ad educare sia a divertire.
Il termine edutainment è un neologismo coniato da Bob Heyman mentre produceva documentari per la società National Geographic. L'espressione è nata dalla fusione delle parole educational (educativo) ed entertainment (divertimento). In alcuni contesti viene tradotto anche come "divertimento educativo". Il termine edutainment è stato utilizzato inizialmente per indicare le forme di comunicazione giocosa-ludica finalizzate alla didattica. Il concetto si è successivamente esteso a tutto quanto può essere comunicato e appreso, attraverso il gioco, in modo produttivo e costruttivo.
Il termine edutainment è, inoltre, usato per riferirsi a quel settore dell'e-learning che cerca di trasmettere dei concetti chiave in modo divertente. Questo metodo può essere fruttuosamente utilizzato per trattare argomenti delicati come l'etica, la diversità e l'uguaglianza, la prevenzione dell'abuso di sostanze, l'educazione sessuale (includendo informazioni sul rischio di contagio di HIV e AIDS), ecc.
Questo tipo di educazione può essere molto utile per raffinare certe sensibilità particolarmente adatte alla società contemporanea, soprattutto grazie al meccanismo della metamorfosi della propria identità che avviene in tutti i giochi e può favorire il decentramento identitario auspicato da molte strategie educative, come ad esempio quello dell'educazione interculturale [8].

1.2 Serious game

I serious game ("giochi seri") sono giochi digitali che non hanno esclusivamente o principalmente uno scopo di intrattenimento, ma contengono

elementi educativi [19].
L'origine dei giochi a scopo formativo viene generalmente ricondotta alle simulazioni di guerra ("Kriegsspiel") dell'esercito prussiano degli inizi del XVIII secolo o ai giochi da tavola della prima metà del Novecento come Monopoly.
I serious game oggi trovano applicazione in numerosi contesti [14], come ad esempio la sensibilizzazione su temi importanti e la formazione, nonché varie attività promozionali e campagne sociali.
In contesti commerciali i serious game vengono solitamente utilizzati per simulare situazioni di vendita faccia a faccia e telefonica, interazioni in occasione di colloqui o di riunioni e, in generale, tutte quelle situazioni in cui si rende necessaria un'esperienza diretta per assimilare contenuti e comportamenti. Per ciò che attiene agli usi militari, il serious game trova applicazione sia nell'addestramento del personale tecnico (quali operatori radar, personale di macchina a bordo delle navi, piloti di aerei ed elicotteri) che nella simulazione di operazioni militari complesse come i cosiddetti war game. Tuttavia, si è anche proposto di introdurre l'uso di serious game nella formazione formale [19].

Molti videogiochi, anche se non concepiti originariamente come serious game, possono trovare uso in ambito educativo. Un esempio è l'uso di giochi di successo come Civilization e Assassin's Creed per l'insegnamento della storia [10].

1.3 GWAP

Un gioco con uno scopo (in inglese Game With A Purpose, da cui la sigla GWAP) è una tecnica di calcolo umano a base di passaggi di outsourcing all'interno di un processo di calcolo per gli esseri umani in modo divertente (gamification) [12] [23].
Luis von Ahn ha proposto per primo l'idea dei GWAP. L'idea è quella di sfruttare l'intelletto umano per affrontare problemi che i computer non sono

ancora in grado di affrontare da soli.
Le attività presenti in un GWAP sono di solito triviali per gli umani, ma complesse per le macchine, come ad esempio interpretazione di testi antichi non riconoscibili da software OCR, il riconoscimento di immagini o la risoluzione di rompicapi logici.
Tutte queste attività, generalmente noiose, sono inserite in un contesto che fa uso di elementi di gamification.
Caratteristiche comuni dei GWAP [12]:

- Timed response. Proporre un limite di tempo, da battere, rende più stimolante e soddisfacente il gioco. Definire obiettivi stimola il giocatore a migliorarsi e ad entrare in competizione con se stesso o gli altri giocatori.

- Score keeping. Assegnare dei punti al conseguimento di obiettivi di gioco, come la fine di un livello, incentiva l'utente a continuare a giocare.

- Player skill levels. Un sistema di progressione del giocatore è un elemento motivazionale comune nei GWAP. Completando livelli e sbloccando determinati obiettivi il giocatore accumula punti che fanno aumentare il suo grado. All'inizio del gioco al giocatore viene assegnato il livello più basso.

- High-score lists. L'utilizzo di una classifica che elenca i migliori giocatori, ad esempio ordinati per numero di punti o per livello, è un altro elemento comune nei GWAP. Possono essere proposte numerosi tipi di classifiche in base al tipo di gioco o all'effetto che si vuole ottenere, ad esempio classifiche orarie, giornaliere, mensili e annuali.

- Randomness. La casualità è un elemento da tenere in considerazione nello sviluppo dei GWAP, ad esempio in un gioco multiplayer la casualità nella scelta dello sfidante introduce unicità alla partita e contrasta comportamenti scorretti (come ad esempio partite truccate). Casualità

nella generazione dei livello o dei rompicapi proposti introduce incertezza in quanto il giocatore non sa cosa dovrà fare per superare il livello; questo motiva anche il continuare a giocare per quei giocatori che già sono esperti.

Due esempi di GWAP, sviluppati presso la Carnegie Mellon University, sono il gioco ESP e Peekaboom. Questi due giochi dimostrano come gli esseri umani, mentre giocano, possono risolvere problemi che i computer non possono ancora risolvere.

1.3.1 ESP Game

Il riconoscimento delle immagini è un esempio di un compito difficile da eseguire in modo autonomo per i computer, ma molto più facile per gli esseri umani. Molte applicazioni online, come i motori di ricerca o programmi per l'assistenza per i non vedenti, hanno bisogno di una descrizione accurata delle immagini. La quantità di immagini presenti sul web aumenta ogni giorno di 1800 milioni di foto [6], e la tecnologia non può ancora determinare con precisione il loro contenuto. Molte delle tecniche utilizzate per determinare il contenuto delle immagini sono inadeguate poichè ipotizzano che il contenuto delle immagini in una pagina Web sia legato al testo adiacente, sfortunatamente questo non sempre è corretto, ragion per cui il riconoscimento manuale è l'unico metodo per poter fornire una descrizione accurata delle immagini [24].
L'ESP Game è un gioco online di riconoscimento delle immagini. Il gioco accoppia casualmente due giocatori, un giocatore non conosce l'identità dell'altro e non possono comunicare fra di loro. L'unica cosa che hanno in comune è un'immagine. L'obettivo del gioco è indovinare quale etichetta è stata associata all'immagine da parte dell'altro giocatore. I giocatori possono scrivere una parola o una frase da sottoporre al gioco. Una volta che tutti e due i giocatori hanno scritto la stessa parola il round termina e viene sottoposta una nuova immagine. I giocatori non devono scrivere la parola nello stesso momento, è sufficiente che ad un certo punto si inserisca la stessa

Figura 1.1: ESP Game - Agreeing on an image

parola scritta dall'altro giocatore. I giocatori possono inserire tutte le parole o frasi che vogliono, infatti più ne inseriscono più aumenta la probabilità di raggiungere l'obiettivo. Il processo di scrivere la stessa stringa è chiamato "agreeing on an image" 1.1. I giocatori sono chiamati ad indovinare più associazioni possibili fino a un massimo di 15 immagini in 2 minuti e mezzo di tempo, si ricevono un certo quantitativo di punti per ogni associazione più un bonus se si raggiunge una intesa per tutte e 15. Per aumentare il livello di sfida alcune immagini possono essere associate a delle parole proibite, taboo, che il giocatore non può utilizzare, in questo caso vengono assegnati più punti se si raggiunge l'intesa. Per incrementare le chance di vittoria, il giocatore deve imparare a pensare come potrebbe pensare un'altra persona, risulta per questo motivo che la stringa sulla quale si raggiunge l'intesa è una buona descrizione per l'immagine. L'esperimento è diventato estremamente popolare tanto che in pochi mesi erano già state indicizzate più di 10 milioni di immagini [24] [13].

1.3.2 Peekaboom

Peekaboom è un gioco online che sfrutta i dati ottenuti dall'ESP Game per determinare la precisa posizione degli oggetti nell'immagine, identificando quali pixels appartengono a un determinato oggetto dell'immagine, così da generare informazioni utili per l'addestramento e il test di algoritmi di

visione artificiale.
Anche in questo gioco i giocatori vengono accoppiati casualmente, a uno dei due viene assegnato il ruolo di "Peek" e uno il ruolo "Boom". Peek inizialmente vede uno schermo bianco mentre Boom vede l'immagine completa e una descrizione. I dati in Peekaboom provengono da quelli ottenuti dall'ESP Game. L'obettivo del giocatore con il ruolo di Peek è quello di indovinare la parola associata all'immagine che Boom gli mostra pian piano, infatti ogni volta che Boom clicca sull'immagine rivela una porzione della stessa all'altro giocatore, il quale può quindi tentare di indovinare. Anche in questo caso, ogni qual volta si indovina la parola, vengono assegnati dei punti, il tempo limite è 4 minuti, meno pixels vengono rivelati da Boom più punti si ottengono. Non appena si indovina la parola i ruoli dei due giocatori si invertono. Intuitivamente, per massimizzare i punti, i giocatori sono incentivati a rivelare solo le aree dell'immagine necessarie per una corretta identificazione. In questo modo si ottengono preziose informazioni utili per affinare gli algoritmi di visione artificiale [24].

1.4 Gamification

La gamification è l'utilizzo di elementi mutuati dai giochi e delle tecniche di game design in contesti esterni ai giochi [20] [1] [9] [7].
Il termine deriva dalla parola "Game", cioè gioco, anche associato al semplice divertimento senza scopi particolari. Traendo vantaggio dall'interattività concessa dai mezzi moderni ed ovviamente dai principi alla base del concetto stesso di divertimento, la Gamification rappresenta uno strumento estremamente efficace in grado di veicolare messaggi di vario tipo, a seconda delle esigenze, e di indurre a comportamenti attivi da parte dell'utenza, permettendo di raggiungere specifici obiettivi, personali o d'impresa. Al centro di questo approccio va sempre collocato l'utente ed il suo coinvolgimento attivo [1].

1.4.1 Elementi base dalla gamification

Gli elementi e le meccaniche di gioco che caratterizzano la gamification sono in continua evoluzione e coinvolgono anche gli studi condotti dall'industria dei videogioco in materia di game design.
Troviamo comunque alcuni elementi base che identificano un prodotto o un gioco in cui sono presenti elementi della gamification [4] [5] [21].

- Punti
  La raccolta punti è uno strumento per aumentare la partecipazione dell'utente. Gli utenti sono spinti a compiere determinate azioni per guadagnarne sempre di più. I punti possono essere utilizzati per ottenere ricompense, oggetti di gioco o premi. Questo meccanismo induce l'utente a pensare che il tempo speso per guadagnare punti è tempo ben speso.

- Livelli
  Il livello, inteso come rango, classe o qualifica, organizzato in modo gerarchico, caratterizza e classifica il giocatore., il quale è spinto ad accumulare punti o completare determinate azioni per poter passare di grado.
  Questo processo stimola l'utente a impegnarsi sempre di più perché ha un reale ritorno visivo del proprio stato gerarchico.

- Beni virtuali
  La possibilità di acquistare dei beni virtuali, sia con valuta reale che con valuta virtuale, spinge l'utente a impegnarsi nel gioco per ottenere un determinato oggetto che permetterà di distinguersi dagli altri giocatori. L'acquisto di beni virtuali con valuta reale rappresenta un modo per ottenere un profitto da parte degli sviluppatori.

- Classifiche
  Le classifiche aggiungono competizione al gioco, i giocatori sono stimo-

lati a raggiungere posizioni sempre più alte per raggiungere la prima posizione.

Capitolo 2

Specifiche di progetto

2.1 MecWilly

2.1.1 Chi è MecWilly?

MecWilly è un robot umanoide alto 1 metro e 20 cm, realizzato con materiali economici e di recupero, grazie alla collaborazione con il Dipartimento di Psicologia dell'Università di Bologna e il Comune di Rimini. Grazie ad apposite fotocamere (posizionate in corrispondenza degli occhi) è in grado di riconoscere le persone dai tratti del viso, seguirle con lo sguardo, capire se gli si sta sorridendo, concentrare l'attenzione su particolari zone dell'ambiente circostante. Grazie ad una sintesi vocale è in grado di dialogare autonomamente con i suoi interlocutori su argomenti generici. Inoltre, può essere addestrato su specifici argomenti conservando comunque la sua capacità di rispondere a qualunque richiesta generica. MecWilly è wireless e connesso ad Internet ed è possibile chiedergli di prelevare in tempo reale informazioni come previsioni del tempo, news, o qualunque altra informazione utile. Funziona a batteria (che lo rende privo di fili) con un'autonomia di circa 4 ore. Può funzionare anche tramite rete elettrica e, nel frattempo, ricaricare la batteria. E' provvisto di un touch screen sul retro che permette di tener sotto controllo il sistema e di gestire impostazioni e configurazioni [18].

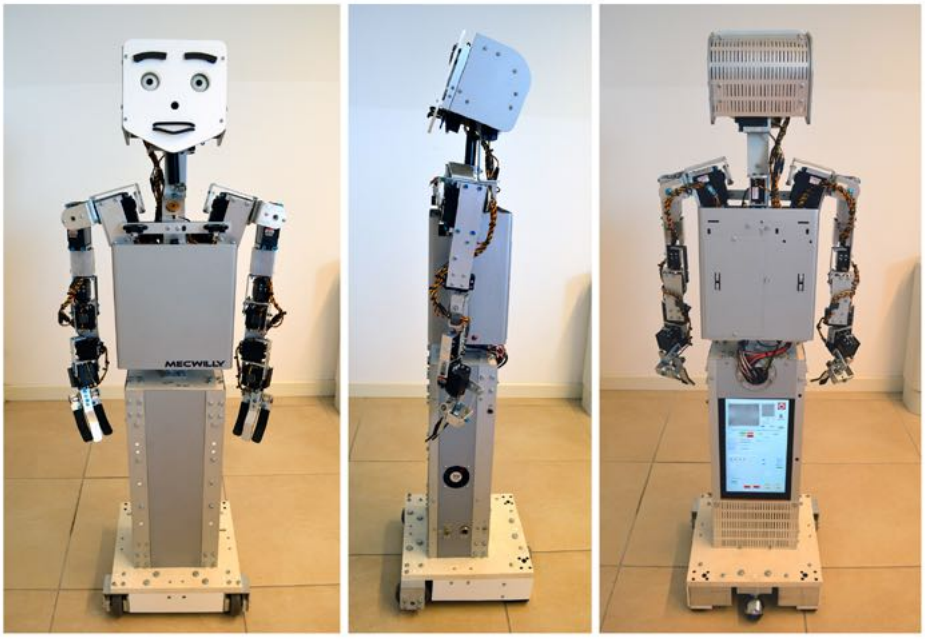

Figura 2.1: Il robot MecWilly

2.1.2 Utilizzo di MecWilly

Background teorico e metodologico

I robot e gli applicativi per smartphone e tablet assumono un ruolo sempre più attivo a supporto della costruzione e potenziamento delle abilità nei bambini [16] [17]. A partire dagli ultimi anni della scuola dell'infanzia (4-5 anni) una delle abilità più importanti che affrontano i bambini è l'orientamento topografico, con particolare riferimento alla relatività delle posizioni (destra e sinistra) degli oggetti rispetto ad un determinato punto. Tali posizioni, infatti, non sono assolute e nemmeno sono caratteristiche intrinseche degli oggetti, bensì sono relative rispetto ad una determinata posizione. L'egocentrismo tipico di quell'età, fra l'altro, rende difficile al bambino il pensare a punti di vista alternativi rispetto ai propri, nonché il mettersi "nei panni" dell'altro. L'utilizzo delle ICT e della robotica, associate al conflitto sociocognitivo offrono molteplici soluzioni per supportare al meglio l'acquisizione

e il potenziamento di tali abilità di orientamento spaziale a partire già dai 5 anni.

Conflitto socio-cognitivo

Nell'ambito della psicologia sociale genetica il conflitto viene ad assumere una forte valenza comunicativa e sociale. Lo sviluppo e la maturazione della capacità conoscitiva è fortemente sostenuta da situazioni in cui i soggetti si trovano a sperimentare contrasti di informazioni e opinioni nell'interazione con altri individui. Nel corso di attività sociali significative, i partner sono chiamati a individuare una soluzione comune ad un compito che costituisce lo scopo della loro interazione, che diventa patrimonio del singolo dopo una prima fase di elaborazione in comune delle risposte. L'esito comunicativo che caratterizza tali attività o esperienze è definito conflitto socio-cognitivo. Il conflitto socio-cognitivo è stato studiato in modo particolare nei bambini nella fascia di età 4/6 anni, quando i soggetti si trovano ai livelli iniziale o intermedio nell'acquisizione di una determinata nozione. In tale periodo, infatti, l'articolazione tra attività sociali e funzionamenti cognitivi si evidenzia in modo particolare nel corso di interazioni durante le quali i bambini sono sollecitati a osservare, comprendere il comportamento altrui [15].

Il conflitto socio-cognitivo è un processo interattivo nel quale le persone rielaborano e riorganizzano il loro punto di vista nella discussione di una idea con qualun altro.
Il conflitto socio-cognitivo consta di una situazione di problem solving dalla quale emergono differenti punti di vista, proposti dagli attori, per giungere alla soluzione. La negoziazione che ne deriva, per arrivare a una decisione condivisa in merito alla soluzione da adottare, fa sì che, in futuri compiti simili, le conoscenze e competenze dei partner migliorino sensibilmente. L'aspetto interessante, evidenziato dai tanti studi effettuati sui bambini, è che il miglioramento non avviene solo nel bambino che si trova ad avere meno conoscenze, ma in tutti i partecipanti all'interazione.

Una delle procedure più comuni per indurre un conflitto socio-cognitivo è quella di far svolgere un compito a due persone che devono mettersi d'accordo e discutere i loro punti di vista per raggiungere una soluzione comune.
In questo contesto entra in gioco MecWilly, il quale funge da mezzo scatenante di questo processo.

MecWilly

Infatti, il robot MecWilly, tra l'altro, viene utilizzato per studiare il conflitto socio-cognitivo nei bambini in età pre-scolare e scolare.
I bambini devono interagire con MecWilly impartendogli degli ordini preconcordati fra loro, per raggiungere un obiettivo comune. In particolare, MecWilly è stato utilizzato nell'ambito di progetti che hanno lo scopo di supportare i bambini nel miglioramento delle loro abilità di orientamento topografico e al tempo stesso nell'educazione alla raccoltà differenziata e al riciclo dei rifiuti [16] [17].
In questo progetto MecWilly si muove in una griglia di gioco all'interno del quale sono presenti dei bidoni per la raccolta differenziata. Lo scopo del gioco è condurre MecWilly, che ha in mano un oggetto, verso il bidone corretto.
I bambini devono impartire gli ordini al robot attraverso la voce, MecWilly eseguirà le azioni solamente nel caso in cui i bambini siano concordi sull'ordine da eseguire.
L'esperimento consiste nel valutare il tempo impiegato per raggiungere l'obiettivo e il numero totale di mosse.

2.1.3 Esperimenti software precedenti

Così come un partner può fungere da supporto e aiutare il bambino a raggiungere risultati che non raggiungerebbe individualmente, la tecnologia può assumere lo stesso ruolo e, quindi, portare un bambino ad agire all'interno della propria zona di sviluppo prossimale [11] per arrivare a risultati cui non arriverebbe da solo.
Un videogioco bidimensionale per tablet è stato realizzato con lo scopo di

verificare se esiste una differenza in termini di risultati utilizzando un ambiente virtuale, anzichè reale [3].
La versione in 2D però non riproduceva fedelmente l'esperimento originale, svolto in un ambiente reale (una classe della scuola primaria). Nello specifico, essendo la app sviluppata su un piano bidimensionale, il punto di osservazione del bambino era posto sopra il labirinto e non di fronte al robot, facendo venire meno alcuni aspetti fondamentali del conflitto (creato dal posizionamento frontale del bambini rispetto al robot). L'applicazione 3D, che riproduce fedelmente l'esperimento originale, offre invece la possibilità di poter mettere a confronto l'efficacia dell'applicazione utilizzata su tablet con quella in ambiente reale, in quanto il bambino è posizionato frontalmente rispetto al robot e, dunque, ha lo stesso punto di vista creato nella situazione reale.

2.2 MecWilly 3d

L'applicazione MecWilly 3D ha la finalità di sviluppare e migliorare le abilità nell'orientamento topografico (mappe) e nell'orientamento spaziale (contesto) dei bambini della scuola primaria.
Il progetto è la realizzazione di un videogioco 3D che simuli il funzionamento e le dinamiche dell'esperimento con il robot reale.

2.3 Modellazione 3d

La modellazione tridimensionale è il processo atto a definire una forma tridimensionale in uno spazio virtuale generata su computer. Generalmente la modellazione rappresenta il primo step di una serie di operazioni successive che determineranno il prodotto finale:

1. Modellazione 3D primaria (studio preliminare dell'elemento da realizzare ed eventuali elaborati preparatori).

2. Modellazione 3D secondaria (finalizzata a dettagliare il modello).
3. Surfacing (definizione dei materiali di superficie).
4. Mappatura (definizione delle coordinate di proiezione).
5. Applicazione delle Texture.
6. Inserimento dello scheletro.
7. Skinning del modello.
8. Definizione della postura del modello.
9. Allestimento scenico.
10. Illuminazione della scena.
11. Rendering della scena.
12. Salvataggio dell'immagine in un file grafico.
13. Output finale.

2.4 Software

Per quanto riguarda la modellazione tridimensionale sono stati presi in considerazione due dei principali software di modellazione, uno dal mondo open source e uno di tipo proprietario, a pagamento.

2.4.1 Autodesk 3ds Max

3ds Max è un programma di grafica vettoriale tridimensionale e animazione, realizzato dalla divisione Media & Entertainement di Autodesk.
3ds Max è uno dei software più utilizzati per la creazione 3D per numerose ragioni tra cui le potenti capacità di editing e la sua architettura di plugin. Infatti anche se molti strumenti non sono parte del prodotto, il prodotto dispone di una grande scelta di plugin realizzati da terze parti.

2.4.2 Blender

Blender è un software libero e multipiattaforma di modellazione, rigging, animazione, compositing e rendering di immagini tridimensionali. Benchè non abbia tutti gli strumenti di 3dsMax, le funzionalità di modellazione sono complete e ricche.

2.4.3 Scelta del software

Sia con 3ds Max che con Blender è possibile realizzare i modelli necessari per il progetto. Autodesk 3dsMax mette a disposizione degli studenti una versione completa e gratuita. La comunità online, la documentazione disponibile e il supporto fornito allo sviluppo sono stati alla base della scelta di Autodesk 3dsMax come strumento per la modellazione degli elementi del progetto.

2.5 Terminologia

Nel seguito di questa sezione si riportano i termini che riguardano tipicamente il mondo della modellazione e della grafica 3D destinata al real-time.

Mesh Identifica una geometria completa. Questo termine può essere considerato sinonimo del termine "geometria" e sinonimo del termine "modello".

Scena Rappresenta l'insieme di tutte le geometrie che sono visibili in nel mondo tridimensionale.

Render/Rendering (Renderizzazione) Renderizzare significa calcolare l'effetto visivo del nostro lavoro di modellazione tramite un motore di rendering. E' l'operazione di calcolo (che compie il processore e la GPU) dei riflessi, delle luci sui diversi materiali e di tutti gli elementi della scena.

Engine (Motore di rendering) Per engine si intende il motore di rendering utilizzato per mostrare e far muovere una scena sullo schermo.

Texture È un'immagine bidimensionale in formato raster che viene riprodotta su una o più facce di un modello poligonale tridimensionale.

Shader Il termine shader indica uno strumento della computer grafica 3D che generalmente è utilizzato per determinare l'aspetto finale della superficie di un oggetto. Consiste essenzialmente in un insieme di istruzioni. Gli shader servono per riprodurre il comportamento fisico del materiale che compone l'oggetto cui sono applicati.

Real-Time È il metodo di visualizzazione che renderizza "al momento" e in modo veloce la scena in cui ci troviamo, prerogativa di un videogioco 3D.

Un engine che renderizza in real-time esegue milioni di calcoli e restituisce la scena renderizzata decine di volte al secondo.

Un engine in tempo reale lavora con modelli a basso numero di poligoni (low-poly) per rendere il più possibile fluido e veloce il movimento della scena.

Viewport Identifica una delle aree in cui si svolge il lavoro di 3dsMax. Tipicamente, se non configurato diversamente dall'utente, Max utilizza quattro viewport che riguardano diverse angolazioni con cui si inquadra la stessa scena.

Wireframe È un metodo di visualizzazione di una geometria o di un'intera scena tridimensionale senza la renderizzazione delle superfici ma soltanto nella sua struttura di "sostegno" senza renderizzare i poligoni tra un "filo e l'altro".

Modifiers (Modificatore) È uno dei termini più importanti di 3dsMax perché caratterizza il modo che usa questo programma per operare su una geometria. Applicando i modificatori a una geometria è possibile

andare a modificare il modello tridimensionale secondo varie modalità, definite dal tipo di modificatore.

Vertex Rappresenta l'elemento più piccolo di una geometria. L'insieme dei vertici determina la forma della geometria, essi sono il vero punto cardine di un modello perchè dalla loro posizione dipende la posizione di ogni altro elemento. I vertici identificano una posizione e non hanno dimensione.

Edge Elemento di una geometria che unisce due vertici. Un edge è in pratica un lato di un poligono. Un edge è sempre una linea retta e non può essere curvo. Durante la visualizzazione in modalità wireframe essi saranno l'unica cosa visibile della geometria. Gli edge rappresentano lo scheletro di una mesh.

Face È la superficie più piccola di una geometria. Si tratta della superficie generata da tre vertici e dai tre edge che li uniscono. È l'elemento di cui tener conto durante il renderig in tempo reale, quando si tratta di contare i poligoni di una geometria è dei triangoli che si sta parlando. Maggiore è il numero di triangoli presenti in una scene e maggiore sarà il tempo di rendering.

Poly/Poligon (Poligono) Un poligono è formato da uno o più triangoli. E' l'insieme di triangoli che definiscono una piccola superficie. Di solito durante la modellazione si lavora con i poligoni e con il modificatore Edit Poly.

2.6 Definizione di motore grafico

Il motore grafico è il nucleo software di un videogioco, tipicamente un motore di gioco include un motore di rendering, un motore fisico, suono, scripting, animazioni, IA, networking.
Per il progetto sono stati valutati i tre principali engine per lo sviluppo di videogiochi 3D, come illustrato nel seguito di questo capitolo.

2.7 Scelta dell'engine

2.7.1 Unity

Unity è uno strumento di authoring integrato multipiattaforma per la creazione di videogiochi 3D o altri contenuti interattivi, quali visualizzazioni architettoniche o animazioni 3D in tempo reale.
L'ambiente di sviluppo Unity gira sia su Microsoft Windows sia su macOS, e i giochi che produce possono essere eseguiti su Windows, Mac, Linux, web browser, Xbox 360, PlayStation 3, PlayStation Vita, Wii, iPad, iPhone, Android, Playstation 4, Xbox One e Wii U.
Unity è stato uno dei primi motori grafici ad essere pensato esclusivamente (o quasi) per gli sviluppatori indie, per questo si è conquistato una grossa fetta di mercato grazie alle sue numerose funzioni di export multi piattaforma, alla possibilità di programmare con 3 linguaggi diversi (C#, Javascript e Boo) e alla sua interfaccia intuitiva.

2.7.2 Unreal Engine 4

Unreal Engine è una suite completa di strumenti di sviluppo di videogiochi prodotto da Epic Games. Il codice sorgente dell'engine è di libero accesso e personalizzabile.
Unreal Engine rappresenta lo stato dell'arte per quanto riguarda le tecnologie per lo sviluppo dei videogiochi, i più grandi successi in campo videoludico degli ultimi 15 anni sono basati su questo motore di gioco.
Così come per Unity, in UE c'è completo supporto per realizzare videogiochi multipiattaforma.
Dal 2015 Unreal Engine è gratuito nella sua versione base, come Unity. Questo lo ha portato ad entrare in diretta competizione con Unity in quanto grazie a un modello di business scalabile, non più proibitivo e sostenibile solamente dalle grandi software house, sta guadagnando fette di mercato tra gli sviluppatori indie. Il linguaggio di sviluppo supportato è C++.

2.7.3 Amazon Lumberyard

Amazon Lumberyard è un engine cross-platform (Windows, iOS, Android e console) gratuito basato sull'architettura del CryEngine di Crytek, dal quale eredita tutte le caratteristiche. L'engine è integrato con i servizi AWS (Amazon Web Services), il codice sorgente è disponibile gratuitamente con limitazione sull'uso da parte dello sviluppatore. Amazon Lumberyard è un engine molto giovane e al momento le risorse disponibili sono ancora poche ma dato il peso dell'azienda che lo sviluppa e gli ingenti capitali investiti non si fatica a credere che in un prossimo futuro possa competere con gli altri.

2.7.4 Scelta

Sia Unity che Unreal Engine forniscono fin dalla loro versione gratuita tutte le caratteristiche necessarie per lo sviluppo del progetto. In termini di documentazione disponibile, risorse, tutorial, comunità online più attive, considerando anche il target del progetto, Unity risulta vincente.
Benchè sia Unity che Unreal Engine permettano di effettuare il deploy su piattaforma iOS e Android, Unity è più leggero ed è compatibile anche con i dispositivi più economici o datati; in particolare su iOS abbiamo il supporto fino a iOS 7.0, quindi iPhone 4 e iPad 2.
Unity permette il deploy anche come web app e questo è un elemento importante per la decisione finale.
Per questi motivi si è scelto di approfondire e utilizzare Unity.

2.8 Unity

2.8.1 Interfaccia

L'interfaccia principale di Unity è costituita da 5 elementi principali che possono essere riorganizzati raggruppati e spostati a piacere (ad eccezione

della toolbar).
I 5 elementi principali sono 2.2:

- Project Window.
  In questa finestra vengono riportati tutti gli asset del progetto.

- Scene View.
  Questa finestra permette di muoversi all'interno di una scena e di modificarla; è possibile impostare la modalità di visualizzazione (2d o 3d) in base al tipo di progetto.

- Hierarchy Window.
  Viene riportata la rappresentazione gerarchica di ogni oggetto presente nella scena.

- Inspector Window.
  Permette di vedere e modificare tutte le proprietà di un oggetto.

- Toolbar.
  Contiene gli elementi essenziali per poter manipolare una scena, un oggetto e per controllare l'esecuzione della scena.

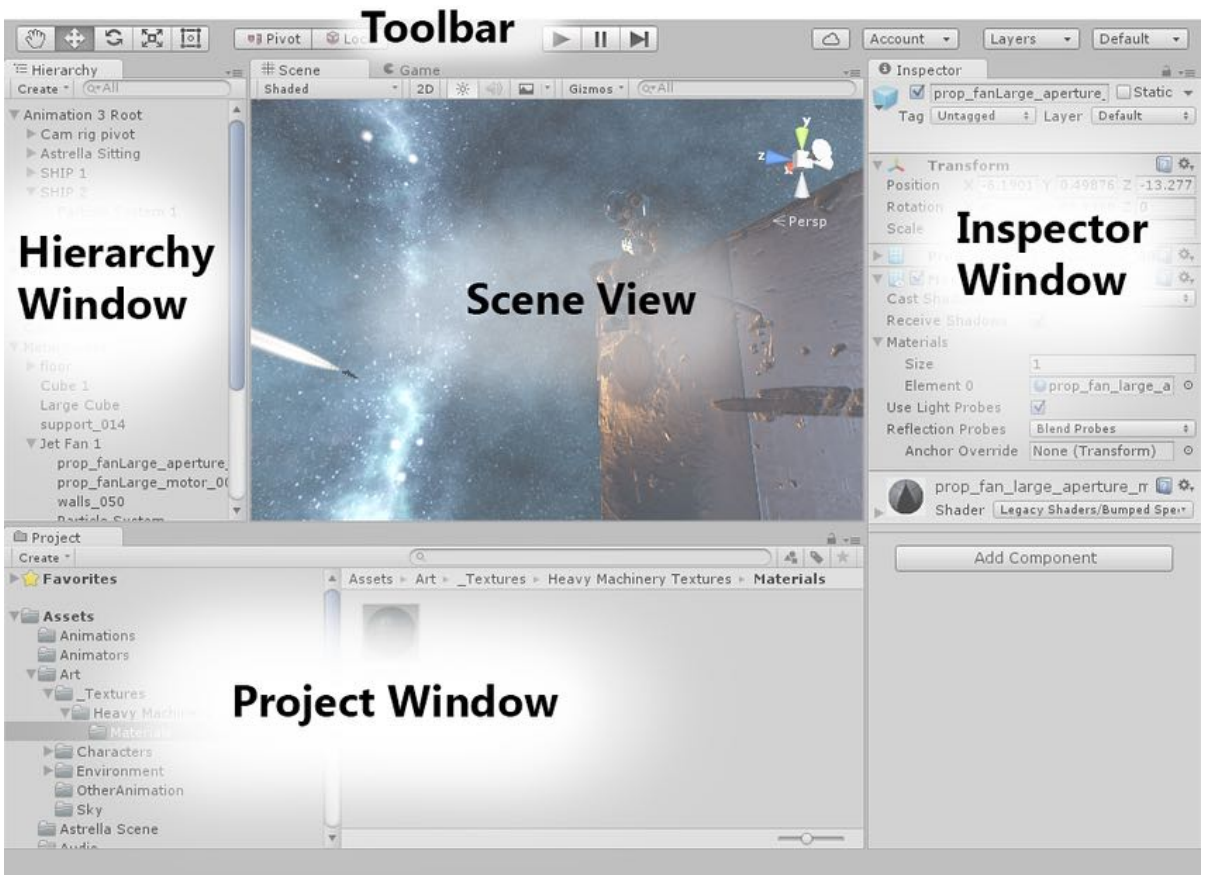

Figura 2.2: Unity - Elementi dell'interfaccia

2.8.2 Scene

La scena è l'elemento che contiene gli oggetti di gioco. Una scena può rappresentare un menù iniziale oppure un livello di gioco e più in generale qualunque schermata di gioco. In ogni scena è possibile posizionare degli oggetti di gioco.

2.8.3 GameObjects

Un Game Object, un oggetto di gioco, è l'elemento più importante in Unity. Ogni oggetto nel gioco è un GameObject. Questo significa che tutto quello che è all'interno di un gioco deve essere un GameObject. Un GameO-

bject ha delle proprietà e una posizione all'interno della scena.
Un GameObject è un contenitore; è possibile aggiungere elementi per alterarne le caratteristiche; questi elementi sono chiamati componenti.
In base al tipo di oggetto che si vuole creare è possibile aggiungere diverse combinazioni di componenti. Unity mette a disposizione una grande varietà di componenti già pronti all'uso ed è possibile creare dei componenti custom.

2.8.4 Prefabs

All'interno di un videogioco possiamo avere diversi elementi che si ripetono, in questo caso Unity mette a disposizione i prefab. I prefab sono un tipo di risorsa che permette di memorizzare un GameObject insieme ai suoi componenti e le sue proprietà. Il vantaggio nell'utilizzo dei prefabs è dato dal fatto che anche se inseriamo all'interno di una scena diversi prefab, una modifica al prefab si ripercuote su tutti gli elementi.

2.8.5 Cameras

Proprio come le telecamere utilizzate nei film per mostrare lo svolgersi della storia agli spettatori, le camere in Unity sono utilizzate per mostrare il mondo di gioco al giocatore. In una Scena deve esserci sempre almeno una camera.

2.8.6 Lights

Le luci sono una parte essenziale di ogni scena. Mentre le mesh e le texture definiscono la forma e il look di una scena, le luci definiscono il colore e l'umore di un ambiente tridimensionale.

Capitolo 3

Implementazione

In questo capitolo verranno illustrate le scelte implementative effettuate descrivendo tutti gli elementi che compongono il videogioco.

3.1 Modelli 3d

L'obiettivo del videogioco che sarà oggetto del progetto di tesi, è quello di aiutare MecWilly a fare la raccolta differenziata; avremo quindi bisogno dei modelli tridimensionali di vari oggetti, come, ad esempio, i bidioni, varie tipologie di rifiuti e il robot stesso.

3.1.1 Bottiglia di vetro

La bottiglia di vetro è stata realizzata partendo da una linea che ne rappresenta il contorno. Alla linea è stato applicato il modificatore Lathe.
Lathe è un modificatore che crea un oggetto 3D ruotando una forma (una shape come una linea in questo caso) attorno ad un asse.
Alla bottiglia è stato applicato un materiale semi-trasparente con tonalità verde scuro.

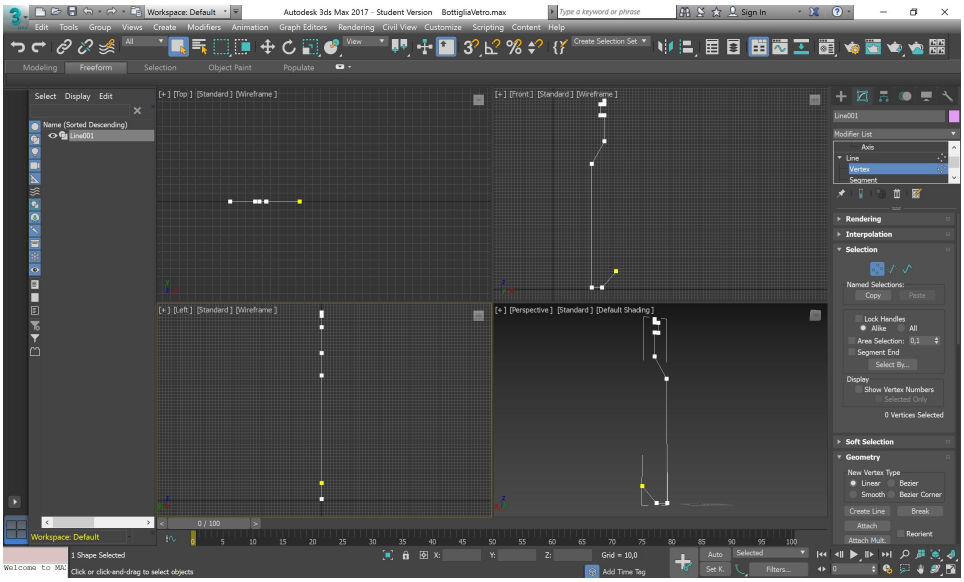

Figura 3.1: Bottiglia di vetro - contorno

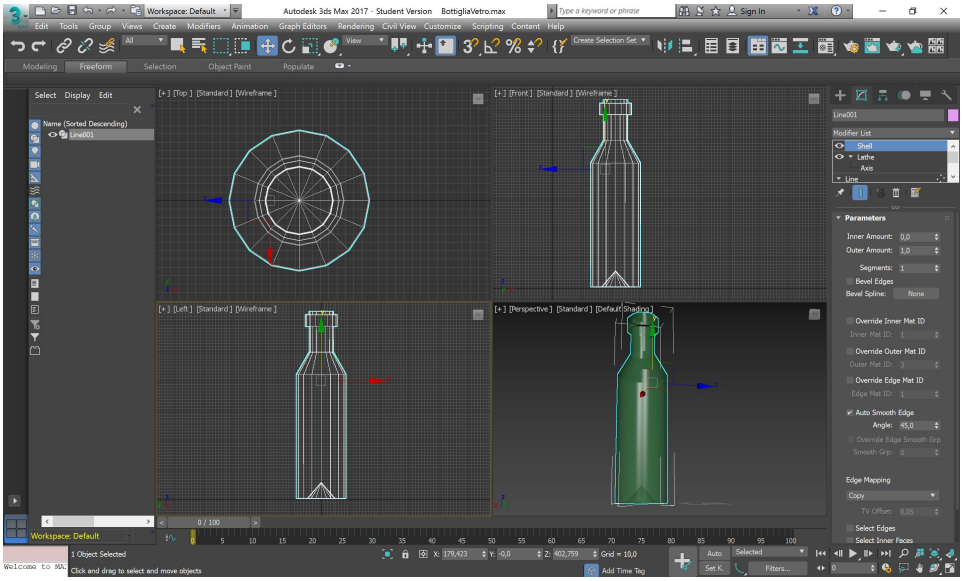

Figura 3.2: Bottiglia di vetro - Lathe

3.1.2 Lattina

La lattina è stata realizzata con la stessa tecnica utilizzata per la bottiglia di vetro. 3.4
Per aiutare la modellazione è stata utilizzata un'immagine di riferimento che rappresenta l'oggetto che vogliamo modellare. 3.3
Il supporto per la linguetta di apertura è stato realizzato partendo dal contorno 3.5, così come per gli elementi precedenti, a cui è stato applicato il modificatore Extrude il quale aggiunge profondità a un elemento. 3.6

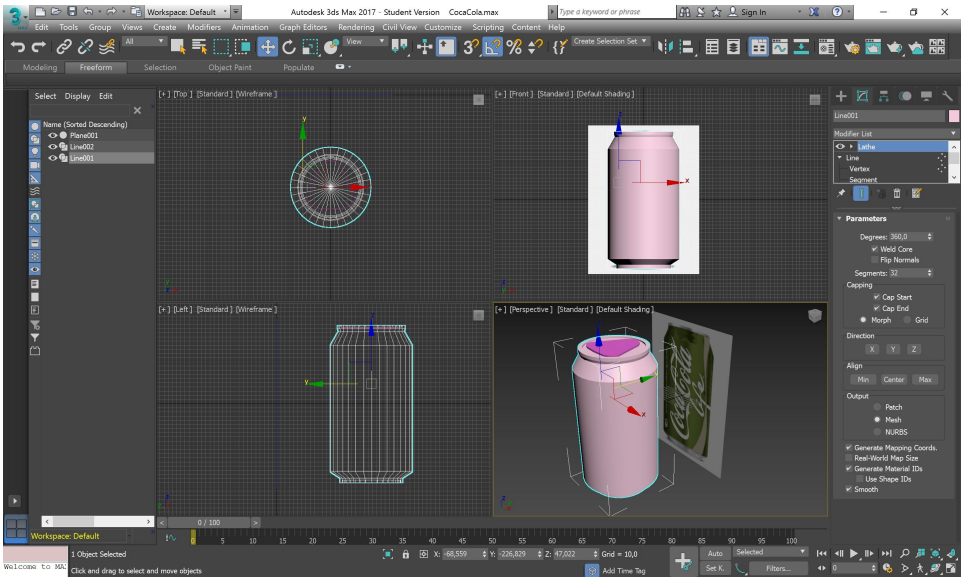

Figura 3.3: Lattina - Completa con immagine di riferimento

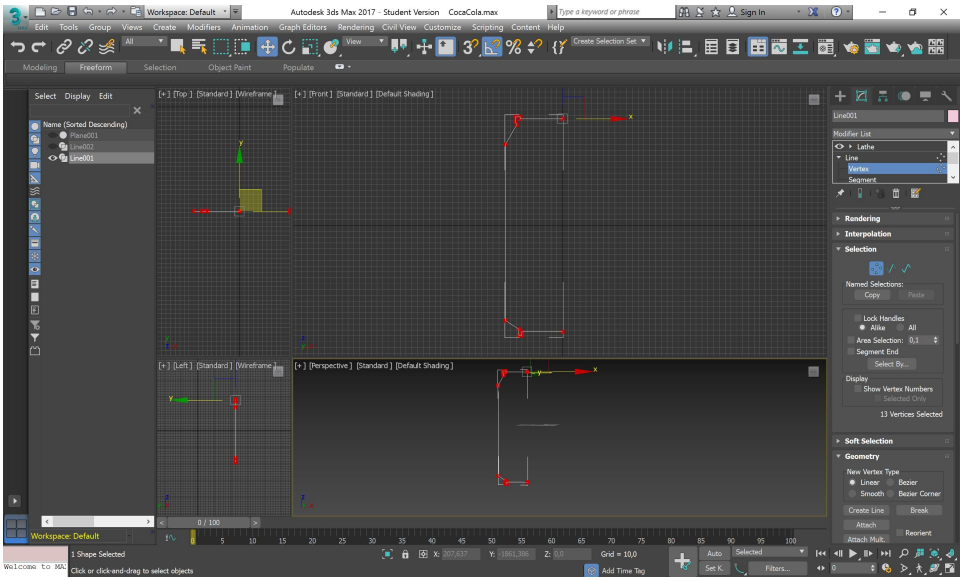

Figura 3.4: Lattina - Contorno

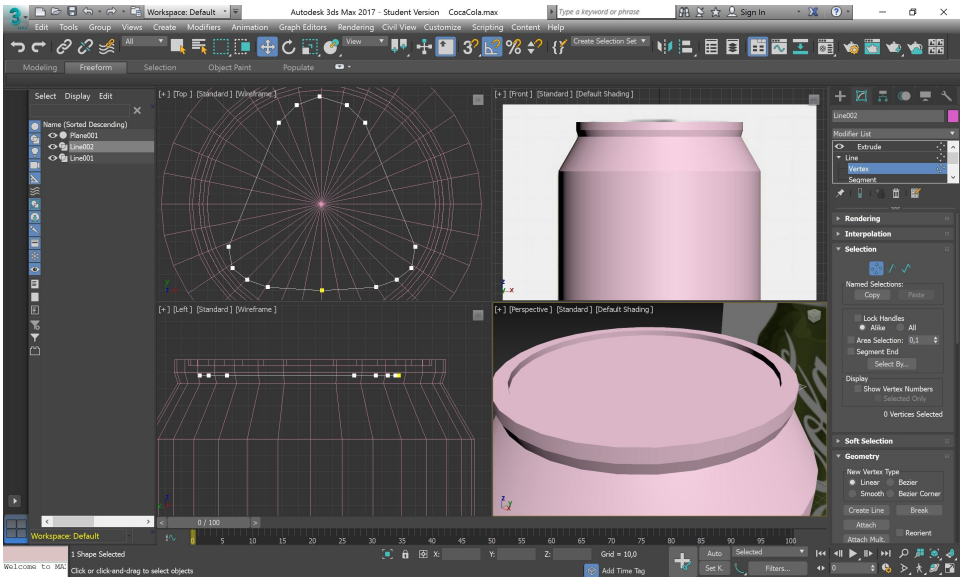

Figura 3.5: Lattina - Contorno supporto

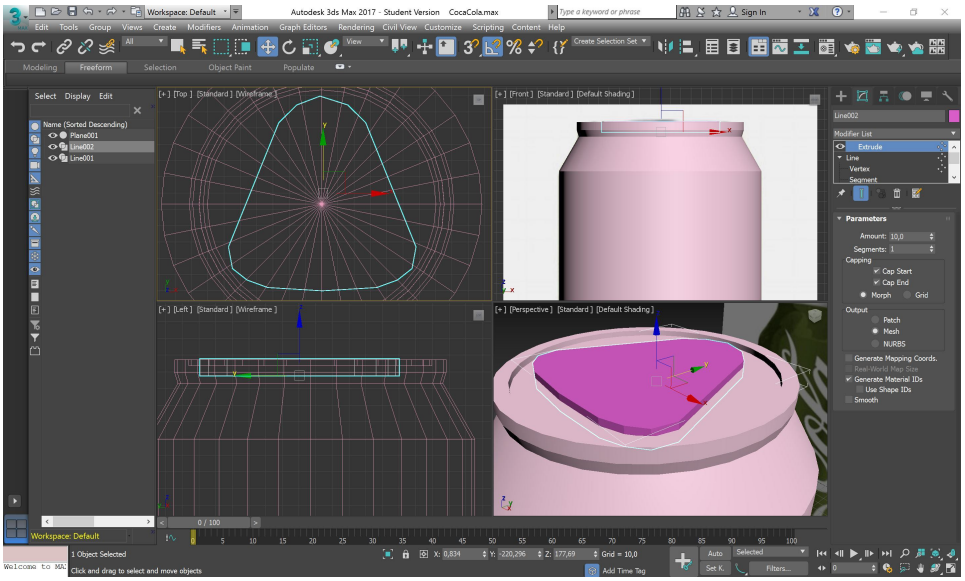

Figura 3.6: Lattina - Contorno estruso

3.1.3 MecWilly primo modello

MecWilly sarà il robot controllato dal giocatore, grazie agli esperimenti e alle prove che hanno permesso di realizzare i modelli precedenti, è stata creata una versione sperimentale di MecWilly 3.10, realizzata utilizzando le tecniche precedenti.
Partendo dalle fotografie del robot originale sono stati applicati dei filtri per mettere in evidenza il contorno del robot e rendere le immagini più semplici da utilizzare durante la modellazione. 3.7, 3.8, 3.9

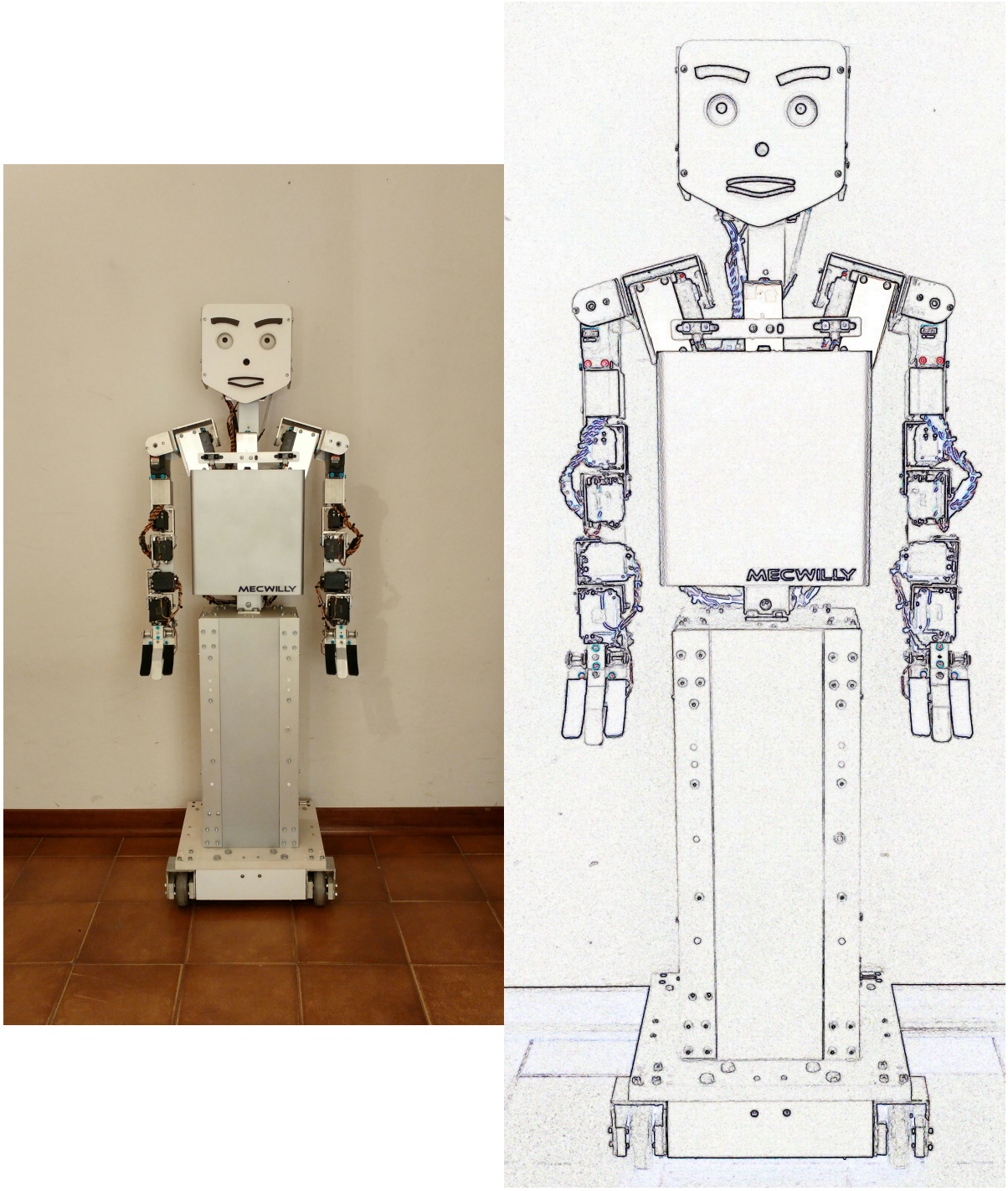

Figura 3.7: MecWilly - Fronte

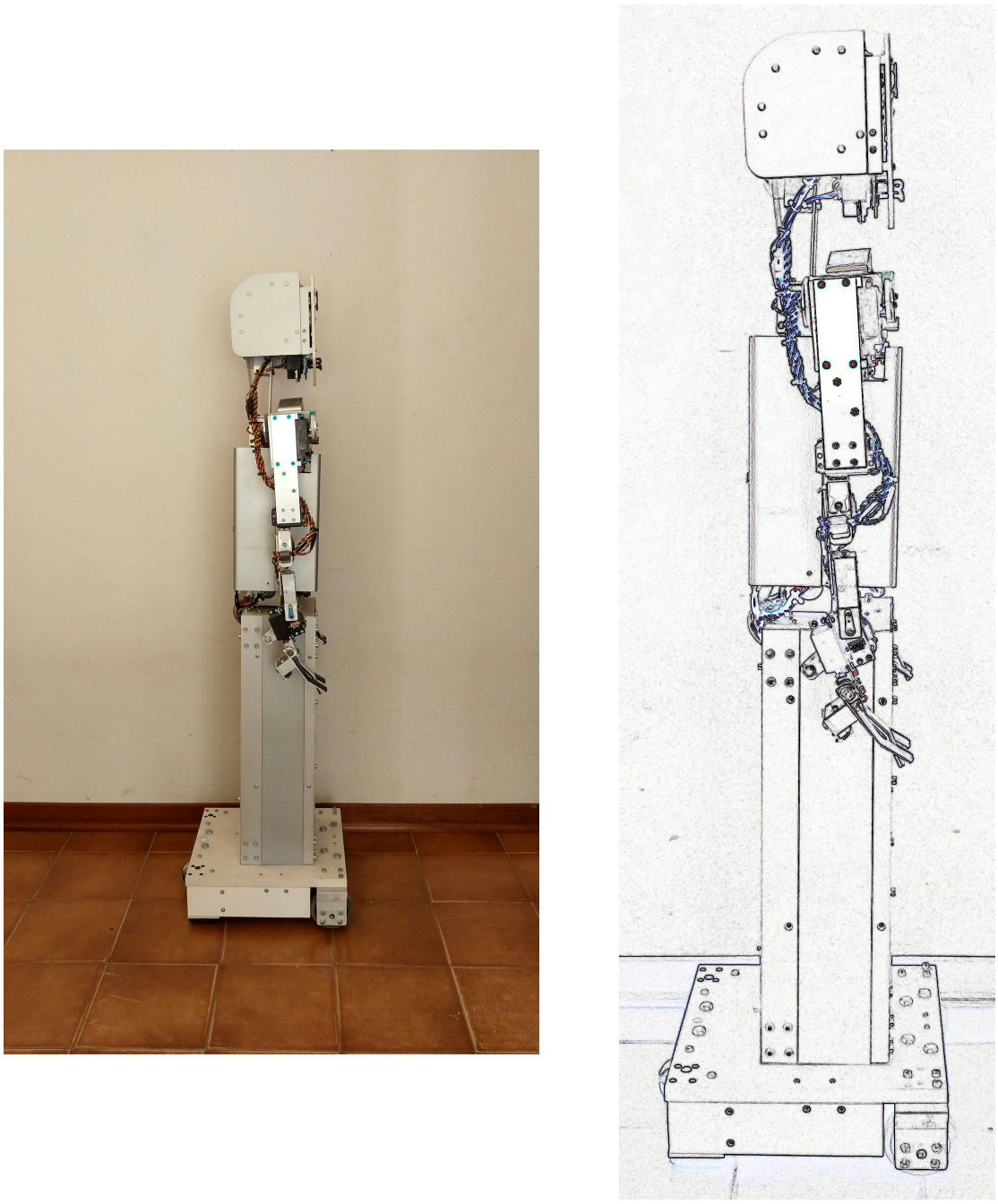

Figura 3.8: MecWilly - Lato

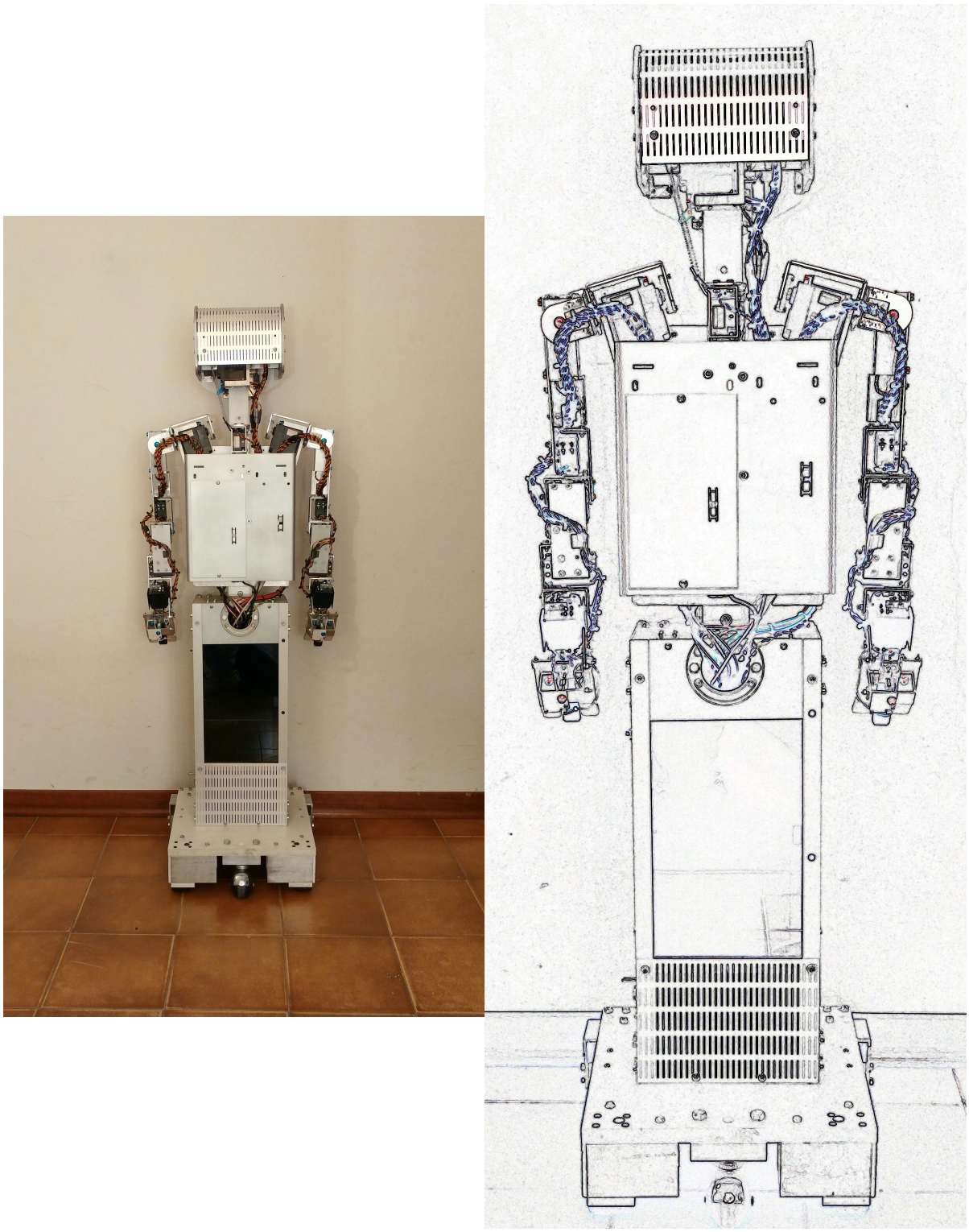

Figura 3.9: MecWilly - Retro

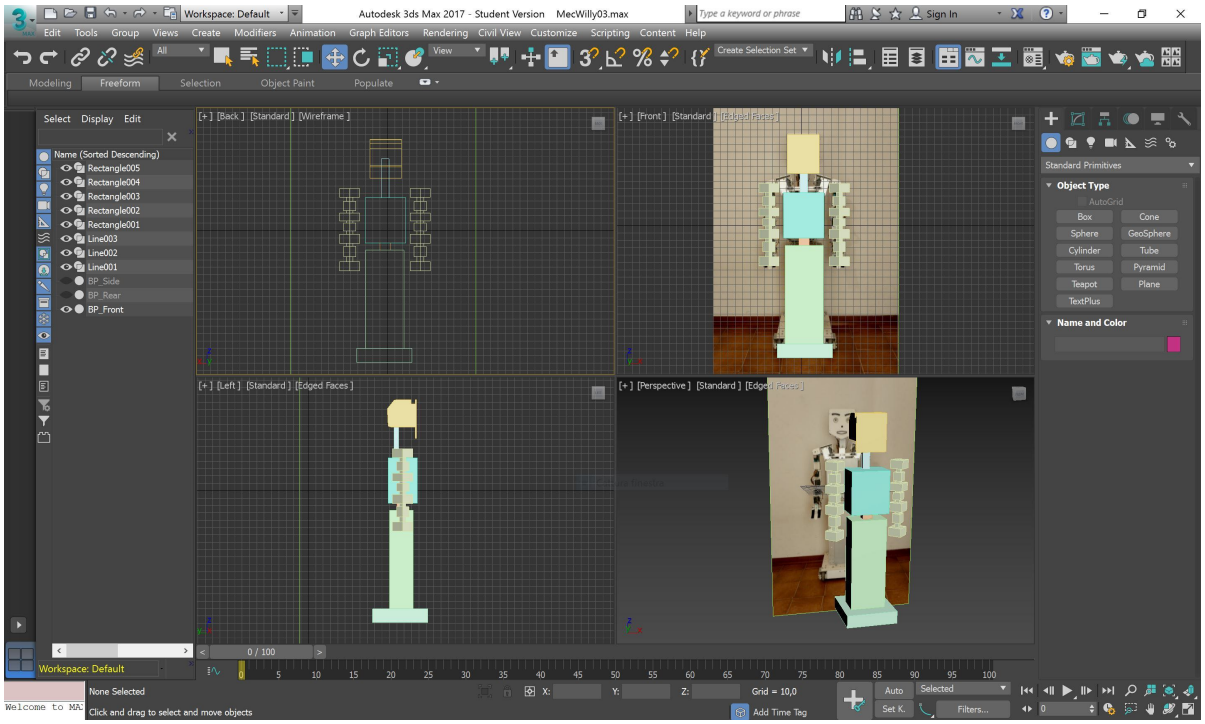

Figura 3.10: MecWilly

3.1.4 Banana

Come rifiuto organico si è scelto di modellare una banana.
Per realizzarla è stata utilizzata la tecnica del lofting.
Lofting è una tecnica importante per la creazione di oggetti 3D. Si crea un oggetto che funge da percorso al quale saranno collegate delle forme che ne andranno a formare il volume. 3.11 3.12 3.13

3.1.5 Bottiglia di plastica

La bottiglia di plastica è stata realizzata con la stessa tecnica utilizzata per la bottiglia di vetro. Il corpo è stato realizzato partendo da un profilo bidimensionale al quale è stato applicato il modificatore Lathe.
Il tappo e il filetto sono stati realizzati partendo da cerchi bidimensionali al quale è stato applicato un modificatore di estrusione. 3.14

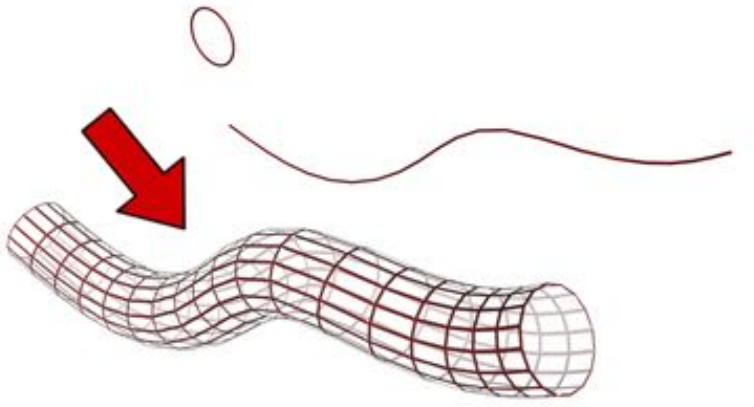

Figura 3.11: Lofting 1

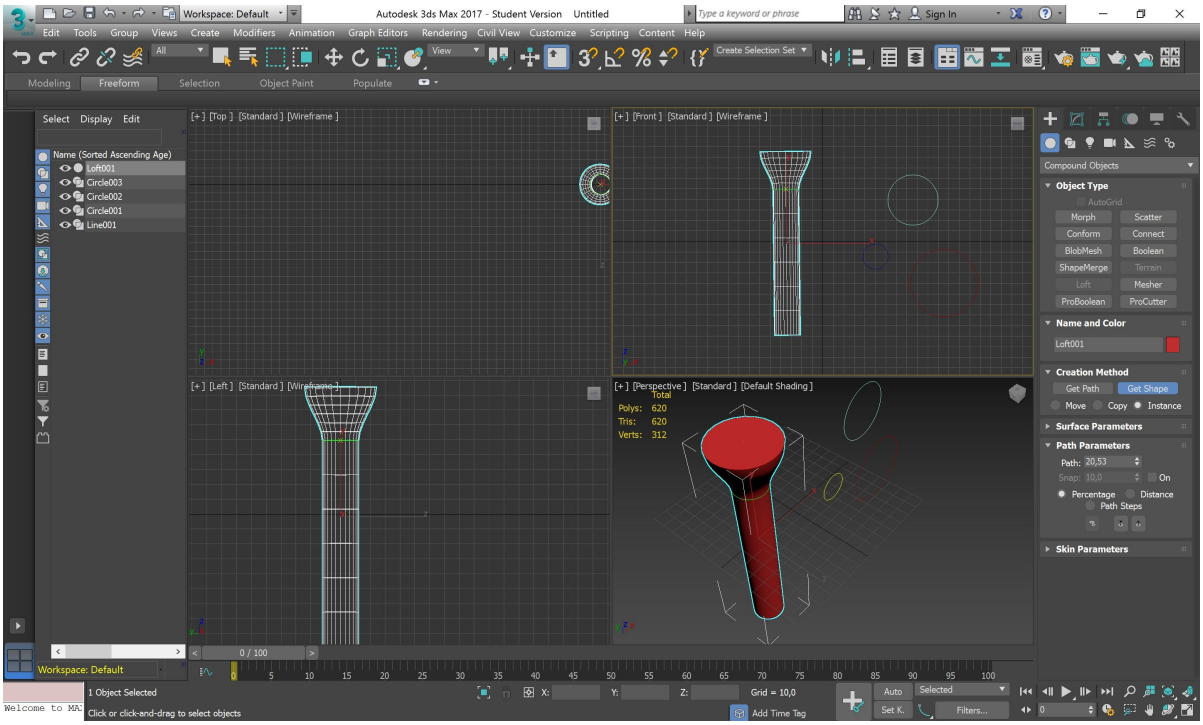

Figura 3.12: Lofting 2

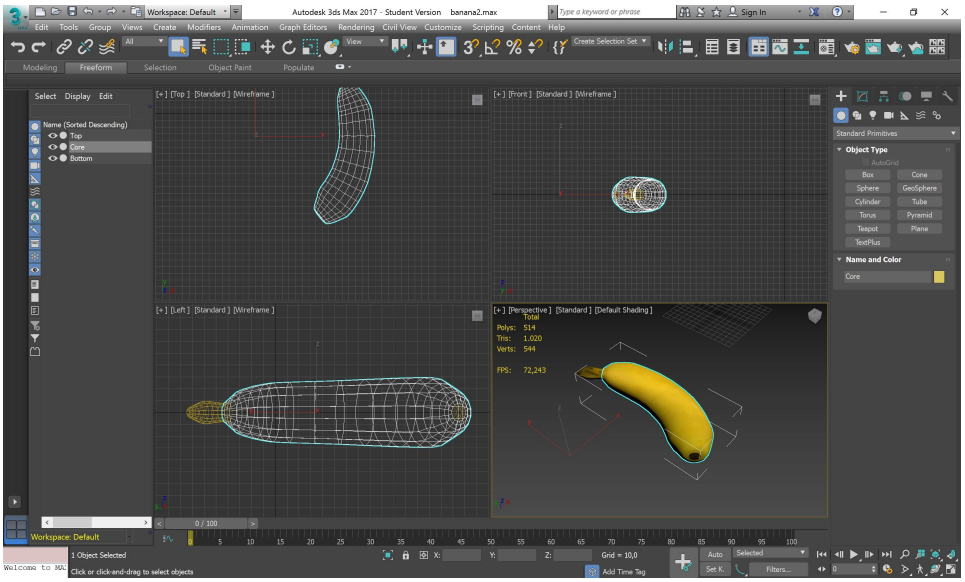

Figura 3.13: Banana

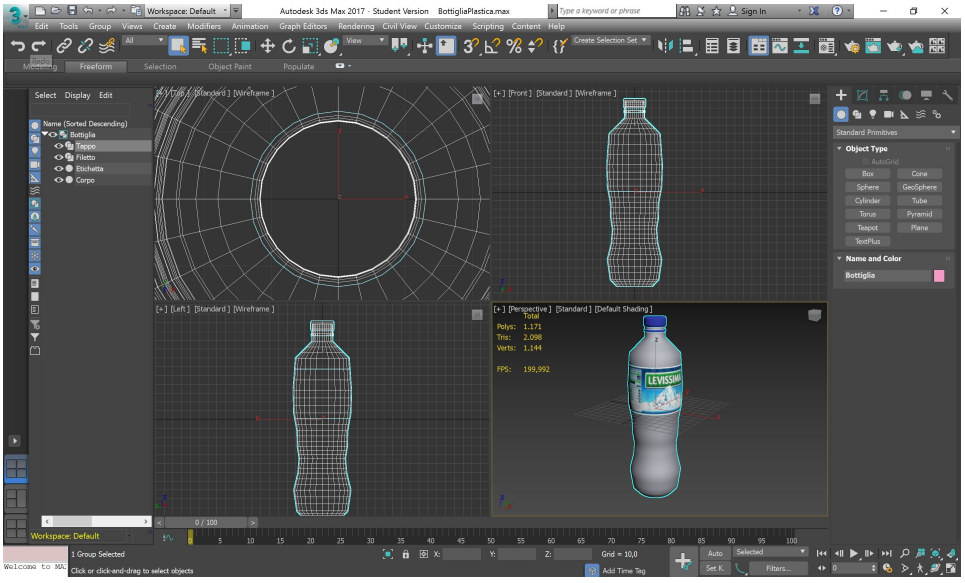

Figura 3.14: Bottiglia di plastica

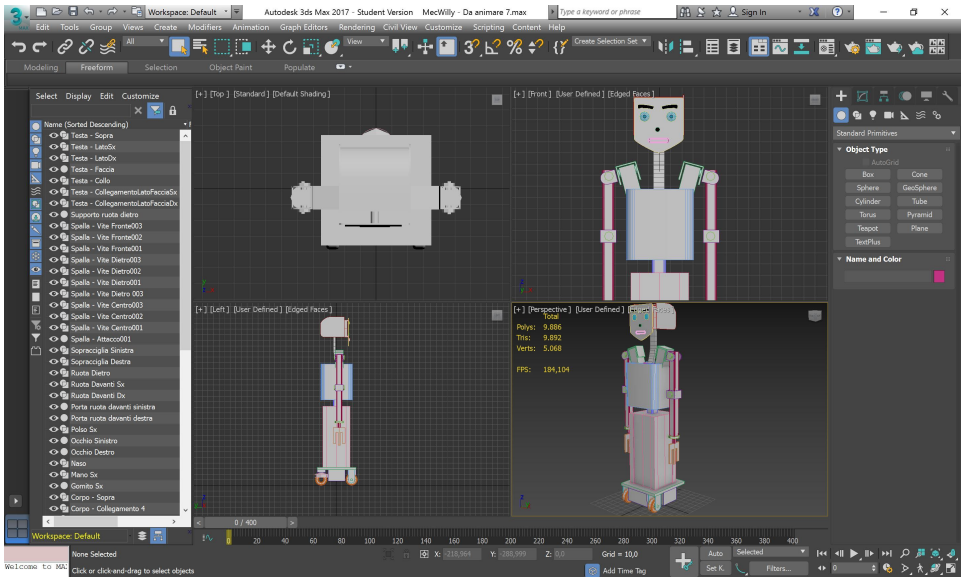

Figura 3.15: MecWilly secondo modello

3.1.6 MecWilly secondo modello

Una versione più definita e curata del robot è stata realizzata in un secondo momento per rendere più realistico e riconoscibile il modello. 3.15

3.1.7 Bidoni

I bidoni sono un elemento fondamentale del gioco. 3.16
Anche in questo caso si è partiti dal contorno della figura a cui è stato successivamente applicato un modificatore Shell. Il modificatore Shell aggiunge spessore agli oggetti.
Le ruote sono realizzate partendo da un profilo esterno circolare a cui è stato applicato il modificatore di estrusione. Il risultato è stato convertito in un poligono modellabile grazie al modificatore Edit Poly, questo ha permesso di smussare il profilo della ruota.
Essi sono un elemento di gioco animato, possono essere aperti o chiusi.3.18 3.17

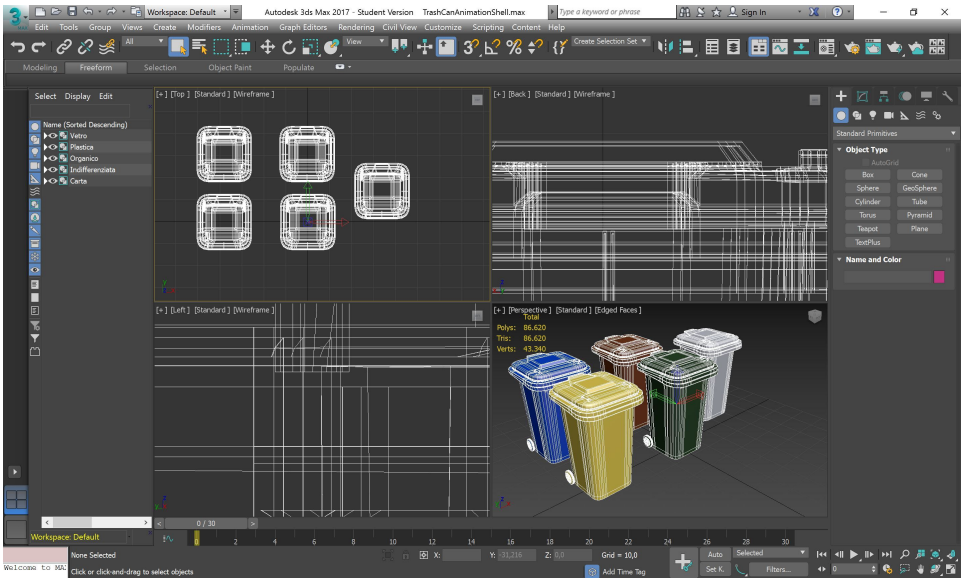

Figura 3.16: Bidoni

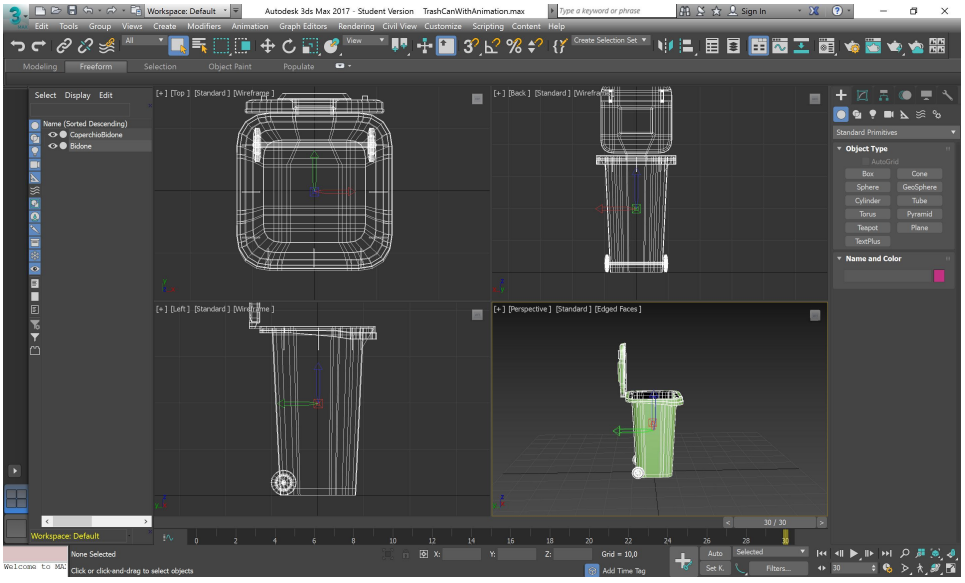

Figura 3.17: Bidone chiuso

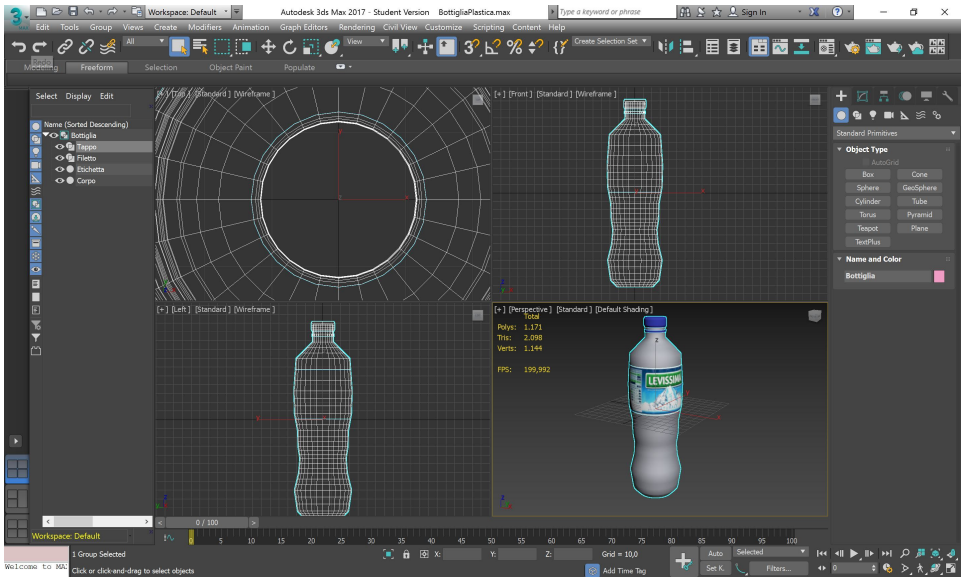

Figura 3.18: Bidone aperto

3.1.8 Carta stropicciata

Per relizzare della carta stropicciata è stato necessario l'utilizzo del modificatore Cloth.
Il modificatore Cloth si applica a tutti gli oggetti nella scena che hanno bisogno di essere parte della simulazione Cloth.
La simulazione Cloth viene utilizzata per accartocciare i fogli di carta. Questo è ottenuto inserendo all'interno di una sfera la carta che vogliamo accartocciare. La sfera rappresenterà l'oggetto solido di collisione nella nostra simulazione mentre la carta sarà l'oggetto che subisce gli effetti.
Poichè Cloth è un modificatore esso deve essere assegnato ad ogni elemento da inserirò nella simulazione. 3.19
Una volta impostato il modificatore sarà possibile far partire la simulazione. La sfera inizialmente avrà un raggio di dimensione superiore al foglio di carta, successivamente, con il passare del tempo, il suo raggio decrescerà fino a raggiungere una dimensione tale da produrre l'effetto desiderato di distorsione della carta. 3.20

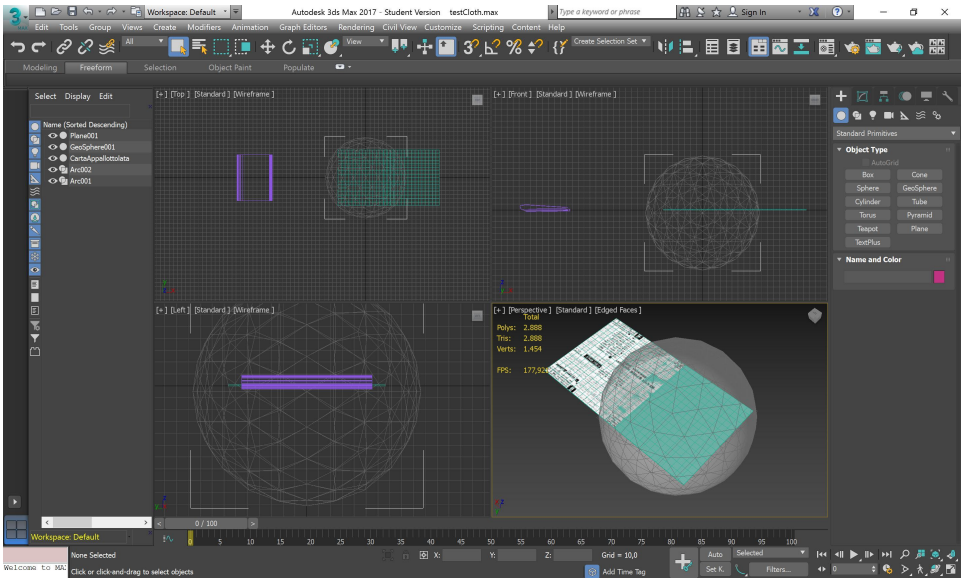

Figura 3.19: Carta stropicciata - Prima

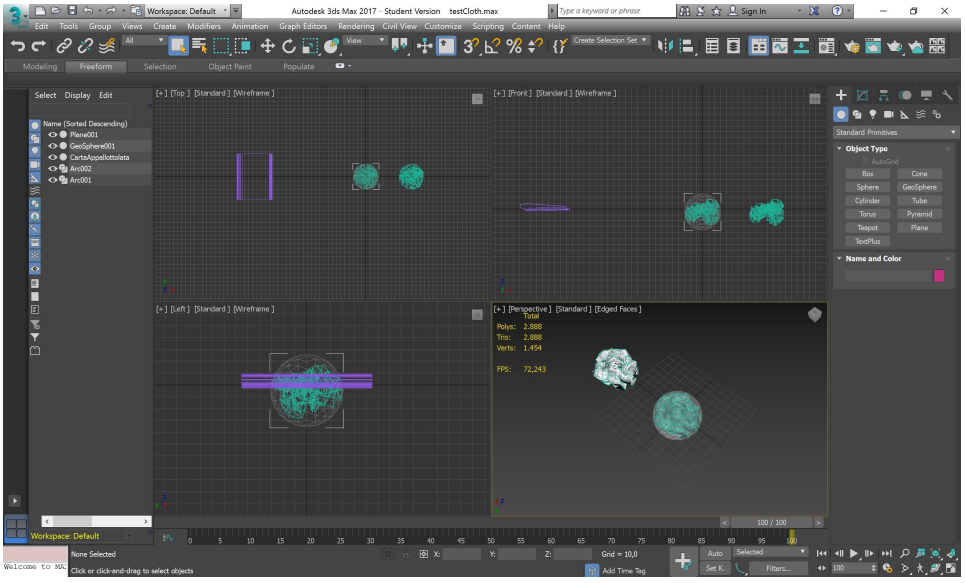

Figura 3.20: Carta stropicciata - Dopo

3.1.9 Esportazione dei modelli 3d

Per poter utilizzare i modelli nell'ambiente Unity è necessario esportarli in un formato compatibile. Per farlo è stata utilizzata l'utility Game Exported di 3dsmax, il quale permette di convertire i modelli in un formato FBX, compatibile tra gli altri anche con Unity. 3.21

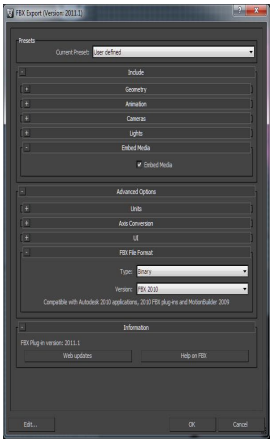

Figura 3.21: FBX Export

3.2 Unity

3.2.1 Prefabs

I modelli 3d sono stati importati in Unity e sono stati realizzati i prefabs degli oggetti istanziabili a runtime. 3.22

3.2.2 Logica di gioco

L'obettivo del gioco è condurre MecWilly al bidone corrispondente al rifiuto da smaltire.
L'area di gioco di MecWilly è una griglia. MecWilly può muoversi all'interno di questa griglia la quale è sede anche degli altri elementi di gioco, i bidoni. Lo stato del gioco viene mantenuto all'interno di un oggetto di classe GameManager che è un Singleton. Questo componente si occuperà di inizializzare

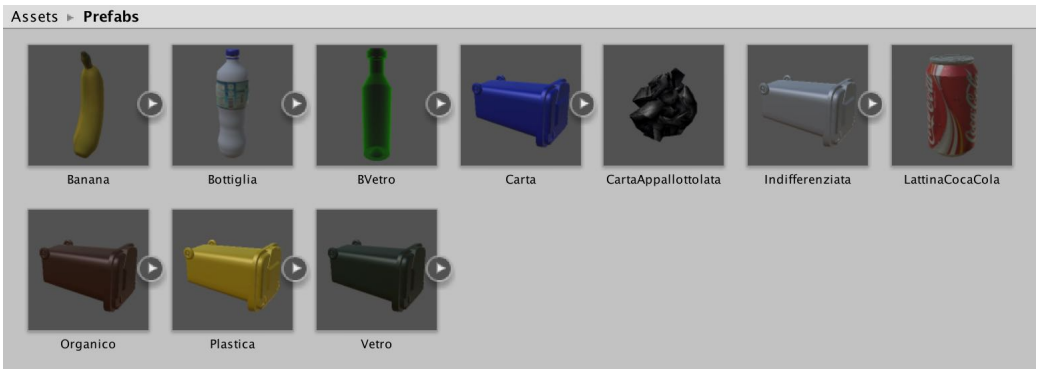

Figura 3.22: Prefabs

il livello, creare gli oggetti necessari, posizionarli nella posizione specificata nel file di configurazione del livello, di interpretare i comandi impartiti a MecWilly tramite i pulsanti su schermo, di contare il numero di passi totali impiegati per raggiungere la meta, del tempo totale impiegato per completare il livello.

Frammento del GameManager - Verifica dell'ordine impartito

```
IEnumerator checkCurrentPositionAction(
    PlayerMovementDirection pmd)
     {
          int oldRow = currentPlayerGridPosition.row;
          int oldColumn = currentPlayerGridPosition.
              column;
          int newRow = oldRow;
          int newColumn = oldColumn;
          bool ultimaMossa = false;
          PlayerOrientation oldOrientation =
              currentPlayerOrientation;
          PlayerOrientation newOrientation =
              currentPlayerOrientation;
          switch (pmd)
```

```
{
    case PlayerMovementDirection.Up:
        switch (currentPlayerOrientation)
        {
            case PlayerOrientation.Up:
                newColumn += numPassi;
                break;
            case PlayerOrientation.Down:
                newColumn -= numPassi;
                break;
            case PlayerOrientation.Left:
                newRow -= numPassi;
                break;
            case PlayerOrientation.Right:
                newRow += numPassi;
                break;
            default:
                break;
        }
        break;
    case PlayerMovementDirection.Down:
        switch (currentPlayerOrientation)
        {
            case PlayerOrientation.Up:
                newColumn -= numPassi;
                break;
            case PlayerOrientation.Down:
                newColumn += numPassi;
                break;
            case PlayerOrientation.Left:
                newRow += numPassi;
```

```
                    break;
                case PlayerOrientation.Right:
                    newRow -= numPassi;
                    break;
                default:
                    break;
            }
            break;
        case PlayerMovementDirection.Right:
            newOrientation = getNewOrientation(
                oldOrientation, pmd);
            break;
        case PlayerMovementDirection.Left:
            newOrientation = getNewOrientation(
                oldOrientation, pmd);
            break;
        default:
            break;
    }
    GarbageCanObject garbage =
        getGarbageCanForPosition(newRow, newColumn,
        newOrientation);
    switch (checkActionForPosition(newRow,
        newColumn, newOrientation, oldRow, oldColumn
        ))
    {
        case Action.None:
            break;
        case Action.Move:
            currentPlayerGridPosition.column =
                newColumn;
```

```
            currentPlayerGridPosition.row = newRow;
            yield return currentGridSystem.
                MovePlayer(currentPlayerGridPosition
                , oldRow, oldColumn, numPassi);
            break;
        case Action.Rotate:
            currentPlayerOrientation =
                newOrientation;
            yield return currentGridSystem.
                RotatePlayer(pmd);
            break;
        case Action.GarbageCan_Valid_Move:
            ultimaMossa = true;
            endTime = Time.time;
            yield return currentGridSystem.
                RotateGarbageCanToPlayer(garbage);
            yield return currentGridSystem.
                PlayApriGarbageCan(garbage);
            currentPlayerGridPosition.column =
                newColumn;
            currentPlayerGridPosition.row = newRow;
            yield return currentGridSystem.
                MovePlayer(currentPlayerGridPosition
                , oldRow, oldColumn, numPassi);
            yield return currentGridSystem.
                AvvicinaPlayer(garbage);
            yield return currentGridSystem.
                LasciaRifiuto();
            break;
        case Action.GarbageCan_Valid_Rotation:
            ultimaMossa = true;
```

```
                endTime = Time.time;
                yield return currentGridSystem.
                    RotateGarbageCanToPlayer(garbage,
                    true, pmd);
                yield return currentGridSystem.
                    PlayApriGarbageCan(garbage);
                currentPlayerOrientation =
                    newOrientation;
                yield return currentGridSystem.
                    RotatePlayer(pmd);
                yield return currentGridSystem.
                    AvvicinaPlayer(garbage);
                yield return currentGridSystem.
                    LasciaRifiuto();
                break;
            case Action.GarbageCan_Invalid:
                break;
            default:
                break;
        }
        numMosse += 1;
        if (ultimaMossa)
        {
            var guiTime = endTime - startTime;
            var minutes = guiTime / 60;
            var seconds = guiTime % 60;
            var fraction = (guiTime * 100) % 100;
            currentGridSystem.EndGame("Numero di mosse:
                "+numMosse+ "\nTempo impiegato: "+
                string.Format("{0:00}:{1:00}:{2:000}",
                minutes, seconds, fraction));
```

```
        }
    }
```

3.2.3 Griglia

La griglia di gioco è a scacchiera, i due tipi di piano sono rappresentati da due differenti materiali, uno grigio e uno verde. 3.2.3

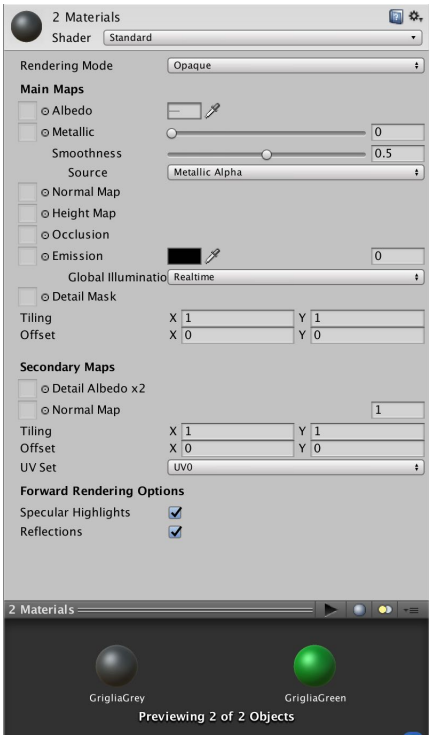

Figura 3.23: Materiali griglia

Frammento inizializzazione griglia

```
GameLevel level = GameManager.Instance.getCurrentLevel
    ();
        grid = new Vector3[level.rows, level.columns];
        Vector3 floorSize = floor.GetComponent<Renderer
            >().bounds.size;
        floorSize.x = floorSize.x - 2 * offset;
```

```
floorSize.z = floorSize.z - 2 * offset;
Vector3 topLeftPos = topLeftPivot.transform.
   position;
topLeftPos.x = topLeftPos.x + offset;
topLeftPos.z = topLeftPos.z + offset;
float rowStep = floorSize.x / level.rows;
float columnStep = floorSize.z / level.columns;
for (int i = 0; i < level.rows; i++)
{
    for (int j = 0; j < level.columns; j++)
    {
        grid[i, j] = new Vector3(topLeftPos.x +
            rowStep / (float)2.0 + rowStep * (i
           ), topLeftPos.y, topLeftPos.z +
           columnStep / (float)2.0 + columnStep
            * (j));
    }
}
for (int i = 0; i < level.rows; i++)
{
    for (int j = 0; j < level.columns; j++)
    {
        GameObject quad = GameObject.
           CreatePrimitive(PrimitiveType.Quad);
        quad.transform.localScale = new Vector3
           (rowStep, columnStep, 1);
        quad.transform.rotation = Quaternion.
           Euler(90, 0, 0);
        quad.transform.position = new Vector3(
           grid[i, j].x, grid[i, j].y + .001f,
           grid[i, j].z);
```

```
            if (j % 2 == 0)
            {
                if (i % 2 == 0)
                {
                    quad.GetComponent<Renderer>().
                        material = griglia1;
                }
                else
                {
                    quad.GetComponent<Renderer>().
                        material = griglia2;

                }
            }
            else
            {
                if (i % 2 == 0)
                {
                    quad.GetComponent<Renderer>().
                        material = griglia2;
                }
                else
                {
                    quad.GetComponent<Renderer>().
                        material = griglia1;
                }
            }
        }
    }
```

3.2.4 Rifiuto

MecWilly all'inizio del livello ha già in mano un oggetto. Questo oggetto è vincolato alla sua mano destra.
Per realizzare questo uno script (ItemFollowPlayer) è stato associato al GameObject che rappresenta il rifiuto.
Questo script permette di mantenere vincolato l'oggetto nella mano destra del robot. Quando ci troveremo in prossimità di un bidone dei rifiuti corretto allora sarà necessario eliminare questo vincolo, permettendo quindi all'oggetto di essere sottoposto alla forza di gravità e cadere liberamente.

Script ItemFollowPlayer

```
using UnityEngine;
using System.Collections;

public class ItemFollowPlayer : MonoBehaviour {

    public GameObject player;
    private bool vincolato = true;
        // Use this for initialization
        void Start () {
        Transform manoDx = player.transform.FindChild("
            AvambraccioDestro").FindChild("Mano Dx");
        transform.SetParent(manoDx);
    }

        // Update is called once per frame
        void Update () {
        if (vincolato)
        {
            transform.localPosition = new Vector3(0, 0,
                GetTotalMeshFilterBounds(transform).
```

```
            extents.z);
        transform.rotation = Quaternion.Euler(0,
            90, 0);
    }
}

public void Lascia()
{
    vincolato = false;
    transform.parent = null;
    this.GetComponent<Rigidbody>().velocity =
        Vector3.zero;
}

void Awake()
{

}

private static Bounds GetTotalMeshFilterBounds(
    Transform objectTransform)
{
    var meshFilter = objectTransform.GetComponent<
        MeshFilter>();
    var result = meshFilter != null ? meshFilter.
        mesh.bounds : new Bounds();

    foreach (Transform transform in objectTransform
        )
    {
```

```
            var bounds = GetTotalMeshFilterBounds(
                transform);
            result.Encapsulate(bounds.min);
            result.Encapsulate(bounds.max);
        }
        var scaledMin = result.min;
        scaledMin.Scale(objectTransform.localScale);
        result.min = scaledMin;
        var scaledMax = result.max;
        scaledMax.Scale(objectTransform.localScale);
        result.max = scaledMax;
        return result;
    }

}
```

3.2.5 Interfaccia utente

L'interfaccia utente è stata studiata in modo da non fornire indicazioni implicite sul significato dei pulsanti, ad esempio i pulsanti sinistra e destra sono stati disposti verticalmente. In questo modo viene stimolato il pensiero e si è portati a riflettere prima di far compiere un'azione al robot.

3.2.6 Livelli

I livelli sono memorizzati all'interno della directory Resources. Questa directory permette di memorizzare delle risorse accessibili a runtime accessibile su tutte le piattaforme.
Un livello è un file JSON con la seguente struttura:

```
{
  "name": "Livello 1",
  "rows": 6,
```

```
"columns": 6,
"startPositionX": 3,
"startPositionY": 2,
"garbage_type": 0,
"garbageCan": [
  {
    "type": 1,
    "positionX": 3,
    "positionY": 1
  },
      {
    "type": 2,
    "positionX": 1,
    "positionY": 1
  },
      {
    "type": 2,
    "positionX": 5,
    "positionY": 2
  },
      {
    "type": 3,
    "positionX": 2,
    "positionY": 3
  },
      {
    "type": 4,
    "positionX": 0,
    "positionY": 3
  },
      {
```

```
        "type": 2,
        "positionX": 4,
        "positionY": 4
      },
          {
        "type": 0,
        "positionX": 2,
        "positionY": 5
      }
    ]
}
```

name Il nome del livello

rows, columns La dimensione della griglia

startPositionX, startPositionY La posizione iniziale di MecWilly

garbage_type Il tipo di rifiuto iniziale

garbageCan Un array rappresentate i bidoni, ogni bidone è caratterizzato dal tipo e dalla posizione sulla griglia.

3.2.7 Esempio di partita

All'avvio viene visualizzato il menù di selezione del livello di gioco. 3.24

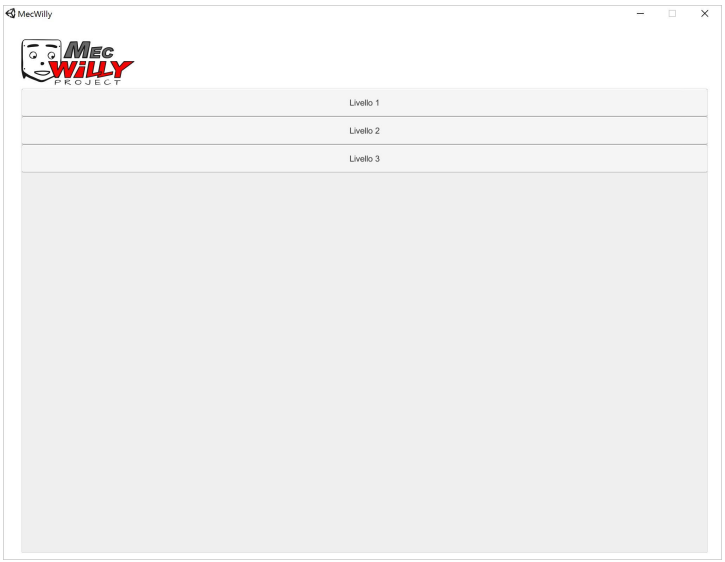

Figura 3.24: Selezione del livello.

Una volta selezionato il livello esso viene caricato e il gioco ha inizio. 3.25

Figura 3.25: Inizio del livello

Premendo i pulsanti possiamo impartire ordini a MecWilly fino al raggiungimento del bidone corretto 3.26.

Figura 3.26: MecWilly si avvicina al bidone aperto

MecWilly si avvicinerà al bidone e lascerà cadere l'oggetto che entrerà nel bidone 3.27.

Figura 3.27: Lattina che cade

Una volta terminata l'animazione viene visualizzata la schermata di completamento di un livello nella quale sono indicati il numero di mosse e il

tempo impiegato 3.28.

Figura 3.28: Completamento di un livello

Capitolo 4

Esperimenti

Una classe di bambini di terza elementare (8 anni) è stata coinvolta per effettuare dei test sull'applicazione.
Ai bambini sono stati fatti provare 3 livelli di test con grado di difficoltà crescente, dopo di che è stata fatta una intervista in cui veniva chiesto ai bambini di valutare o esprimere delle opinioni:

- In una scala da 1 (Per nulla) a 10 (Moltissimo) quanto ti è piaciuto il gioco?
- In una scala da 1 (Per nulla) a 10 (Moltissimo) quanto è stato facile usare il gioco?
- In una scala da 1 (Per nulla) a 10 (Moltissimo) quanto sono chiare le informazioni del gioco?
- C'è qualcosa che non ti è piaciuto del gioco? Se si, che cosa non ti è piaciuto?
- C'è qualcosa in particolare che ti è piaciuto del gioco? Se si, che cosa ti è piaciuto di più?
- C'è qualcosa che non hai capito del gioco? Se si, cosa non hai capito?
- Se dovessi fare delle modifiche al gioco, che cosa cambieresti?

La valutazione data dai bambini in merito al gioco in 3D è risultata molto positiva, soprattutto per quanto riguarda "Quanto è piaciuto il gioco" e "Quante sono chiare le informazioni".
Il punto da evidenziare però è l'autostima che il gioco stimola nei bambini, evidenziata dai vari commenti a "Cosa ti è piaciuto in particolare":
Muovere il robot
Mettere i rifiuti nel bidone giusto
Il robot che andava dove volevo
Guidare il robot per buttare il rifiuto
Riuscire nel gioco, in particolare in giochi di "problem solving", permette ai bambini di acquisire fiducia in sé stessi (e nelle proprie capacità), facendo sì che si sentano più sicuri in situazioni simili future. Quest'aspetto è evidente anche nelle risposte alla domanda "Se dovessi fare delle modifiche, che cosa cambieresti?", nelle quali molti bambini sottolineano che vorrebbero più livelli e maggiori difficoltà, probabilmente determinata dall'essere riusciti nel gioco attuale. A ciò contribuisce, ovviamente, il fatto che ai bambini viene lasciato il tempo di terminare comunque il gioco cosicché possano sperimentare il "successo" anziché trovarsi dinanzi ad un insuccesso che, se non adeguatamente corredato da una gestione consapevole dei motivi che portano all'insuccesso, potrebbe portare a future insicurezze.
Nel gioco, l'insuccesso è presente, sempre, ogni volta che il bambino sbaglia il movimento, ma ha la possibilità di comprendere l'errore (aver confuso la destra con la sinistra, in quanto non ha ruotato mentalmente l'immagine rispetto al robot) e correggerlo nel movimento successivo.

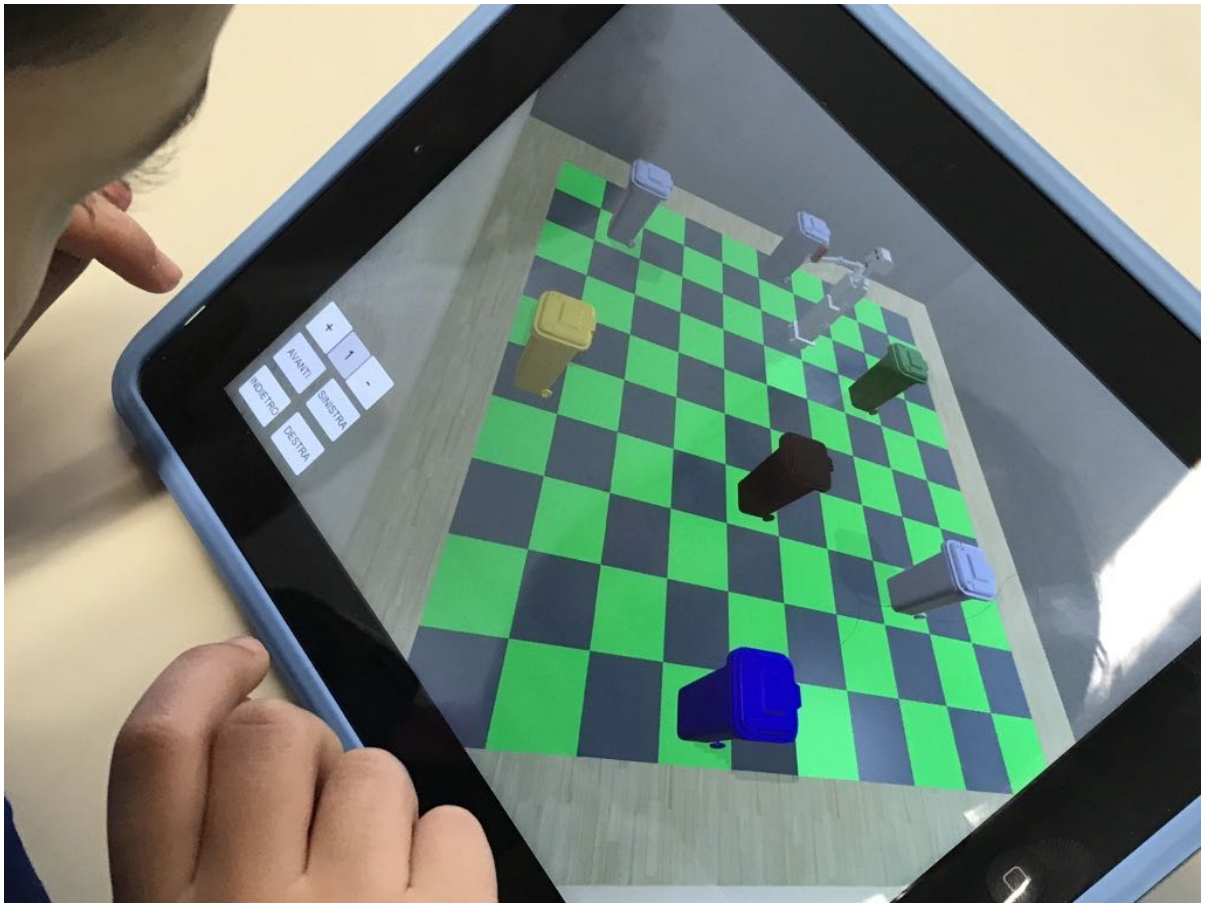

Figura 4.1: Utilizzo dell'applicazione

4.1 Livelli di gioco

Figura 4.2: Livello 1

Il Livello 1 consiste nel condurre MecWilly, che ha in mano una lattina, al bidone giallo.
La griglia di gioco è 6x6.

Figura 4.3: Livello 2

Il Livello 2 consiste nel condurre MecWilly, che ha in mano una lattina, al bidone giallo.
La griglia di gioco è 10x10.

Figura 4.4: Livello 3

Il Livello 3 consiste nel condurre MecWilly, che ha in mano della carta, al bidone blu.
La griglia di gioco è 6x6.

4.2 Risultati del test

Il campione di test è formato da 11 femmine e 7 maschi. Ogni bambino ha giocato tutti e 3 i livelli. Il tempo medio di gioco è stato di circa 2 minuti. In questo test non era importante valutare il numero di mosse impiegate per completare il livello ma bensì l'esperienza di gioco.
Al termine della sessione di gioco è stata fatta loro un'intervista con i seguenti risultati:

Tabella 4.1: Valutazione MecWilly 3d

	Bello	**Facile**	**Chiaro**	**Maschio/femmina**
1	10	1	1	F
2	10	2	3	F
3	8	7	9	F
4	8	9	7	F
5	10	8	10	F
6	9	10	10	F
7	8	7	10	F
8	10	10	10	F
9	10	9	10	F
10	10	9	10	F
11	10	9	10	F
12	3	10	10	M
13	10	10	10	M
14	10	8	5	M
15	9	9	10	M
16	9	6	8	M
17	10	10	10	M
18	9	10	9	M

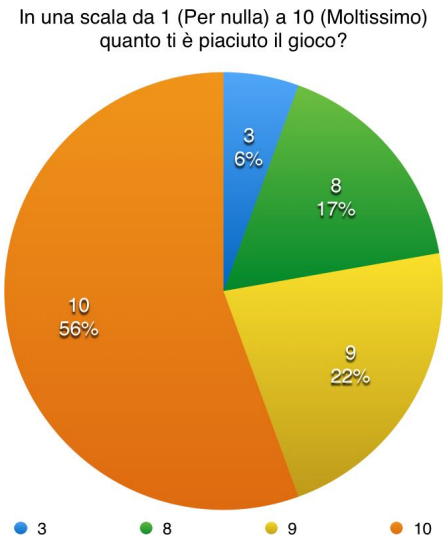

Figura 4.5: Grafico gradimento gioco

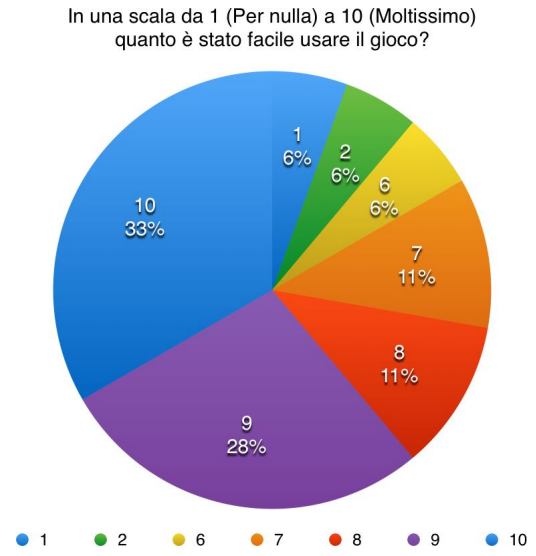

Figura 4.6: Grafico semplicità del gioco

Figura 4.7: Grafico chiarezza delle informazioni

4.3 Considerazioni sull'esperimento

L'applicazione MecWilly 3D è stata installata su 2 tablet, uno Android e un iPad. Ai bambini sono state fornite informazioni base sull'utilizzo del gioco, è stato spiegato l'obiettivo del gioco e indicata l'area dei comandi 4.8.

Figura 4.8: UI - Comandi

La quasi totalità dei bambini ha immediatamente capito come fare per muovere il robot avanti e indietro. Le difficoltà sono iniziate quando è stato necessario ruotare il robot in quanto, siccome nel gioco il robot parte in

posizione frontale rispetto al giocatore, la sua destra è a sinistra dal punto di vista del gioco e viceversa. La quasi totalità dei bambini ha commesso un errore durante la prima rotazione in quanto non ha ruotato mentalmente l'immagine rispetto al robot.
Il gioco permetteva di specificare il numero di passi da far compiere a MecWilly 4.8. Durante il test è emerso che in tutti i casi (salvo qualche eccezione ma realtiva solamente ad alcune fasi della partita e non dall'inizio della stessa) nessun bambino ha sfruttato questa possibilità, limitandosi a dare comandi con passo 1.
A tutti i bambini è stata data la possibilità di giocare tutti e 3 i livelli, senza limiti di tempo.

Conclusioni

Il videogioco presentato in questa tesi ha l'obiettivo di sviluppare e migliorare le abilità nell'orientamento topografico (mappe) e nell'orientamento spaziale (contesto) dei bambini della scuola primaria.
La realizzazione di un videogioco tridimensionale ha permesso di avere uno strumento per sperimentazioni più reali, simulando la stessa situazione della sperimentazione con il robot fisico, cosa che non era possibile fare con la versione bidimensionale.
Sono stati ipotizzati diversi sviluppi futuri:

- Aggiunta del sonoro, per migliorare i feedback.
- Interazione vocale.
- Posizionamento casuale dei comandi del robot, per non dare punti di riferimento.
- Modalità di gioco cooperativo.
- Possibilità di generazione automatica di livelli mediante un pannello di amministrazione in cui è possibile specificare la dimensione dell'area di gioco, la modalità di gioco, la posizione degli elementi.
- Più test ed esperimenti che coinvolgano scuole primarie.

Bibliografia

[1] Cos'è la gamification. http://www.gamification.it/gamification/introduzione-alla-gamification/.

[2] Sam S. Adkins. The 2012-2017 worldwide game-based learning and simulation-based markets. http://www.ambientinsight.com/Resources/Documents/AmbientInsight_SeriousPlay2013_WW_GameBasedLearning_Market.pdf, 2013.

[3] Paola Salomoni Elvis Mazzoni Catia Prandi, Silvia Mirri. Mecwilly in your pocket: on evaluating a mobile serious game for kids. http://ieeexplore.ieee.org/document/7543737/, 2013.

[4] Paola Salomoni Valentina Nisi Nuno Jardim Nunes Catia Prandi, Marco Roccetti. Fighting exclusion: a multimedia mobile app with zombies and maps as a medium for civic engagement and design, 2015.

[5] Silvia Mirri Catia Prandi, Paola Salomoni. Gamification in crowdsourcing applications, 2015.

[6] Rose Eveleth. How many photographs of you are out there in the world? https://www.theatlantic.com/technology/archive/2015/11/how-many-photographs-of-you-are-out-there-in-the-world/413389/, 2015.

[7] Christopher Cunningham Gabe Zichermann. Gamification by design: Implementing game mechanics in web and mobile apps, 2011.

[8] C. Infante. Imparare giocando. l'interattività tra teatro e ipermedia, 2000.

[9] Karl M. Kapp. The gamification of learning and instruction: Game-based methods and strategies for training and education, 2011.

[10] Scott Steinberg Kyle Orland, Dave Thomas. The videogame style guide and reference manual, 2007.

[11] Vygotskij L.S. Mind in society: The development of higher psychological processes, 1978.

[12] Laura Dabbish Luis von Ahn. Designing games with a purpose, 2008.

[13] Laura Dabbish Luis von Ahn. Designing games with a purpose, 2008.

[14] Oikonomou Andreas Jain Lakhmi C. Ma, Minhua. Serious games and edutainment applications. http://www.springer.com/us/book/9781447121602, 2011.

[15] Marina Maselli. Conflitto sociocognitivo. http://www.comune.forli.fc.it/servizi/menu/dinamica.aspx?idArea=23949&idCat=54464&ID=58399.

[16] Benvenuti M. Mazzoni, E. A robot-partner for preschool children learning english using socio-cognitive conflict. educational technology & society, 2015.

[17] Sapio B. Mazzoni E., Nicolò E. Children, multimedia content and technological artefacts: An exploratory study using text analysis tools. interactive technology and smart education, 2015.

[18] MecWilly Project. Mecwilly project. http://www.mecwilly.it/, 2010-2016.

[19] Ben Sawyer. Serious games taxonomy, 2008.

[20] Deterding Sebastian. From game design elements to gamefulness: defining gamification., 2011.

[21] Rilla Khaled Lennart Nacke Sebastian Deterding, Dan Dixon. From game design elements to gamefulness: defining gamification, 2011.

[22] Cordell Steiner — TEDxUniversityofStThomas. Individualization, failure and fun. `http:https://www.youtube.com/watch?v=P-djW4uj7rI`, 2014.

[23] Luis von Ahn. Games with a purpose, 2006.

[24] Luis von Ahn. Games with a purpose, 2006.