



UNIVERSITÀ DI PISA

Corso di Laurea in Informatica Umanistica

RELAZIONE

**Un sistema di menù con stile conforme al  
contesto storico per la replica virtuale  
dell'addizionatore della prima CEP**

**Candidato:** *Federico Talarico*

**Relatore:** *Giovanni Antonio Cignoni*

**Correlatore:** *Mirko Aurelio Tavosanis*

Anno Accademico 2020-2021

# Indice generale

Introduzione.....	4
1. L'ispirazione.....	6
1.1 L'addizionatore della prima CEP.....	6
1.2 Progetto HMR.....	9
1.3 Videogiochi didattici.....	11
1.3.1 Finalità didattiche e non solo.....	12
1.3.2 Videogiochi e storia, una classificazione.....	14
1.3.3 Esempi dal mondo dei videogiochi.....	15
1.3.4 Limiti dell'utilizzo dei videogiochi nella didattica.....	21
1.3.5 L'importanza dei menù.....	26
1.4 Obiettivi della tesi.....	31
1.4.1 Realizzazione del sistema di menù.....	31
1.4.2 Strumenti.....	32
2. L'interazione dell'utente col gioco.....	34
2.1 Il concetto di sessione.....	34
2.2 Fuori dalla sessione.....	35
2.2.1 Esempi dal mondo dei videogiochi.....	36
2.2.2 Requisiti per la replica virtuale dell'addizionatore a 6 bit.....	39
2.3 Dentro la sessione, menù in sovrapposizione.....	40
2.3.1 Esempi dal mondo dei videogiochi.....	40
2.3.2 Requisiti per la replica virtuale dell'addizionatore a 6 bit.....	44
2.4 Dentro la sessione, interazione con la scena 3D.....	47
2.4.1 Esempi dal mondo dei videogiochi.....	47
2.4.2 Requisiti per la replica virtuale dell'addizionatore a 6 bit.....	50
2.4.2.1 Interruttore generale dell'addizionatore.....	51
2.4.2.2 Interruttore del ventilatore.....	51
2.4.2.3 Chiavi telefoniche per gli addendi dell'addizionatore.....	52
2.4.2.4 Ventilatore Marelli.....	52
2.4.2.5 Indicatori luminosi.....	52
2.4.2.6 Telaietti.....	52
2.4.2.7 Attori e interazione in Unreal.....	53
2.5 Lo stile dei menù: criteri e soluzioni.....	56
2.5.1 Interazione nel sistema di menù: classi di interazione.....	57
2.5.2 Riferimenti storici al mondo telescriventi.....	60
2.5.3 Riferimenti storici al memex.....	63
2.5.4 Stile telegrafico.....	64
2.5.5 Stile meccanico.....	66
2.6 Progettazione del sistema dei menù.....	67
2.6.1 L'architettura dell'applicazione.....	67
2.6.2 I diagrammi UML.....	68
3. Come lo feci.....	76
3.1 Lo sviluppo in Unreal, l'esperienza del prototipo.....	76
3.1.1 Blueprint e User Interface Designer.....	77

3.1.2 L'integrazione del simulatore: tra C++ e BP.....	78
3.1.3 Modelli 3D, texture, materiali, suoni ed effetti.....	79
3.1.3.1 Modellazione 3D.....	79
3.1.3.2 Texture e materiali.....	80
3.1.3.3 Suoni.....	80
3.1.3.4 Effetti.....	80
3.1.4 Altri problemi affrontati e risolti.....	81
3.2 Il lavoro sui menù: widget e funzionalità.....	81
3.2.1 Architettura object oriented.....	81
3.2.2 Variazioni di bitmap.....	84
3.2.3 Widget modulari.....	84
3.2.4 Widget telegrafici: realizzazione in Unreal.....	85
3.2.5 Widget telegrafici: realizzazione in Gimp.....	87
3.2.6 Widget meccanici: realizzazione in Unreal.....	91
3.2.7 Widget meccanici: realizzazione in Gimp.....	93
3.2.8 Traduzioni e localizzazione.....	96
3.2.9 Gestione delle impostazioni (GameUserSettings + MoneyBin).....	97
3.3 L'archivio e i suoi contenuti.....	97
Conclusioni.....	101
Strumenti di sviluppo e dimensione del lavoro.....	102
Sviluppi futuri.....	102
Appendici.....	104
Glossario.....	104
Codici.....	106
Bibliografia.....	114
Bibliografia videoludica.....	122
Sitografia e altri media.....	125

# Introduzione

La prima Calcolatrice Elettronica Pisana (CEP) del 1957 fu l'apripista dell'informatica italiana. HMR (ProgettoHMR, sito web), un progetto indipendente di ricerca in storia dell'informatica, ha recuperato e studiato la documentazione originale, scoprendo fra l'altro l'esistenza di un addizionatore a 6 bit costruito nel 1956 come prova di fattibilità. Nel 2011 HMR ha realizzato il simulatore della prima CEP e la replica fisica dell'addizionatore. Oggi, per rendere l'addizionatore più fruibile, è in via di sviluppo una sua replica virtuale con grafica 3D e interazione in prima persona realizzata usando Unreal Engine 4 come motore di gioco.

La tesi si inserisce in questo progetto proseguendo il lavoro iniziato nel tirocinio. Nel tirocinio era stata sviluppata l'architettura di gioco integrando il simulatore e realizzando in forma prototipale tutti i componenti principali della scena 3D. La tesi approfondisce il tema dell'interazione con l'utente curando il sistema di menù dell'applicazione e portandolo alla sua forma finale. I menù gestiscono le opzioni di gioco e di simulazione, il montaggio e smontaggio dei componenti dell'addizionatore, la navigazione fra i testi e le foto della documentazione storica.

In particolare, i menù sono stati un'opportunità sfruttata per creare uno stile conforme al contesto che arricchisce l'applicazione con riferimenti storici alle tecnologie dell'epoca e narrazioni di contorno della ricerca scientifica che ha portato alla realizzazione dei primi calcolatori elettronici.

La tesi è organizzata in tre capitoli. Il primo è dedicato a cosa è ispirata la tesi. Al suo interno vengono introdotti il contesto storico delle CEP e il progetto di ricerca in cui è stata sviluppata l'applicazione, la sua classificazione come videogioco didattico, gli obiettivi della tesi e gli strumenti utilizzati per raggiungerli.

Nel secondo capitolo sono discussi i criteri adottati per gestire l'interazione dell'utente col gioco e i criteri seguiti nella ricerca di uno stile grafico ricco di riferimenti storici e nella progettazione del sistema di menù.

Nel terzo capitolo vengono ripercorse e descritte le varie fasi dello sviluppo in Unreal Engine e dell'integrazione del simulatore a partire dal periodo di tirocinio. È presente

anche un sottocapitolo dedicato alla documentazione dei menù, con approfondimenti mirati sui principali aspetti della realizzazione e del metodo modulare adottato sia in Unreal che in Gimp.

# 1. L'ispirazione

*Il contenuto viene prima del design. Il design, senza contenuti, non è design, è decorazione.*

Jeffrey Zeldman, 5 maggio 2008.

Con questo simulatore per la prima volta HMR si avventura nella realizzazione di un'applicazione in grafica tridimensionale. Questo capitolo tratta delle premesse, la storia delle CEP e gli obiettivi di HMR, e del contesto del risultato: i videogiochi didattici applicati alla narrazione della storia. Segue una discussione sull'utilizzo dei videogiochi nella didattica e sull'importanza del coinvolgimento del giocatore. Il capitolo si conclude con la dichiarazione degli obiettivi di tesi e la descrizione degli strumenti utilizzati.

## 1.1 L'addizionale della prima CEP

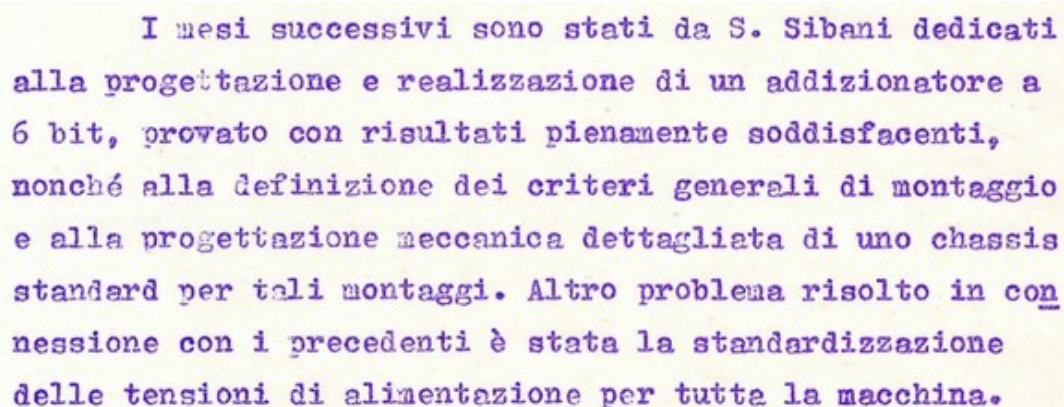
Nel 1954 l'Università di Pisa, che aveva inizialmente ottenuto 150 milioni di lire di finanziamenti per realizzare un sincrotrone, di fronte alla decisione di realizzarlo a Frascati, decise di reinvestire i fondi destinati all'acceleratore di particelle nella costruzione di una macchina calcolatrice. Nel marzo del 1955 viene quindi istituito il Centro di Studi sulle Calcolatrici Elettroniche (CSCE).

Il progetto, sviluppato in collaborazione con l'Olivetti, cominciò a dare i primi frutti nel 1957 con la realizzazione della prima Calcolatrice Elettronica Pisana (CEP) anche nota come Macchina Ridotta (MR). La macchina fu utilizzata per circa un anno per poi essere smantellata per riusare i materiali elettronici nella costruzione della seconda CEP, completata nel 1961 e usata per circa sette anni.

Completata il 24 luglio 1957 (Conversi 1957), la MR fu il primo calcolatore digitale interamente progettato e costruito in Italia. Fu utilizzata come strumento di calcolo a servizio di ricerche esterne al progetto CEP durante il 1958.

Grazie alle due CEP, l'Università iniziò quel percorso di costruzione di competenze che negli anni successivi rese Pisa il luogo di altri importanti risultati quali il Centro Nazionale Universitario di Calcolo Elettronico (CNUCE) nel 1965, il corso di laurea in Scienze dell'Informazione nel 1969/70, il primo nodo internet nel 1986. Per una trattazione approfondita delle vicende delle CEP si rimanda a (Cignoni e altri, 2013; Cignoni e Gadducci, 2020).

Recuperando e studiando la documentazione originale del progetto CEP HMR ha scoperto che, fra i primi componenti completati nel 1956 quando iniziò la costruzione della prima CEP, c'era un addizionatore a 6 bit. I disegni dell'addizionatore offrono un'opportunità per confrontare la MR con la tecnologia dell'epoca. L'addizionatore fu realizzato da Sergio Sibani (Relazione CSCE 1956, v. fig. 1), ricercatore proveniente dall'Istituto di Fisica di Roma e assunto dall'Olivetti nel contesto della collaborazione con l'Università di Pisa.



I mesi successivi sono stati da S. Sibani dedicati alla progettazione e realizzazione di un addizionatore a 6 bit, provato con risultati pienamente soddisfacenti, nonché alla definizione dei criteri generali di montaggio e alla progettazione meccanica dettagliata di uno chassis standard per tali montaggi. Altro problema risolto in connessione con i precedenti è stata la standardizzazione delle tensioni di alimentazione per tutta la macchina.

Figura 1. Relazione in cui viene nominato l'addizionatore (dettaglio).

Fu un importante banco di prova per verificare alcune delle soluzioni logiche ed elettroniche poi utilizzate dalle due CEP: furono messi a punto componenti come lo chassis dei telaietti, pensato per dare alla MR (e successivamente alla seconda CEP) una modularità costruttiva che avrebbe agevolato gli interventi di manutenzione. Inoltre i disegni hanno permesso di riconoscere le fonti d'ispirazione dei ricercatori del CSCE. La rete logica di uno stadio dell'addizionatore per esempio, riprende una soluzione di derivazione IBM pubblicata in un articolo del 1953 che descriveva l'unità aritmetica dell'IBM 701 (Ross 1953), una parentela mai dichiarata dalle note del CSCE, ma che impreziosisce la storia del primo calcolatore italiano.



Figura 2. Il quadro di controllo della prima CEP. (Foto della prima CEP, 1957. Archivio Generale di Ateneo dell'Università di Pisa).



## 1.2 Progetto HMR

Hackerando la Macchina Ridotta (ProgettoHMR, sito web) è un progetto indipendente di ricerca in storia dell'informatica italiana in particolare, ma non esclusivamente. Attivo dal 2006, ha obiettivi di diffusione della cultura scientifica che raggiunge sia attraverso metodi storici tradizionali (Cignoni e Gadducci, 2013), sia tramite l'utilizzo dell'informatica di oggi per simulare o ricostruire quella del passato (per esempio Cignoni e altri 2015), usando tecniche di simulazione particolari (Cignoni e Paci, 2012).

HMR sta per “Hackerando la Macchina Ridotta”, nome del progetto che si rifà alla vera definizione di hacker (Malkin 1996; The Jargon File, hacker) come una persona che ama esplorare i dettagli e i limiti dei sistemi programmabili in maniera quasi ossessiva. HMR ha sposato questo atteggiamento inizialmente perché, per ricostruire le macchine del passato, è necessario comprenderne la tecnologia fino in fondo esplorandone con appassionato accanimento tutti i dettagli. La prima CEP, detta Macchina Ridotta, è stato il primo progetto di ricostruzione affrontato da HMR, per questo è rimasto nel nome del progetto.

Nel 2009 HMR ha realizzato il simulatore della MR secondo la documentazione del 1956 (Cignoni e altri, 2009), nel 2011 quello della MR effettivamente realizzata nel 1957 dopo le modifiche al primo progetto – l'approccio hacker ha permesso di scoprire l'esistenza di due versioni della MR, informazione che si era persa prima dell'indagine di HMR. Nel 2020 HMR ha realizzato il simulatore del *Calcolatore Automatico Numerico Educativo* (CANE) del 1970, uno strumento didattico frutto di due tesi e utilizzato dagli studenti dell'appena attivato corso di laurea in Scienze dell'Informazione (Cignoni e altri 2020).

L'attenzione per i dettagli è caratteristica di tante iniziative del progetto: dall'indagine documentale sul ruolo di Fermi nel progetto CEP (Cignoni, 2014; Cignoni e Gadducci, 2020), all'identificazione della stanza delle CEP nell'attuale pianta di Palazzo Matteucci (Cignoni e altri 2019), oppure alla ricostruzione della storia dei PC d'oltre cortina disassemblando le ROM degli interpreti BASIC (Bodrato e altri, 2019).

In questa tesi “l’ossessione hacker” si manifesta nella cura per i dettagli che caratterizzano le *bitmap* (v. glossario) dei menù, il numero delle loro variazioni utilizzate nel gioco e l’oculata scelta dei riferimenti storici significativi dai quali prendere ispirazione sia per la veste grafica che per le modalità di gestione dell’interazione.

HMR ha anche obiettivi di diffusione per raccontare una storia dell’informatica corretta a un pubblico più ampio possibile (Cignoni e Colosimo, 2014; ProgettoHMR, archivio di PaginaQ; Cignoni e Pratelli, 2018; ProgettoHMR, OggiSTI). Dal 2015 HMR fornisce anche i contenuti al corso di Storia dell’Informatica della laurea in Informatica Umanistica all’Università di Pisa (ProgettoHMR, pagina del corso).

La simulazione di vecchi strumenti per il calcolo è uno dei metodi preferiti da HMR per raccontare la storia dell’informatica, ma il progetto si è cimentato anche nella ricostruzione di repliche fisiche.

Nel 2011 HMR, seguendo i progetti originali e utilizzando in parte componenti dell’epoca, ha costruito una replica fisica dell’addizionatore del 1956 di Sibani. La replica è stata presentata in manifestazioni internazionali (Cignoni e Gadducci, 2013; Youtube, VCFe - Vintage Computer Festival Europe, Munich 2012) ed è oggi conservata al Museo degli Strumenti per il Calcolo di Pisa dove è utilizzata come strumento per la didattica dell’aritmetica binaria.

I simulatori realizzati finora da HMR permettono di interagire con le macchine ricostruite attraverso il quadro di controllo manuale realizzato in grafica 2D.

*MR-VR* è il sottoprogetto di HMR per dare ai suoi simulatori interfacce in realtà virtuale: l’impegno obiettivo finale è creare un videogioco didattico che permetta di usare la prima CEP attraverso visore e controller VR (Cignoni 2021).

La tesi si inserisce nel primo esperimento di questo percorso: la realizzazione della replica virtuale dell’addizionatore a 6 bit, motivato anche dalla volontà di superare la scarsa fruibilità e i limiti fisici della replica del 2011.

La replica virtuale può essere distribuita in formato digitale, il che la rende accessibile a chiunque e in qualsiasi momento. Anche senza le emergenze che abbiamo vissuto nella recente pandemia Covid-19, questo tipo di distribuzione ha indubbi vantaggi per sperimentare e fare esperienze senza dover per forza visitare una città lontana.

Inoltre la replica fisica non può essere utilizzata liberamente: è possibile assistere a dimostrazioni, ma essendo un oggetto delicato e in esemplare unico, non è possibile sperimentare osando comportamenti rischiosi. La replica virtuale dell'addizionale può essere tranquillamente maltrattata, dando la possibilità all'utente di sperimentarne i limiti, di superarli, di danneggiare l'addizionale, di studiare le conseguenze dei danni e provare a sostituirne i pezzi.

Il tirocinio che ha preceduto la tesi ha realizzato l'architettura del gioco integrando il simulatore e realizzando in forma prototipale tutti i componenti principali della scena 3D. La tesi approfondisce e realizza nella sua forma definitiva una delle componenti principali di ogni gioco: il sistema di menù.

La progettazione del sistema di menù, come parte delle finalità di HMR, ha tenuto di conto la necessità di raccontare la storia come ulteriore obiettivo didattico del videogioco.

### **1.3 Videogiochi didattici**

Andare a scuola e studiare è un impegno che per alcuni studenti può essere faticoso e anche noioso. Un hobby è invece un'attività che si pratica per interesse personale. Nei due casi è diversa la capacità di mantenere attenzione e concentrazione, in particolare di fronte a ostacoli e difficoltà. Per questo motivo la didattica è interessata a utilizzare strumenti capaci di suscitare la curiosità e mantenere alto l'interesse degli studenti. I videogiochi possono aiutare a rendere le lezioni più coinvolgenti (Di Serio e altri, 2013), offrendo la possibilità di sperimentare in prima persona e di affinare abilità specifiche (McMichael 2007).

Nell'ottica più globale della *gamification*, che consiste nell'utilizzare giochi o elementi mutuati da essi in ambiti non ludici (Deterding e altri, 2011) al fine di incuriosire e coinvolgere un pubblico più ampio, sono molti i contesti culturali in cui i videogiochi vengono utilizzati. Nello spazio limitato di questa tesi li citeremo soltanto, in favore di una trattazione più approfondita del rapporto tra videogiochi e storia, dei limiti dell'utilizzo nel contesto scolastico e dell'importanza del sistema di menù nei videogiochi pensati per la didattica e la diffusione culturale.

### 1.3.1 Finalità didattiche e non solo

L'uso didattico dei videogiochi non è legato soltanto alle scuole, ma si estende anche agli eventi culturali, ai festival della scienza, ai musei e presenta diverse declinazioni:

1. videogiochi usati a fini didattici, ma provenienti dal mercato generalista;
2. videogiochi didattici, sviluppati appositamente per coprire argomenti che sono parte di programmi scolastici o di obiettivi di diffusione culturale;
3. videogiochi usati come esercizi nell'ambito della didattica della programmazione;
4. videogiochi usati in ambito medico o come supporto per studenti con disabilità;

Uno dei campi in cui si trovano numerosi esempi è quello dei videogiochi per la didattica della programmazione, da circa sessant'anni terreno di molti studi, in particolare di ricercatori del MIT. Sin dai tempi del *Fortran* infatti (Backus 1956), vengono sviluppati strumenti per supportare l'insegnamento della programmazione. Celebri raccolte di esempi in Basic (Ahl 1978), linguaggi come il *Logo* di Wallace Feurzeig (Feurzeig e altri, 1969) in cui il codice disegnava forme in grafica vettoriale, o *Scratch* (Resnick e altri, 2009) in cui oltre a disegnare è possibile creare animazioni e semplici videogiochi, fino ad arrivare a *Rabbids Coding!* (Ubisoft, 2019), gioco didattico sviluppato dalla Ubisoft in cui il giocatore deve risolvere dei rompicapo utilizzando correttamente i blocchi di codice a sua disposizione, sono testimonianza del fatto che questa tipologia di giochi sia riuscita a suscitare anche l'interesse di case di sviluppo appartenenti al mercato generalista.

Rimanendo nel contesto della didattica, l'ambito medico è un altro dei campi in cui i videogiochi sono sempre più presenti sia in sede diagnostica che a supporto della terapia (Lieberman 2012). Attraverso l'uso dei videogiochi infatti si possono raggiungere obiettivi di integrazione sociale, esercizio e supporto aggiuntivo di studenti con problemi quali disabilità, disturbi dello spettro autistico (Bondioli 2020), dislessia (Di Tore e altri, 2014), disgrafia e discalculia e disturbi dell'attenzione.

I videogiochi possono inoltre essere utilizzati a scopo commemorativo per ricordare o celebrare eventi storici (*memorial games*), per sensibilizzare (*moral games*), o addirittura per fare proselitismo, propaganda e anche parodia e satira politica.

Quest'ultimo aspetto aiuta a definire anche il videogioco come fenomeno culturale. *America's Army: Proving Grounds* (U.S. Army, 2015) è un gioco commissionato dal governo degli Stati Uniti per pubblicizzare l'esercito e favorire il reclutamento. Non mancano giochi antiamericani sviluppati da simpatizzanti di Al-Qaeda o dell'ISIS, come *Quest for Bush* (Global Islamic Media Front, 2005), gioco che invita a compiere atti terroristici. Nel 2011 anche la nostra Presidenza del Consiglio commissionò *Gioventù ribelle* (Gruppo di Filiera dei Produttori Italiani di Videogiochi, 2011), un gioco commemorativo per il 150° dell'unità d'Italia — un fiasco colossale. Ha avuto miglior fortuna critica, forse per la sua vena satirica e goliardica, *Call of Salveenee* (Marco Alfieri, 2015), videogioco indipendente il cui unico obiettivo è ironizzare su alcuni episodi recenti della politica italiana rappresentandoli in chiave demenziale.

Un altro contesto che si presta all'utilizzo dei videogiochi come strumento didattico è quello degli studi storici, un ambito vicino alla ricerca di HMR, per quel che riguarda storia dell'informatica. I simulatori di HMR del CANE e della MR57: sono principalmente orientati a mostrare le macchine in funzione, in pratica offrono un'esperienza più tecnica che coinvolgente. Strumenti più immersivi come la realtà virtuale (VR) e la realtà aumentata (AR) garantiscono livelli di coinvolgimento più alti rispetto a quelli che solitamente si raggiungono con la meno sofisticata combinazione di tavolo, sedia e schermo (Schutte e Stilinović, 2017). La realtà virtuale infatti trova il suo utilizzo sia nella ricerca (Frerich e altri, 2014) e nella formazione medica (Nagendran e altri, 2013), che in ambito museale sia con fini turistici di promuovere la visita fisica (The National Museum of Computing, 3D Virtual Tour) sia puramente culturali (Sinclair, Finn. *The VR Museum of Fine Art*. Sinclair, 2016). Per una trattazione più approfondita su VR e AR si rimanda a (Anthes e altri, 2016; Azuma 1997; Van Krevelen e Poelman, 2010; Billinghurst e altri, 2015). In questo senso, l'adozione della prospettiva personale in un ambiente 3D offre alla replica virtuale dell'addizionale la possibilità di migliorare il coinvolgimento,

affiancando alla simulazione tecnica nuove opportunità di rappresentazione e ricostruzione storica.

### **1.3.2 Videogiochi e storia, una classificazione**

L'impiego dei videogiochi nell'insegnamento della storia non va visto come insolito, ma comparabile a quelle attività di supplemento alle lezioni e utili a favorire le discussioni in classe, come per esempio la visione di film (McMichael 2007, p. 204). Secondo McMichael, i videogiochi sono un valido strumento per guidare gli studenti verso una visione olistica della storia: vivere la storia attraverso un intreccio narrativo aiuta lo studente a notare i collegamenti tra i vari eventi e fatti storici mostrati. Inoltre, a differenza degli altri media, i videogiochi non si limitano a soddisfare delle domande, ma offrono la possibilità di trovarsi le risposte sperimentando in prima persona (Schut 2007). Lo studente infatti, diventando uno degli attori della storia e operando scelte che si ripercuotono sulla sua partita, è stimolato a riflettere su temi storicamente rilevanti relativi a cause e conseguenze, intraprendendo un percorso di analisi e speculazione sui possibili "what if" della storia che trascende i limiti imposti dalla realtà storica (Uricchio 2005, pp. 328-329).

Si possono distinguere 3 classi di videogiochi a tema storico (Schrier 2018, pp. 74-75), a seconda che siano incentrati su:

1. *la rappresentazione del passato*; giochi che offrono al giocatore la possibilità di "rivivere" fra i protagonisti un particolare momento o evento storico attraverso una ricostruzione videoludica dichiaratamente citata;
2. *l'interazione con temi e scelte storiche*; permettono un'interazione più astratta con la storia, mettendo il giocatore nel ruolo di "un dio" (Uricchio 2005, p. 328) che deve amministrare risorse e decidere strategie per far crescere il suo popolo ed espandere il proprio dominio;
3. *la storia come ambientazione*; la storia in questo caso rappresenta la fonte d'ispirazione per una realtà alternativa verosimile, talvolta anche accuratamente ricostruita, ma che fa solo da sfondo a una trama di fantasia;

Come per tutte le classificazioni ci sono giochi che stanno a metà tra una classe e l'altra o che presentano elementi provenienti da più classi. Nel seguito esamineremo alcuni titoli sia commerciali che di nicchia e il loro rapporto con la storia.

### **1.3.3 Esempi dal mondo dei videogiochi**

La storia può essere raccontata oppure utilizzata come fonte d'ispirazione. Questo paragrafo fornisce alcuni esempi per ognuna delle classi descritte in precedenza.

*Mission US* (WNET, 2009), preso in esame in (Schrier 2018, p. 78), è un browser game gratuito organizzato in capitoli ambientati in determinati momenti della storia degli Stati Uniti: dal massacro di Boston agli eventi della rivoluzione americana alle guerre indiane. Il giocatore rivive queste vicende attraverso dialoghi e situazioni ispirate a documenti reali. È un tipico esempio della prima classe: è esplicitamente didattico e mira a insegnare il pensiero storico e coinvolgere lo studente con l'immedesimazione e il trasporto empatico.

*Reliving the Revolution* (Schrier, 2005) è un gioco in realtà aumentata da fare *in-site* nei luoghi della battaglia di Lexington (Schrier 2018, p. 83). Il gioco invita a svolgere un'indagine storica percorrendo il luogo fisico, esplorandolo alla ricerca delle testimonianze scritte dei soldati che appaiono sul dispositivo del giocatore a seconda della sua posizione. Anche questo è un gioco esplicitamente didattico appartenente alla prima classe anche se tecnologicamente molto diverso.

La serie di *Civilization* (presa in esame da Uricchio 2005 e McMichael 2007), come rappresentante della seconda classe, è ambientata nel periodo di tempo compreso tra il 2500 a.C. e il presente, caratteristica che rende la ricostruzione storica grossolana e poco accurata, ma che consente di giocare con l'evoluzione della società umana nel suo complesso. Il gioco, uno strategico a turni, nelle sue varie versioni tende a seguire un percorso in linea di massima storicamente valido: il giocatore dovrà prima investire risorse nello sviluppo della scrittura e solo dopo potrà costruire biblioteche, praticare la filosofia e la letteratura, dovrà aver fatto progredire la fisica prima di teorizzare la forza di gravità, dovrà sviluppare in ordine le tecnologie necessarie a colonizzare lo spazio. Contestualmente, il giocatore è chiamato ad amministrare il

proprio impero, condurre spedizioni militari, gestire le risorse, affrontare catastrofi e problematiche sociali.

Un altro esempio della seconda classe è la serie di *Europa Universalis* (Paradox Interactive) (McMichael 2007). Il periodo storico principale dell'ambientazione è quello tra la scoperta dell'America (1492) e la rivoluzione francese (1789), con estensioni dal 1399 al 1821. A differenza di *Civilization* appartiene alla categoria dei giochi strategici ma, sebbene anche in questo caso l'obiettivo del gioco sia conquistare il mondo, *Europa Universalis* ha un livello di complessità più elevato nell'amministrazione dell'impero: gestione dell'inflazione, alleanze internazionali, tratte commerciali, campagne militari, integrazione culturale sono solo alcune delle problematiche che il giocatore deve fronteggiare. Anche la religione può essere utilizzata come strumento di controllo o di conquista promuovendo campagne di evangelizzazione per favorire l'annessione delle colonie.

La serie di *Total War* (Creative Assembly) rappresenta l'anello di congiunzione tra *Civilization* e *Europa Universalis*, alternando fasi di gioco a turni e in tempo reale. La serie è generalmente ambientata nel passato con contesti diversi, dall'epoca degli *Shogun: Total War* (Creative Assembly, 2000) a *Napoleon: Total War* (Creative Assembly, 2010), ma non mancano capitoli ambientati in mondi fantastici come quello di *Warhammer* (Creative Assembly. *Total War: Warhammer*. 2016). Per esempio, *Total War: Rome II* (Creative Assembly, 2013) è ambientato tra l'epoca repubblicana (509 - 27 a.C.) e l'età imperiale (27 a.C.- 395 d.C.). Nelle fasi a turni il giocatore deve amministrare il potere in funzione delle campagne militari: rapporti diplomatici, economia, miglioramento dell'esercito. Nelle fasi in tempo reale invece è chiamato a condurre il proprio esercito in battaglie terrestri e navali prendendo decisioni tattiche in base allo schieramento nemico, la conformazione del terreno e le condizioni climatiche. Il gioco permette di ricostruire battaglie su larga scala curando la quantità di soldati a schermo e la scelta di unità diverse ispirate alla realtà storica.

I cosiddetti *memorial games* si collocano fra la prima e la seconda classe: attraverso una fedele rappresentazione del passato, fanno riflettere il giocatore su eventi storici rilevanti (Rughiniş e Matei, 2015, p. 629). Un esempio di memorial game è *Valiant Hearts: The Great War* (Ubisoft, 2014), ambientato sul fronte occidentale della prima



guerra mondiale. Per coinvolgere il giocatore, fa leva sulla trama e sul trasporto empatico sfruttando il punto di vista dei quattro protagonisti appartenenti a entrambi gli schieramenti. Gli oggetti presenti nel mondo di gioco, spesso foto e documenti d'archivio, immergono il giocatore nella rappresentazione del passato, la trama e la voce narrante lo portano a riflettere sul conflitto e gli orrori della guerra.

Con il termine *moral games* invece, si fa riferimento ai giochi che hanno come obiettivo primario quello della sensibilizzazione. Un buon esponente è *This War of Mine* (11 bit studios, 2014) ambientato durante l'assedio di Sarajevo nella guerra fra Bosnia ed Erzegovina (1992-1996). Il gioco è centrato sulla gestione delle risorse come *Civilization*, ma sono diversi il coinvolgimento e il punto di vista: il giocatore gestisce una piccola comunità di civili vittime del conflitto che, oltre a sopravvivere, devono relazionarsi con altre vittime come loro aiutandole o meno. Più che decisioni riguardo allo svolgersi della storia, il gioco impone scelte molto forti dal punto di vista etico-morale, invitando alla riflessione (De Smale e altri, 2019).

La serie di *Assassin's Creed* ha un'ambientazione storica che varia a seconda del capitolo, dal Rinascimento alla rivoluzione industriale, tornando all'antico Egitto e all'ambientazione vichinga dei capitoli più recenti. Per i capitoli ambientati nel Rinascimento gli sviluppatori hanno prestato particolare attenzione a ricostruire le città di Firenze, Venezia, Roma com'erano al tempo (v. fig. 3, nel gioco la costruzione della cupola è anticipata, ma la vecchia facciata è riprodotta accuratamente).



Figura 3. La facciata della basilica antica di San Pietro prima del rinnovamento di Michelangelo così com'è stata ricostruita in una stampa del 1891 e in *Assassin's Creed: Brotherhood* (Ubisoft, 2010).

La trama incrocia personaggi ed eventi storici di quel periodo: Leonardo Da Vinci, Niccolò Machiavelli, Cesare e Rodrigo Borgia, la congiura dei Pazzi e la repubblica di Savonarola. Ogni capitolo è corredato di un database contenente le informazioni storiche, che offrono approfondimenti al giocatore più attento.

Anche l'ambientazione della serie di *Grand Theft Auto (GTA)* può essere considerata storica. Gli sviluppatori hanno modellato il mondo di gioco, dalla fine degli anni '80 a oggi, con particolare attenzione e cura per i dettagli. Nel corso dei capitoli della serie, per esempio, è possibile notare l'evoluzione della telefonia mobile prestando attenzione ai modelli di cellulare presenti in ogni gioco (v. fig. 4).



Figura 4. I telefoni dei protagonisti di *GTA*.

In *GTA Vice City* (Rocstar North, 2002), ambientato nel 1986, il giocatore ha a disposizione un telefono ispirato al *Motorola DynaTAC 8000X* (il primo telefono cellulare commerciale, 1984). In *GTA San Andreas* (Rockstar North, 2004), ambientato nel 1992, un *Motorola Microtac*. In *GTA IV* (Rockstar North, 2008), ambientato nel 2008, un telefono probabilmente ispirato al *Sony Ericsson W302*.

Nell'ultimo capitolo della serie, *GTA V* (Rockstar North, 2013), ambientato nel 2013, il giocatore ha la possibilità di giocare vestendo i panni di tre personaggi principali, caratterizzati anche a livello di dotazione telefonica con telefoni ispirati al *BlackBerry Curve 9380*, all'*iPhone 4s* e al *Nokia Lumia 800* (v. fig. 4). Il contesto storico è descritto anche a livello di abitudini sociali: in *GTA San Andreas* per scattare fotografie ai passanti bisogna essere muniti di macchina fotografica, da *GTA IV* invece è possibile utilizzare il telefono, in *GTA V* non è insolito incontrare passanti che stanno facendo videochiamate o che si scattano *selfie* con il loro smartphone. Le stesse osservazioni possono essere fatte per gli elettrodomestici: gli schermi a tubo catodico che diventano a led negli ultimi capitoli, le console per i videogiochi, i veicoli e la presenza sempre più marcata di internet. Questi particolari, uniti a una fedele ricostruzione delle città in cui si svolgono le vicende, offrono al giocatore la possibilità di rivivere l'atmosfera della New York dei primi anni 2000 (*GTA IV*) o della Los Angeles attuale (*GTA V*).

Altri esempi della terza classe sono *Red Dead Redemption 2 (RDR2)* (Rockstar Studios, 2019) e *Call of Juarez: Gunslinger (CoJ)* (Techland, 2013). Entrambi i giochi sono ambientati nel vecchio West, ma si avvicinano alla storia in modi diversi. *RDR2* si ispira a eventi storici e fatti di cronaca realmente accaduti riproponendoli al giocatore evitando però i nomi reali. *CoJ* invece usa questi avvenimenti come canovaccio per raccontare una storia di fantasia i cui protagonisti sono personaggi storici e fornendo al giocatore, come *Assassin's Creed*, un archivio ricco di approfondimenti su di essi e sulle loro vicende.

Altre serie riconducibili alla terza classe sono *Call of Duty (CoD)*, *Medal of Honor* e *Battlefield*, in cui l'ispirazione storica riguarda le caratteristiche e la denominazione delle armi e la trama della campagna in *single player*.

In conclusione, prendendo in considerazione la quantità di esempi di titoli che hanno a che fare con la storia, è possibile dire che:

1. i giochi della prima classe sono quasi esclusivamente videogiochi didattici, generalmente di nicchia, e nascono con il preciso scopo di affrontare e rappresentare fedelmente la storia; sono perlopiù sviluppati da piccoli team

indipendenti o da enti culturali e le grandi case di produzione sembrerebbero ignorare quasi del tutto questa classe;

2. la seconda classe comprende sia titoli molto commerciali che titoli di nicchia; sono presenti sia titoli in cui la storia è poco approfondita e solo di contesto, sia titoli in cui le tematiche storiche sono importanti quasi al livello della prima classe e costituiscono un'opportunità per veicolare il sapere storico e spunti di riflessione non banali;
3. nei giochi della terza classe, la storia generalmente è solo uno sfondo che serve a dare contesto e rendere più interessante il mondo di gioco; sembra essere la classe prediletta delle grandi produzioni, che hanno risorse umane e finanziarie per realizzare mondi di gioco estremamente vasti e particolareggiati;

La replica virtuale dell'addizionatore si colloca perfettamente nella prima classe essendo nata dall'intento di raccontare una storia e sviluppata da un piccolo team per fini didattici.

### **1.3.4 Limiti dell'utilizzo dei videogiochi nella didattica**

I videogiochi possono supportare la didattica tradizionale, ma non sono esenti da problemi. Come rilevato da McMichael (2007, pp. 214-215), è un errore dare per scontato che tutti gli studenti, in quanto giovani, abbiano lo stesso grado di interesse per i videogiochi. Inoltre il contesto scolastico potrebbe togliere al videogioco parte della sua componente ludica, facendolo percepire come un altro compito imposto al pari di una lettura o di una produzione scritta. In quest'ottica è comprensibile come un'esperienza macchinosa e una finitura grafica non al passo coi tempi possano spingere lo studente non solo al disinteresse più totale, ma anche a un vero e proprio rigetto del gioco e con esso degli argomenti che dovrebbe aiutare a imparare.

Uno dei problemi principali dei videogiochi didattici infatti è la cattiva finitura. A differenza dei videogiochi commerciali, ormai grandi produzioni con alle spalle considerevoli investimenti, i videogiochi didattici sono spesso realizzati da piccoli team indipendenti oppure da enti culturali senza grandi risorse. In molti casi questo si traduce in un risultato molto al di sotto dello standard dell'industria. Non bisogna

dimenticare che gli studenti sono anche consumatori: mettono naturalmente a confronto i videogiochi didattici con i videogiochi che sono abituati a giocare. Per quanto un piccolo team di sviluppo non possa aspirare a competere con le grandi aziende, allo stesso tempo non può sottrarsi al confronto. È vitale che gli sforzi di sviluppo siano finalizzati a obiettivi raggiungibili in modo da evitare che il confronto risulti impietoso.

Alcuni esempi di produzioni didattiche che non reggono il confronto col mercato generalista sono *Gioventù Ribelle*, *Difendiamo le Mura* (Pietroni e Rescic, 2015) e la ricostruzione 3D dei Lungarni di Pisa (Berretta 2012) (v. fig 5, 6 e 7).



Figura 5. In alto *Gioventù ribelle*, commissionato dalla Presidenza del Consiglio, in basso *Crysis 2* (Crytek, 2011), uscito nello stesso anno. Il gioco non risulta al passo coi tempi sotto molti aspetti, uno dei più evidenti è quello della resa grafica.

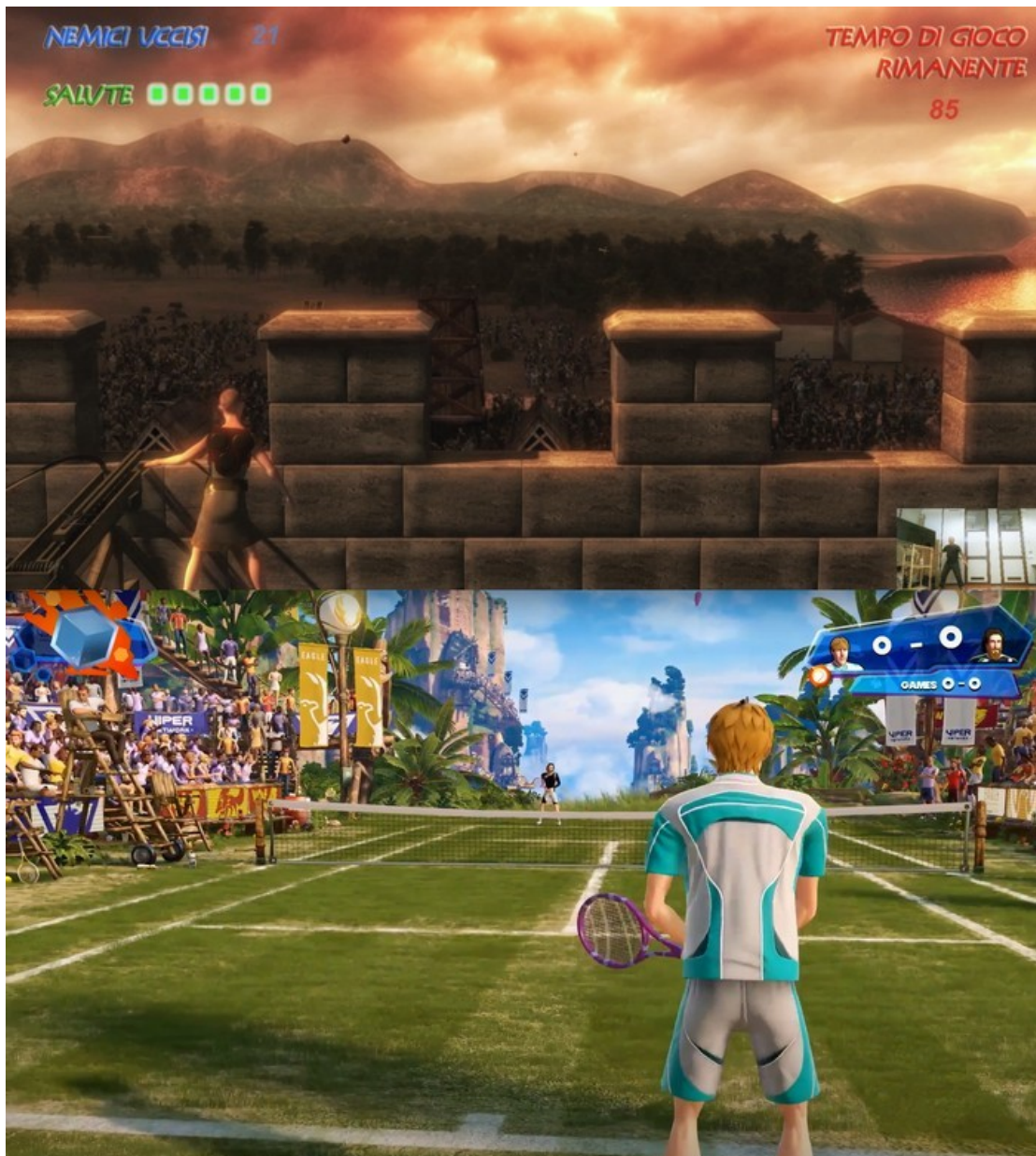


Figura 6. In alto alto *Difendiamo le mura*, sviluppato in partnership con CNR ISTC, Fondazione Paestum, Università di Salerno. In basso un gioco che utilizza una tecnologia simile: *Kinect Sports Rivals* (Rare, 2014), uscito un anno prima.





Figura 7. In alto la ricostruzione 3D dei Lungarni di Pisa, sponsorizzata dalla Banca di Pisa e Fornacette e Gli Amici dei Musei e Monumenti Pisani. In basso *Assassin's Creed II* (Ubisoft Montréal, 2009), uscito tre anni prima. Le differenze si notano in particolare nel livello di realismo della simulazione dell'acqua e nella ricchezza di dettaglio che caratterizza i palazzi che si affacciano sul canale veneziano.

Ai problemi di finitura si aggiungono quelli di inserimento nei programmi didattici. Esattamente come qualsiasi altra attività a supporto della didattica, anche l'utilizzo dei videogiochi ha bisogno di essere integrato e supportato da un percorso didattico in

classe nel quale l'insegnante deve essere il riferimento e la guida (Schrier 2018). I videogiochi, infatti, si prestano anche a una fruizione passiva generata dalla noia o da modalità di gioco eccessivamente avvincenti che rubano la scena ai contenuti didattici. È compito dell'insegnante guidare gli studenti incentivando uno stile di gioco analitico e speculativo che metta in risalto i particolari utili agli obiettivi didattici prefissati.

La discussione in classe, specialmente se il gioco è proveniente dal mercato generalista, diventa anche un'occasione per comprendere meglio l'industria del videogioco e sviluppare lo spirito critico. Nei giochi a tema storico, per esempio, la storia viene spessoedulcorata rimuovendo elementi come il razzismo, la schiavitù e le disuguaglianze tra uomo e donna (McMichael 2007). È importante che l'insegnante segnali queste inesattezze e si faccia promotore della discussione sui motivi che hanno portato gli sviluppatori a operare questi rimaneggiamenti.

### **1.3.5 L'importanza dei menù**

Per quanto si tenda spesso a darli per scontato, in realtà i menù sono una delle parti del videogioco con cui si interagisce maggiormente (Ng e altri, 2018). Attraverso i menù si gestiscono tutti i parametri di gioco, dal volume dei suoni alle impostazioni grafiche, alle opzioni specifiche del gioco come il livello di difficoltà; sempre dai menù si amministrano risorse determinanti nel *gameplay* (v. glossario) come l'inventario o l'evoluzione del personaggio, si consultano documentazione e contenuti aggiuntivi.

Se i menù sono ben realizzati quasi se ne trascura l'importanza; viceversa, ogni difetto salta subito all'occhio e si ripercuote sull'esperienza di gioco. In alcune categorie di giochi, di strategia, di ruolo o nelle avventure grafiche, l'utente passa dentro i menù una considerevole percentuale del suo tempo. È importante che tutto sia ben organizzato sia come funzionamento che come design (v. fig. 8).



Figura 8. Menù di avanzamento di livello di *Dark Souls* (FromSoftware, 2011). La suddivisione dell'interfaccia in tre colonne chiaramente distinte permette al giocatore di individuare facilmente i vantaggi offerti dal miglioramento della caratteristica selezionata. Il colore blu segnala l'aumento delle statistiche, il colore rosso l'insufficienza delle risorse necessarie per compiere l'avanzamento di livello.

Considerazioni che a maggior ragione valgono per i videogiochi didattici. Alcuni utenti, mossi dall'interesse per il soggetto trattato, possono accontentarsi della fedeltà di una ricostruzione o della precisione di una simulazione, sopportando un'iterazione non perfetta. Lo stesso non si può dire degli studenti, che non devono trovare motivi per essere delusi o infastiditi dal gioco. È in questo contesto che una veste grafica accattivante e un buon sistema di menù possono fare la differenza.

Inoltre, lo stile grafico dei menù è uno degli elementi che definiscono l'identità del gioco, che altrimenti potrebbe risultare anonimo. Molti giochi infatti, soprattutto tra i didattici, sottovalutano l'importanza di mantenere uno stile coerente con il contesto di gioco, presentando interfacce stranianti o generiche.

Ad esempio *Gioventù ribelle* (v. fig. 9).



Figura 9. Menù principale “di” *Gioventù ribelle*, nessuna personalizzazione rispetto al default del motore di gioco di cui conserva anche il titolo.

I menù di *Cuphead* sono invece un ottimo esempio. Il gioco, un *run 'n' gun* (v. glossario) a scorrimento orizzontale, è ambientato in un mondo ispirato ai cartoni animati americani degli anni '30 e '40. Quest'ispirazione è stata utilizzata anche per i menù, che vengono sfruttati per fare citazioni al mondo dell'animazione (v. fig. 10) o dei videogiochi (v. fig. 11).

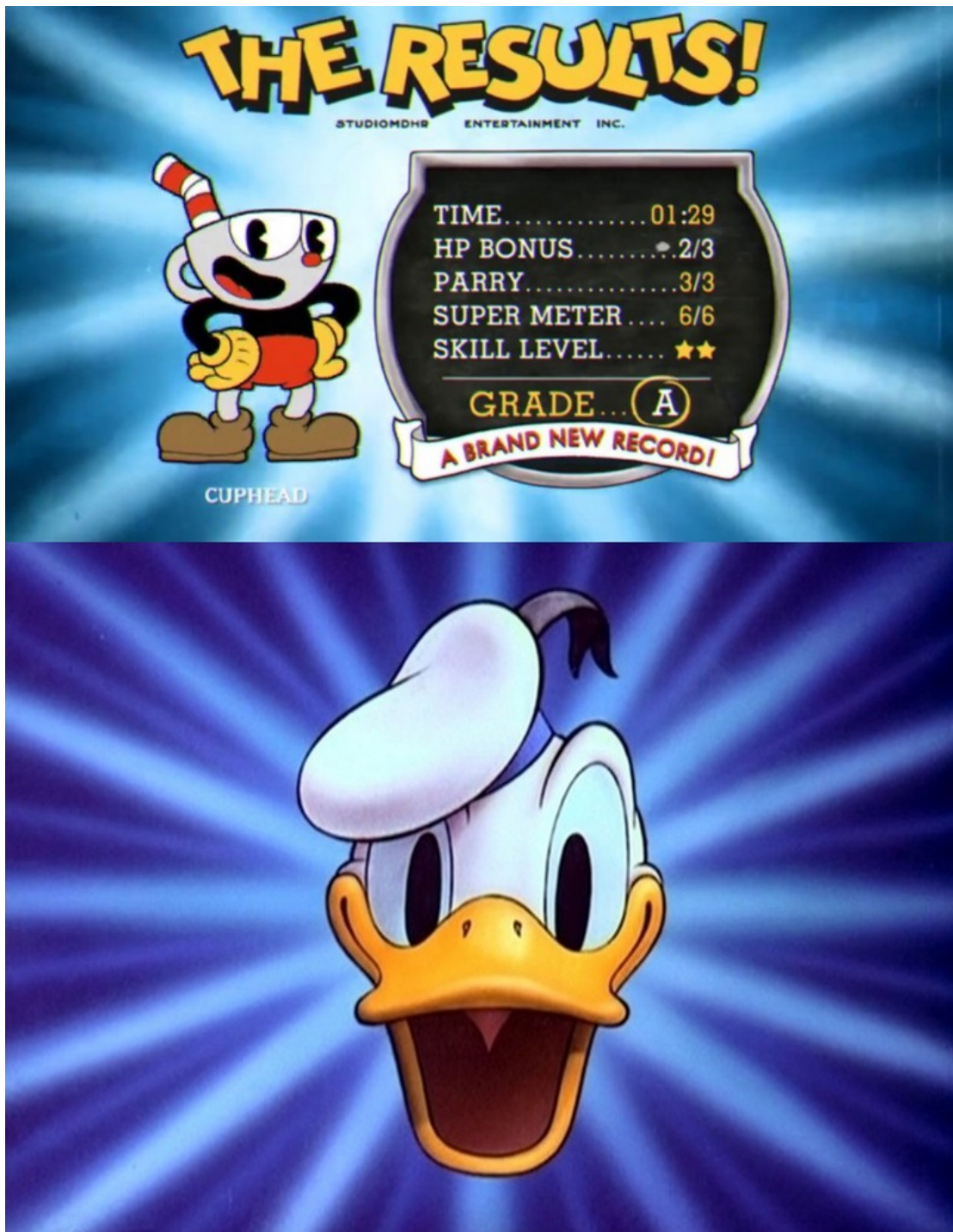


Figura 10. *Victory Screen* di *Cuphead*, ispirata alla sigla utilizzata per i cartoni animati di *Donald Duck* a partire dal 1937.

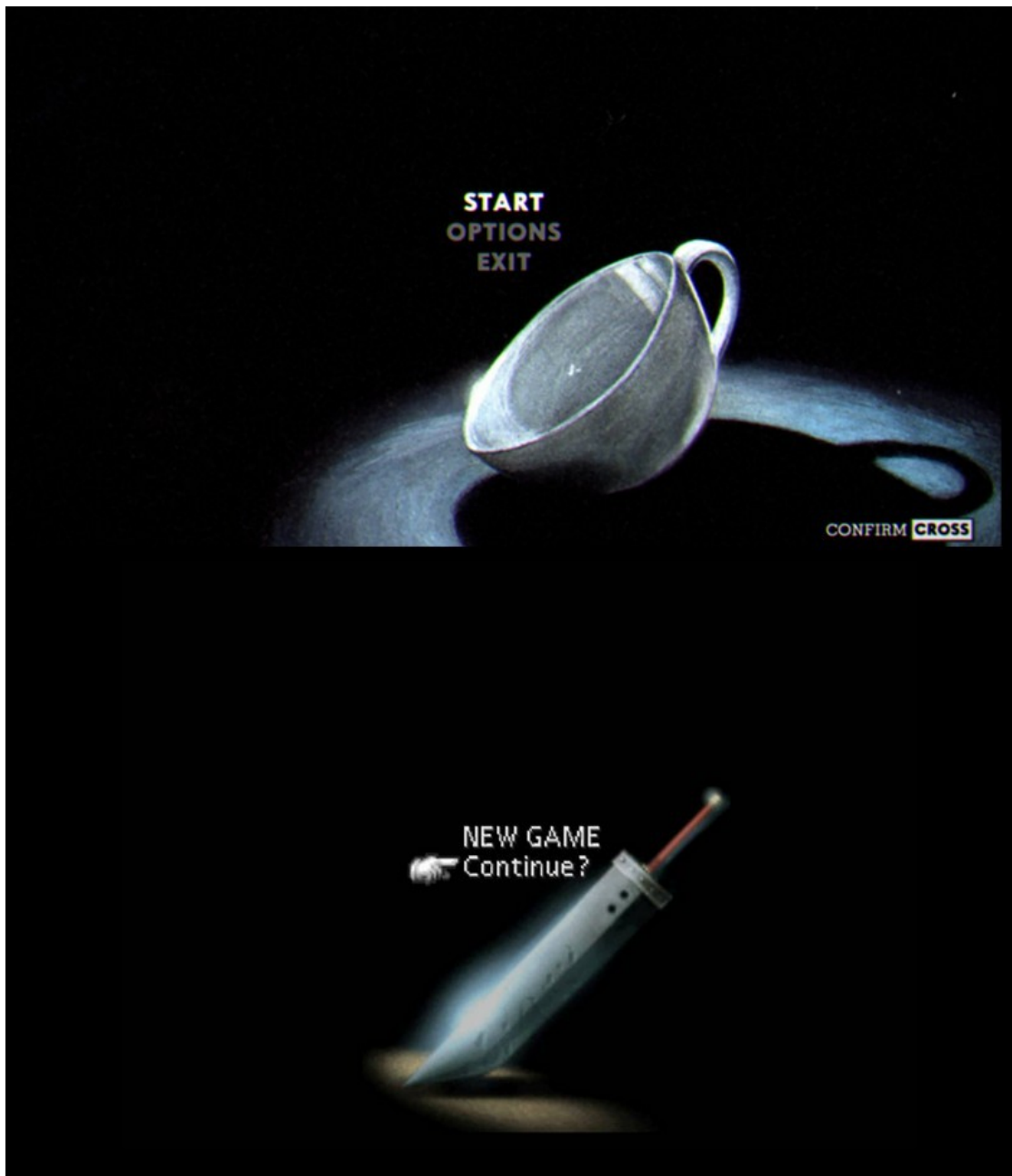


Figura 11. Menù principale di *Cuphead*, ispirato a *Final Fantasy VII* (Square, 1997).

## 1.4 Obiettivi della tesi

La replica virtuale dell'addizionatore è un progetto di HMR che ha scopi di simulazione, di didattica e di comunicazione: la tesi contribuisce alla sua realizzazione.

Nel seguito sono presentati gli obiettivi della tesi e gli strumenti utilizzati per raggiungerli.

### 1.4.1 Realizzazione del sistema di menù

Il progetto, in definitiva la realizzazione di un videogioco abbastanza complesso, è stato segmentato in vari componenti. Nel tirocinio era stata sviluppata l'architettura di gioco integrando il simulatore e realizzando in forma prototipale gli oggetti principali della scena 3D. La tesi prosegue curando la realizzazione definitiva di uno dei componenti fondamentali del gioco: il sistema di menù. In questo contesto, gli obiettivi della tesi possono essere divisi tra qualitativi e quantitativi. I primi sono:

1. progettare un sistema di menù che soddisfacesse le esigenze di *gameplay* e di gestione del gioco;
2. documentare il sistema di menù con diagrammi UML, una notazione standard e diffusa nell'ambito dell'ingegneria del software;
3. definire uno stile consistente ispirato alle tecnologie dell'epoca delle CEP a supporto del coinvolgimento e dell'aggiunta di altri contenuti storici;
4. curare il dettagli grafici e la "coerenza meccanica" del funzionamento degli elementi presenti nei menù;
5. aumentare il livello di coinvolgimento mediante l'utilizzo della grafica 3D e modalità d'interazione in visuale in prima persona.

Gli obiettivi quantitativi sono:

1. introduzione nello stile dei menù di più riferimenti storici possibili;
2. gestione dei contenuti storici in funzione della loro futura espansione;
3. realizzazione di un nutrito insieme di elementi modulari per la costruzione dei menù a supporto della varietà grafica dello stile e di evoluzioni del gioco.

Inoltre l'introduzione della visuale soggettiva in un mondo 3D apre le porte allo sviluppo dello stile grafico fotorealistico che caratterizzerà gli oggetti che popoleranno la stanza dell'addizionatore nelle future versioni del gioco.

Come tutte le tesi sperimentali, fra gli obiettivi ci sono anche l'apprendimento e l'approfondimento dell'uso di diversi strumenti di sviluppo.

## 1.4.2 Strumenti

Lo sviluppo della replica virtuale dell'addizionatore ha richiesto l'utilizzo di:

1. un motore di gioco per realizzare e gestire la grafica e la struttura dell'intero applicativo (scena e interfaccia utente);
2. un software di modellazione 3D per creare la stanza e i componenti della scena;
3. un software di fotoritocco per creare *texture*, *bitmap* e *sprite*;
4. un ambiente di sviluppo per integrare il codice dell'MR57.

Come motore di gioco nel tirocinio ho optato per *Unreal Engine* (Unreal Engine, sito web) nella versione 4.25. Oltre ad essere uno dei più diffusi sul mercato, *Unreal* offre la possibilità di sviluppare per tutte le piattaforme, supporta nativamente il C++ e mette a disposizione molti strumenti proprietari per lavorare nelle diverse aree di sviluppo. Quelli che ho usato maggiormente sono i fogli *Blueprint (BP)* e l'editor di interfacce *Unreal Motion Graphics UI Designer (UMG)*.

*Blueprint Visual Scripting* è un sistema di *scripting* visuale basato sull'utilizzo di grafi. Nello sviluppo del progetto di tesi è stato fondamentale per l'integrazione del codice C++ e la gestione degli eventi di gioco.

*UMG* è un *editor* visuale che serve per realizzare tutti gli elementi dell'interfaccia. In congiunzione con i fogli *BP* è stato lo strumento principale per la realizzazione dei menù.

*Visual Studio 2019 (VS)* (Visual Studio, sito web) è l'ambiente di sviluppo esterno per il codice consigliato da Unreal. È stato utilizzato per integrare il codice del simulatore e creare la libreria di funzioni BP necessaria per integrarlo.



*Blender* (Blender, sito web) è il software di modellazione 3D scelto dalle linee guida di HMR. È servito per realizzare i modelli 3D degli oggetti presenti nella scena. La versione usata è la 2.8.

*Gimp* (Gimp, sito web) è il software di fotoritocco scelto dalle linee guida di HMR. È servito per realizzare le *bitmap* presenti menù, le *texture* degli oggetti 3D e gli *sprite* degli effetti speciali. Insieme a *Unreal* e *VS* è uno dei software che ho utilizzato maggiormente. La versione usata è la 2.10.

## 2. L'interazione dell'utente col gioco

*Definire cosa farà il prodotto prima di progettare come lo farà.*

Alan Cooper, *About Face: The Essentials of Interaction Design*, 1995.

Nei videogiochi è possibile distinguere due tipi principali d'interazione: quella con il mondo di gioco, che avviene solo nella sessione di gioco (in Inglese *in-game*), e quella con l'interfaccia utente (UI), che avviene sia durante la sessione con menù e informazioni sovrapposte alla visuale, tecnicamente dette in *head-up-display* (HUD), sia all'esterno della sessione, per esempio con i menù di pausa e il menù principale.

In questo capitolo vengono descritte le diverse possibilità d'interazione con l'utente e i criteri per la definizione dello stile e la progettazione del sistema di menù.

### 2.1 Il concetto di sessione

La sessione si svolge all'interno di un livello (o mappa) nel quale il giocatore interagisce con l'ambiente di gioco. La sessione comincia al caricamento della mappa e termina quando si ritorna al menù principale o si esce dal gioco. Durante la sessione avvengono tutti gli eventi di gioco: inizializzazione e creazione degli oggetti (*spawn* è il termine tecnico), movimenti del giocatore e degli altri personaggi interpretati da altri giocatori o da intelligenze artificiali, interazione con loro e con la scena e così

via. Il gioco è in effetti una simulazione in cui lo stato del mondo viene aggiornato e presentato sullo schermo a intervalli regolari: con il termine *tick* si intende sia l'evento di aggiornamento che l'intervallo di tempo fra due aggiornamenti successivi. Nello spazio di un tick la macchina (PC, console o dispositivo mobile sono sempre macchine universali) deve eseguire tutto il codice per aggiornare lo stato del mondo e costruire la schermata da presentare al giocatore. Mettere in pausa il gioco significa sospendere il ciclo di tick e quindi la sessione. Nel gioco oggetto di questa tesi, il codice è espresso in C++ o in BP.

A seguire si discutono le soluzioni adottate per realizzare i vari metodi d'interazione dentro e fuori la sessione.

## **2.2 Fuori dalla sessione**

Il menù principale e il menù di pausa sono tipicamente menù esterni alla sessione di gioco. Servono per gestire le opzioni di gioco e di simulazione, consultare i contenuti aggiuntivi e gestire la sessione di gioco: entrare nella sessione, metterla in pausa, uscire dal gioco.

Nella prassi dei videogiochi, il menù principale è lo strumento attraverso il quale si può accedere alla sessione, caricare salvataggi, selezionare la difficoltà, impostare le opzioni di gioco e svolgere altre attività preparatorie come la selezione di mappe, livelli e missioni. Nei giochi di ruolo è il luogo dove si crea il personaggio e se ne cura l'evoluzione, nei contesti multigiocatore in rete il menù principale serve per esempio a scegliere l'*avatar*, a impostare le modalità delle partite e a selezionare i server a cui collegarsi.

I menù di pausa forniscono grosso modo le stesse funzionalità astraendosi dalla sessione, ma senza abbandonarla. Tipicamente durante la navigazione nel menù la sessione è sospesa: il mondo del gioco, nel modello eseguito dal simulatore, resta fermo all'ultimo tick. Nei giochi multigiocatore in rete invece, la prassi è che il menù di pausa non sospenda la sessione.

### 2.2.1 Esempi dal mondo dei videogiochi

Spesso il menù principale rappresenta il primo contatto con il gioco, diventando un vero e proprio biglietto da visita che è possibile sfruttare per fornire all'utente fin da subito indizi sull'ambientazione e sul tipo di esperienza di gioco. Lo stile della schermata iniziale (in inglese *title screen*) e del menù principale si sono evoluti nel corso degli anni.

Nei primi anni '90 le schermate iniziali erano caratterizzate da loghi giganteschi e animazioni molto vivaci: erano un lascito del mondo degli arcade dove servivano per attirare l'attenzione di chi entrava nelle sale giochi (v. fig. 12).



Figura 12. La schermata iniziale di *Sonic the Hedgehog* (Sega, 1991), caratterizzata dalla presenza del grande logo animato.

Quando i titoli diventano scelte personali, i menù adottano stile più minimali che seguono la filosofia del *less is more*, o soluzioni più creative come come le schermate

senza soluzione di continuità (*seamless*), in cui la schermata iniziale o il menù principale anticipano al giocatore l'ambiente della sua avventura (vedi fig 13).



Figura 13. Il menù principale di *Half-Life 2* (Valve, 2004), uno dei primi in stile minimale e con il mondo di gioco sullo sfondo.

Caratterizzato da un'estetica mirata alla pulizia e la semplicità, il menù di *Half-Life 2* utilizza come sfondo ambientazioni selezionate in base ai progressi della partita: anche l'interfaccia del gioco è influenzata dalle azioni dell'utente.

A volte soluzioni originali introducono il mondo di gioco attraverso menù con modalità di navigazione ibride 2D-3D.

In *Bioshock Infinite* (2K Games, 2013) per esempio le pagine del menù vengono stampate sugli elementi di una scena del gioco (insegne, manifesti, cartelli ecc.) rendendo la navigazione nei menù anche una navigazione fisica nel mondo virtuale (v. fig. 14).



Figura 14. Il menù principale di *Bioshock Infinite* (2K Games, 2013). In alto la pagina iniziale, stampata su un'insegna, in basso la pagina dedicata ai contenuti scaricabili, stampata sulla parete di un negozio.

Il menù diventa quindi un'occasione per introdurre l'ambientazione del gioco offrendone uno spaccato vivo e dinamico.

## 2.2.2 Requisiti per la replica virtuale dell'addizionatore a 6 bit

Proponendosi nella forma del videogioco, anche la replica virtuale ha un menù principale e un menù di pausa. I due menù differiscono solo per i pulsanti relativi alle azioni di entrata e uscita dalla sessione e dal gioco. Per il resto, condividono la stessa organizzazione:

1. *pagina principale* (v. fig 15): funziona da indice e presenta i collegamenti alle altre pagine e i pulsanti per entrare/uscire dal gioco (menù principale) o quelli per tornare alla sessione/menù principale (menù di pausa);
2. *pagina delle opzioni*: presenta i collegamenti alle pagine delle opzioni grafiche e di simulazione e permette di impostare la lingua, i livelli audio, la temperatura ambiente, la visualizzazione delle informazioni di simulazione;
3. *pagina delle opzioni grafiche*: accessibile dalla pagina delle opzioni, permette di impostare i parametri della scalabilità grafica che il giocatore può personalizzare in base alle prestazioni del suo hardware;
4. *pagina delle opzioni di simulazione*: accessibile attraverso la pagina delle opzioni, è destinata ai giocatori esperti che vogliono sperimentare modificando i parametri del modello di simulazione;
5. *pagina dei controlli*: è una guida di riferimento per presentare al giocatore l'elenco completo dei tasti e delle azioni a cui sono correlati;
6. *pagina dell'archivio*: permette di consultare i contenuti documentali associati alla storia dell'addizionatore e dintorni; l'archivio è ulteriormente strutturato in sezioni, ognuna delle quali copre un argomento con un breve testo e una galleria di immagini;
7. *pagina dei crediti*: presenta le note di copyright e le informazioni relative al contesto e alle persone che hanno realizzato il gioco;

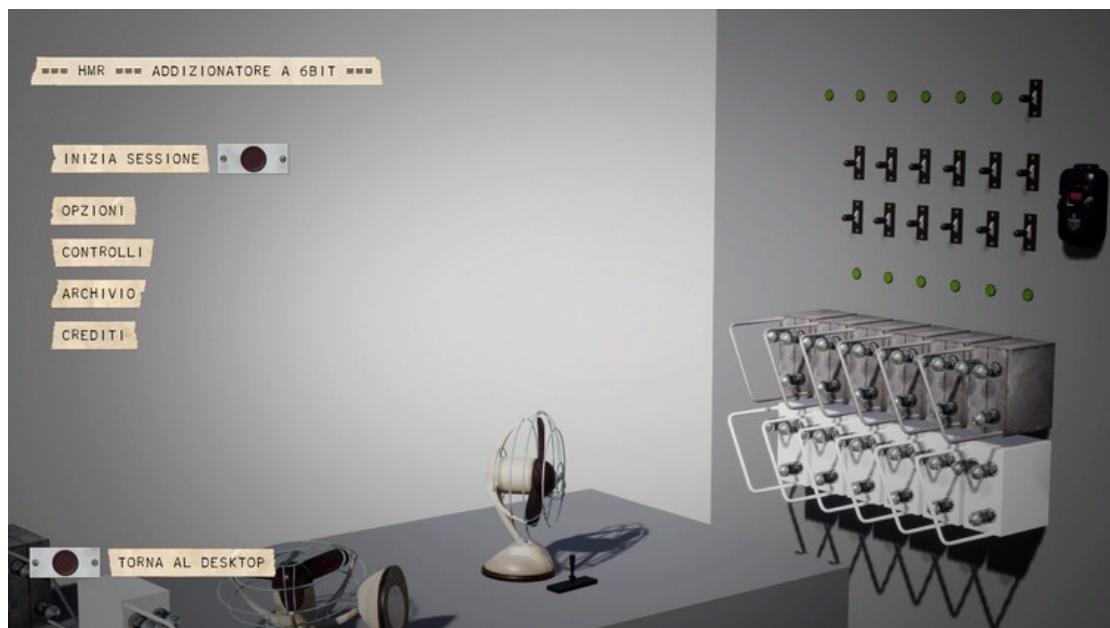


Figura 15. Il menù principale così come appare nel gioco.

## 2.3 Dentro la sessione, menù in sovrapposizione

Per compiere o semplificare alcune interazioni complesse all'interno della sessione, può essere necessario ricorrere all'utilizzo di elementi d'interfaccia o di veri e propri menù in sovrapposizione alla visuale del giocatore. Inventari, ghiera di scelta rapida, menù di fabbricazione e modifica degli strumenti, menù di selezione dei dialoghi e delle missioni, sono solo alcuni degli esempi che la storia e l'attualità dei videogiochi ci offrono. In alcuni casi, come ad esempio le avventure grafiche o i giochi di ruolo a turni, quasi tutta l'esperienza di gioco si svolge attraverso questo tipo di oggetti.

### 2.3.1 Esempi dal mondo dei videogiochi

*Among Us* (Innersloth 2018), un *party game* di deduzione ispirato al classico “guardia e ladri” giocato in cortile ma a tema avventure spaziali, è uno titolo in cui i menù in sovrapposizione ricoprono un ruolo fondamentale. I giocatori impersonano i membri di una missione spaziale, ma alcuni di loro sono degli impostori alieni. Ogni astronauta ha delle mansioni da svolgere in alcune aree della mappa, gli impostori



devono impedire che le compiano sabotando il loro lavoro e uccidendoli. Sia le mansioni che il loro sabotaggio vengono svolti attraverso dei minigiochi in menù in sovrapposizione . Le mansioni non interrompono la sessione, il tempo e gli eventi del gioco continuano a svolgersi sullo sfondo (v. fig. 16).

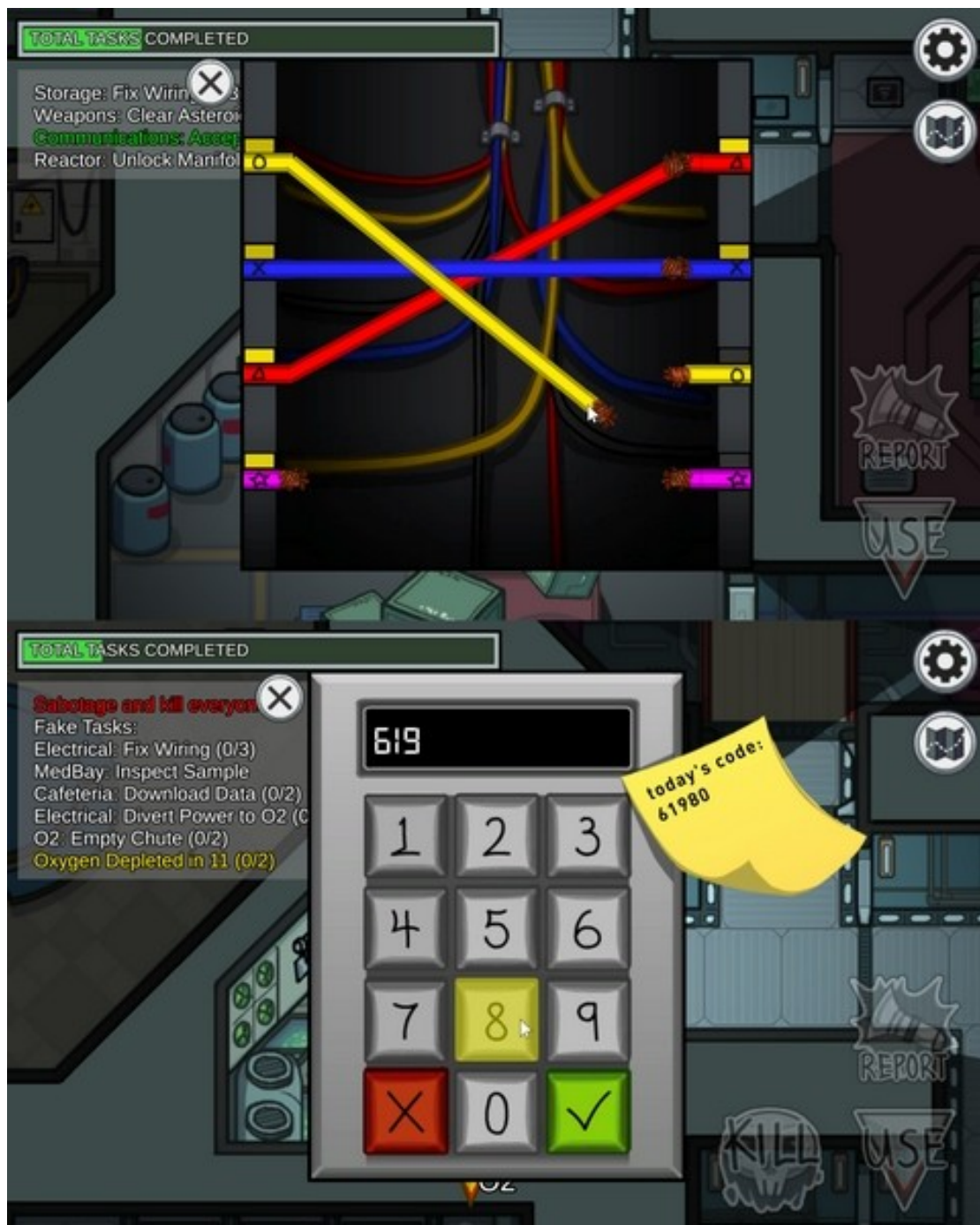


Figura 16. Un esempio di menù in sovrapposizione da *Among Us* (Innersloth, 2018). In alto una mansione in cui il giocatore deve collegare dei fili elettrici, in basso una che richiede l'immissione di un codice. Sullo sfondo è possibile riconoscere la scena di gioco (in prospettiva isometrica).

In *Minecraft* (Mojang, 2009), gioco *sandbox* (v. glossario) ambientato in un mondo fatto di cubi, è possibile fabbricare oggetti estraendo o recuperando materiali dalla scena e ricombinandoli; per questa funzione viene utilizzato un menù in sessione in sovrapposizione. Il processo di fabbricazione avviene senza che la sessione sia sospesa (v. fig. 17).



Figura 17. Menù di fabbricazione di *Minecraft* (Mojang, 2009). Il giocatore attinge ai materiali presenti nel suo inventario (sezione in basso) e li piazza nella zona di fabbricazione (in alto a sinistra) ricevendo l'oggetto fabbricato (in alto a destra). Sullo sfondo è possibile intravedere il mondo di gioco.

In *GTA V* per accedere al proprio arsenale è possibile richiamare un menù che consente al giocatore di impugnare un'arma o di aggiungerla alla rosa di selezione rapida. Anche in questo caso si tratta di un menù in sovrapposizione, ma, per aiutare il giocatore, la sessione è rallentata. Di fatto il simulatore agisce sul tick: o aumenta l'intervallo di tempo, o nei giochi più sofisticati che vogliono mostrare la scena rallentata senza scatti, il tick viene mantenuto, ma il simulatore calcola gli aggiornamenti del mondo su tempo più piccolo (v. fig. 18).



Figura 18. Rosa delle armi in *Grand Theft Auto V* (Rockstar North, 2013). In questo momento il giocatore è disarmato (selettore sull'icona del pugno a mani nude) e sulla rosa sono presenti le icone delle armi a disposizione. Sullo sfondo è possibile osservare il mondo di gioco.

### 2.3.2 Requisiti per la replica virtuale dell'addizionatore a 6 bit

Nella replica virtuale dell'addizionatore c'è l'esigenza di interagire con i componenti dell'addizionatore, montare e smontare i telaietti, monitorare i parametri di simulazione. Ragionando in termini di sessione, queste azioni si svolgono senza interromperla.

Il *menù di montaggio* dei telaietti è un menù in sovrapposizione che permette di montare e smontare i telaietti dell'addizionatore (v. fig. 19). Il menù rappresenta in forma schematica lo stato di montaggio dei telaietti e la loro ubicazione corrente. Occupa circa metà dello schermo, caratteristica che lo rende utilizzabile insieme al menù *SimInfo* e che permette di vedere cosa succede nella scena durante l'utilizzo del menù. Per fare uno spostamento, l'utente deve selezionare una posizione occupata da un telaietto e una vuota in cui spostarlo cliccando sulle icone. Sui display meccanici appaiono gli identificatori delle posizioni selezionate, lo spostamento deve essere

confermato cliccando sul pulsante etichettato “sposta”. È presente anche un *selettore a rotella singola* (v. §2.5) per mettere in pausa il gioco, permettendo di pensare con calma a sperimentare con i telaietti senza correre il rischio che l’addizionatore si surriscaldi nel frattempo. Il tasto su cui è stato mappato il menù è “M”.

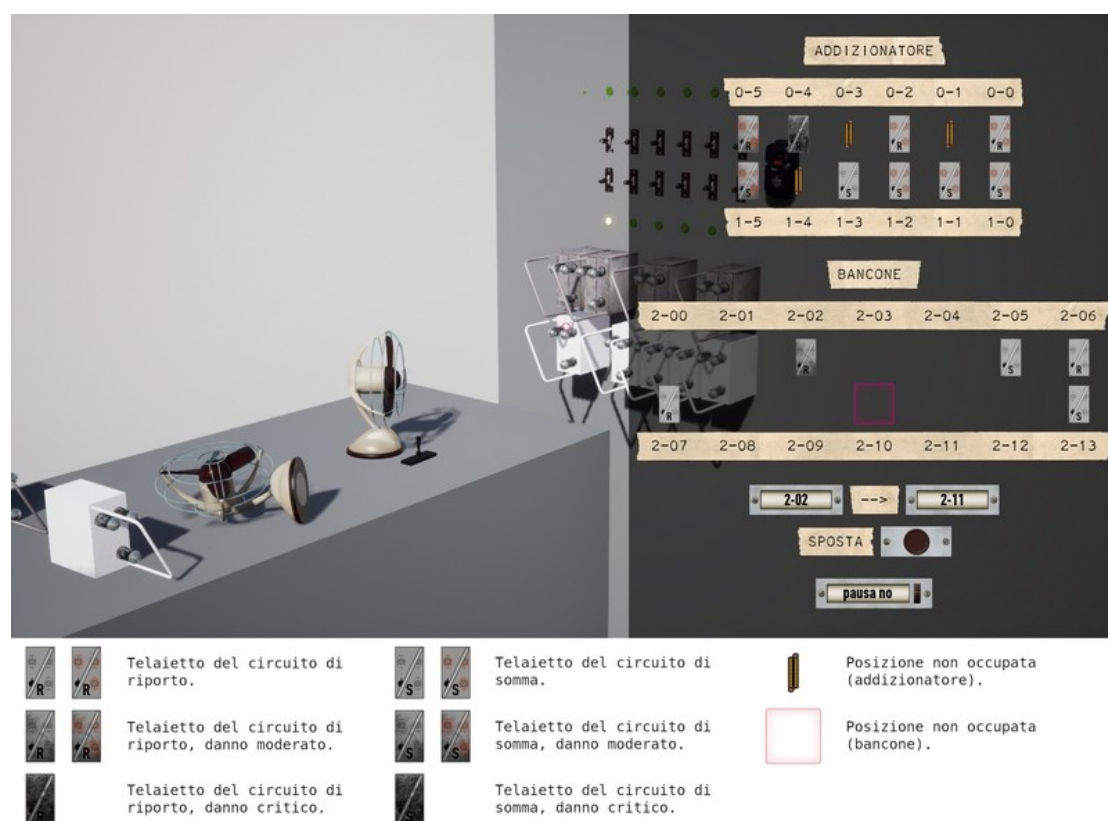


Figura 19. Il menù di montaggio come appare nel gioco, sotto la schermata è presente la legenda delle icone utilizzate nel menù.

Il *SimInfo* è un menù in sovrapposizione (v. fig. 20) in sola lettura che serve per controllare in tempo reale i dettagli della simulazione: le temperature, lo stato di riscaldamento e di danneggiamento dei singoli telaietti dell’addizionatore, lo stato di addendi e risultati, la temperatura massima e minima raggiungibili dall’addizionatore in base alla temperatura ambiente selezionata, la velocità di rotazione del ventilatore. Organizzato in tre sezioni principali (*general info, rack, workbench*), la prima è dedicata alle informazioni generali quali velocità di rotazione del ventilatore, numero



funzione di Unreal per mostrare in sovrapposizione nell'HUD gli *fps* correnti. Il tasto su cui è stata mappata è “U”.

## 2.4 Dentro la sessione, interazione con la scena 3D

Nei giochi in grafica tridimensionale la mappa in cui si svolge la sessione di gioco è chiamata scena 3D. La scena può essere popolata da oggetti di scena (*props*) e oggetti attore, con i quali è possibile interagire. La gestione dell'interazione all'interno della scena è uno degli aspetti più problematici dei videogiochi: sono necessarie molte semplificazioni per collegare azioni complesse a limitate possibilità di input. In questo senso i dispositivi VR stanno cercando di avvicinarsi alla complessità della realtà utilizzando tecnologie che tracciano il movimento delle dita e del corpo nello spazio, ma per quanto gli sviluppi recenti abbiano portato a risultati impressionanti come quelli di *Half Life Alyx* (Valve, 2020), la strada verso il completo utilizzo dei movimenti del corpo in uno spazio virtuale presenta ancora molte sfide. Inoltre continueranno comunque a esistere videogiochi con meccaniche di gioco troppo complesse o inadatte all'utilizzo della realtà virtuale, e con essi i problemi relativi all'interazione.

Nei paragrafi seguenti vengono descritte le soluzioni tecniche per l'interazione nella scena, sviluppate durante il tirocinio, e le caratteristiche degli oggetti attore presenti nel mondo di gioco.

### 2.4.1 Esempi dal mondo dei videogiochi

Per far sì che l'esperienza nel mondo di gioco non risulti poco chiara o frustrante, l'interazione deve essere suggerita, attuata e confermata.

L'interazione nella scena 3D è basata sulle collisioni. Gli oggetti con cui si può interagire sono dotati di un box di collisione (in inglese *hitbox*) per rilevare la compenetrazione con altri oggetti. Anche interazioni semplici, come il fatto che un oggetto non attore è di ostacolo al movimento in una certa direzione sono gestite così. In *Serious Sam 3: BFE* (Devolver Digital, 2011), uno *sparatutto in prima persona* (v. glossario), l'interazione con gli oggetti attore è suggerita evidenziandone il contorno:

il giocatore capisce che gli unici oggetti con i quali può interagire sono quelli che luccicano. Per compiere l'interazione, il gioco si affida al contatto fra il box di collisione del giocatore con quella di un oggetto attore: andandogli addosso il giocatore ottiene l'oggetto. L'interazione è confermata da un suono e dalla comparsa di un testo nell'HUD (v. fig. 21).





Figura 21. L'interazione con gli oggetti in *Serious Sam 3: BFE* (Devolver Digital, 2011). Gli oggetti interattivi sono segnalati con un contorno luminoso lampeggiante. Il giocatore, avvicinandosi, li raccoglie e compare un testo nell'HUD (in basso).

In *Bioshock* (2k Games, 2007), un altro sparattutto in prima persona, l'interazione è suggerita in modo simile, ma essendo disponibili più azioni, quando il giocatore punta la visuale su di un oggetto, appare un testo che segnala le azioni possibili e il tasto

corrispondente da premere. Anche in questo caso l'interazione è confermata da un suono, da un'animazione e dalla segnalazione nell'HUD (v. fig. 22).



Figura 22. L'interazione con gli oggetti in *Bioshock* (2K Games, 2007). A sinistra una radio presente nella scena, a destra la stessa radio segnala l'interazione luccicando e suggerendo al giocatore il tasto da premere.

#### **2.4.2 Requisiti per la replica virtuale dell'addizionatore a 6 bit**

Il giocatore può muoversi liberamente utilizzando mouse e tastiera. Le azioni di movimento sono state mappate sui tasti "W", "A", "S", "D", mentre la gestione della visuale e dell'interazione nella scena 3D viene gestita dal mouse.

La scena riproduce la stanza di Palazzo Matteucci in cui furono costruiti l'addizionatore e le CEP. Molti degli attori e degli oggetti di scena hanno attualmente modelli semplificati o addirittura semplici solidi segnaposto (v. fig. 23).

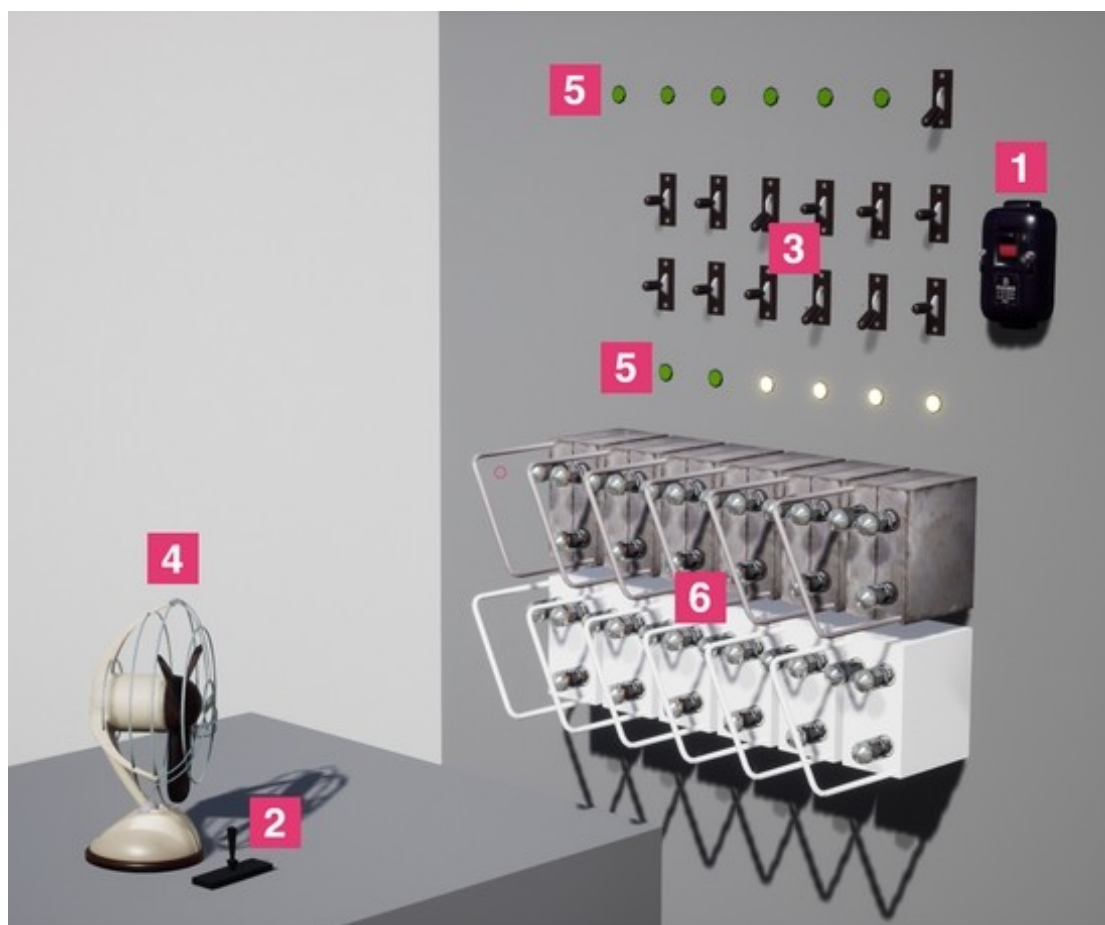


Figura 23. Gli attori del gruppo addizionale. Ogni classe di attori è identificata da un numero.

Allo stato attuale, a parte la gestione degli oggetti come ostacoli, il giocatore può interagire solo con gli attori del gruppo addizionale.

#### 2.4.2.1 Interruttore generale dell'addizionale

Un classico interruttore magnetotermico bipolare; ha due posizioni, circuito aperto e chiuso. L'interazione, rilevata con un box di collisione e implementata con codice BP, esegue le chiamate al simulatore per aggiornare lo stato, anima lo spostamento della levetta e riproduce il suono dello scatto (v. fig. 23, numero 1).

#### 2.4.2.2 Interruttore del ventilatore

Un classico interruttore passante a due posizioni. L'interazione viene rilevata con un box di collisione e implementata con codice BP, esegue le chiamate al simulatore per

aggiornare lo stato e riprodurre l'animazione della levetta e il suono dello scatto (v. fig. 23, numero 2).

#### **2.4.2.3 Chiavi telefoniche per gli addendi dell'addizionatore**

Interruttori tipici dei centralini telefonici, avevano tre posizioni: centrale di circuito aperto, in un verso pulsante, usata per provare la linea, nell'altro stabile di circuito chiuso, per mantenerla una volta connessa. Per queste caratteristiche erano usate nel quadro di controllo della prima CEP e sono state adottate anche nella ricostruzione dell'addizionatore a 6 bit; i due tipi di interazione sono rilevati con due box di collisione e implementati con codice BP che esegue le chiamate al simulatore per aggiornare lo stato, anima la levetta nelle due differenti modalità e riproduce i suoni dei due movimenti (v. fig. 23, numero 3).

#### **2.4.2.4 Ventilatore Marelli**

Come documentato dalle foto d'archivio, era usato per raffreddare l'unità aritmetico logica della prima CEP; è stato inserito nella scena con la stessa funzione nei confronti dell'addizionatore. Il modello termodinamico include l'azione del ventilatore e l'utente deve usarlo per controllare la temperatura della macchina; il ventilatore reagisce all'interazione con il suo interruttore accendendosi e spengendosi, il codice BP esegue le chiamate al simulatore per ottenere la posizione delle pale visualizzando la corretta velocità di rotazione e cambiandone il mesh, solido o trasparente, in funzione della velocità di rotazione. Il codice BP riproduce il suono, modulandolo sempre rispetto alla velocità (v. fig. 23. numero 4).

#### **2.4.2.5 Indicatori luminosi**

Erano triodi a catodo freddo pieni di gas luminescente, utilizzati nei pannelli di controllo di molti dei primi calcolatori. Nella replica virtuale, attraverso il codice BP ottengono dal simulatore il loro livello di illuminazione (v. fig. 23. numero 5). Per simulare i vari livelli di luminosità, utilizzano un materiale emissivo gestito dal codice BP (v. §3.1.3).

#### **2.4.2.6 Telaietti**

Alloggiavano i circuiti di somma e di riporto dell'addizionatore ed erano caratterizzati dalle valvole termoioniche a vista. Attraverso il codice BP ottengono dal simulatore il

livello di illuminazione delle resistenze delle valvole e il livello di danno dei circuiti. Nel caso in cui il telaietto si surriscaldi, il codice BP attiva effetti sonori e visivi corrispondenti al livello di danno corrente (v. fig. 23, numero 6).

#### **2.4.2.7 Attori e interazione in Unreal**

Ogni oggetto attore è un *asset* (v. glossario) definito in Unreal come una classe BP di tipo attore; il suo stato, incluse eventuali collisioni, è aggiornato a ogni tick. Attraverso il codice BP possono essere definiti i suoi componenti (e.g. hitbox, modelli 3D, ecc.) e il suo comportamento (e.g. gestione degli eventi, codice da svolgere a ogni tick, ecc.). Il codice BP della classe vale per ogni istanza della classe presente sulla scena 3D.

Nel tirocinio, per realizzare il suggerimento dell'interazione si è sfruttata la collisione con la retta ideale che rappresenta dove guarda il giocatore (*line tracing*).

L'oggetto "guardato" si tinge di *HMR Deep Pink 57* (#E43972), il colore caratteristico di HMR, che, nella convenzione dell'interfaccia utente, rappresenta l'interattività meccanica (vedi §2.5.5). Così l'utente solo guardandosi in giro distingue gli oggetti interattivi da quelli statici. Un cerchietto al centro della visuale aiuta il giocatore a capire meglio dove sta guardando (v. fig. 24).

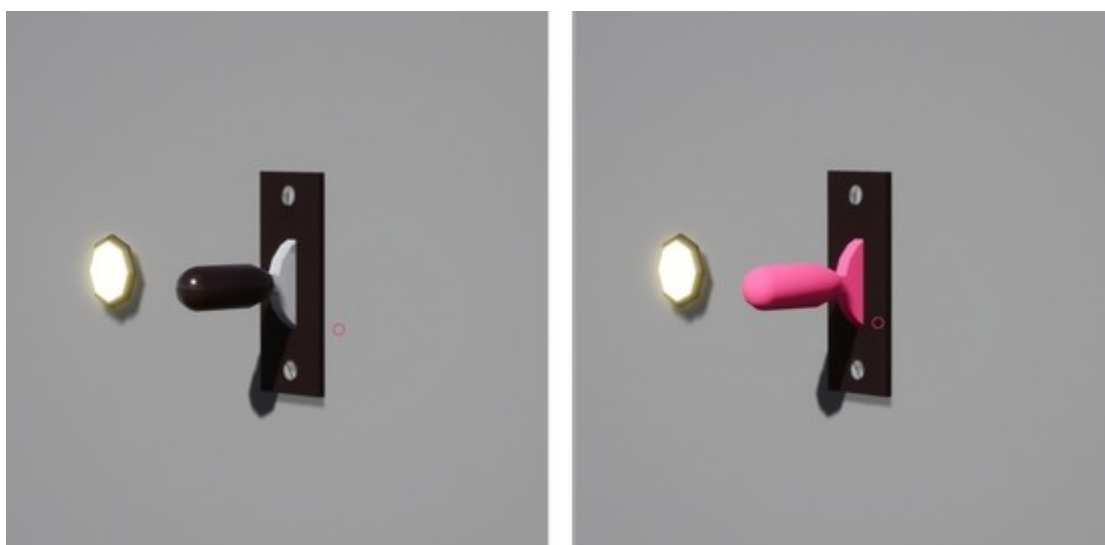


Figura 24. Una chiave telefonica reagisce allo sguardo del giocatore. A sinistra il cerchietto non è allineato con la chiave. A destra il giocatore sposta la visuale e la chiave suggerisce l'interazione.

Il tasto sinistro del mouse attua l'interazione: il click genera l'evento di inizio interazione, che viene notificato all'attore con una chiamata BP. Per confermare l'interazione, l'attore riproduce un'animazione con un suono appropriato: per esempio, un interruttore cambia posizione e si sente lo scatto.

Un'altra interazione possibile all'interno della scena 3D è lo zoom: non c'è suggerimento perché è attivabile in ogni momento tenendo premuto il tasto destro del mouse (v. fig. 25).

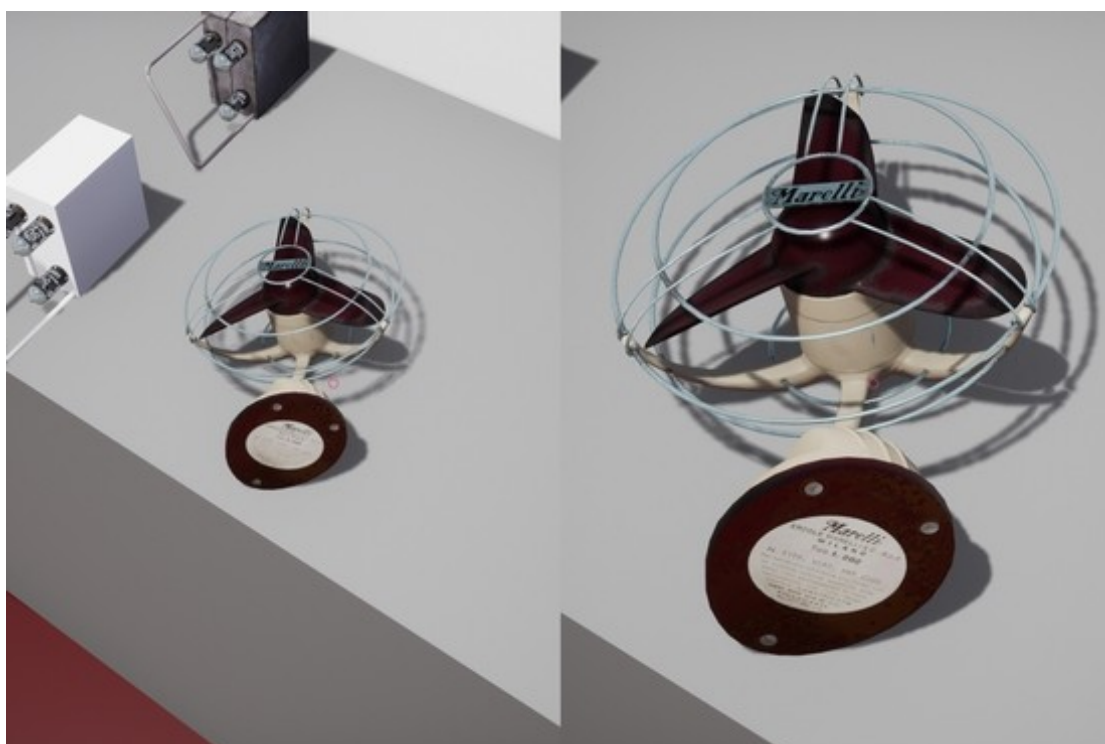


Figura 25. A sinistra il particolare del ventilatore poggiato su un solido segnaposto, a destra lo stesso ventilatore visto con lo zoom attivo.

Rientra tra le soluzioni per l'interazione nella scena anche il menù di montaggio (v. §2.3.2): sebbene l'interazione avvenga nel menù, la conferma riguarda anche la posizione degli oggetti nella scena 3D.

Tenendo premuto il tasto "L" (*labels*) appaiono nel mondo di gioco le etichette fluttuanti che indicano, con le stesse convenzioni usate nel menù di montaggio, le possibili posizioni dei telaietti sull'addizionale e sul bancone da lavoro (v. fig. 26).

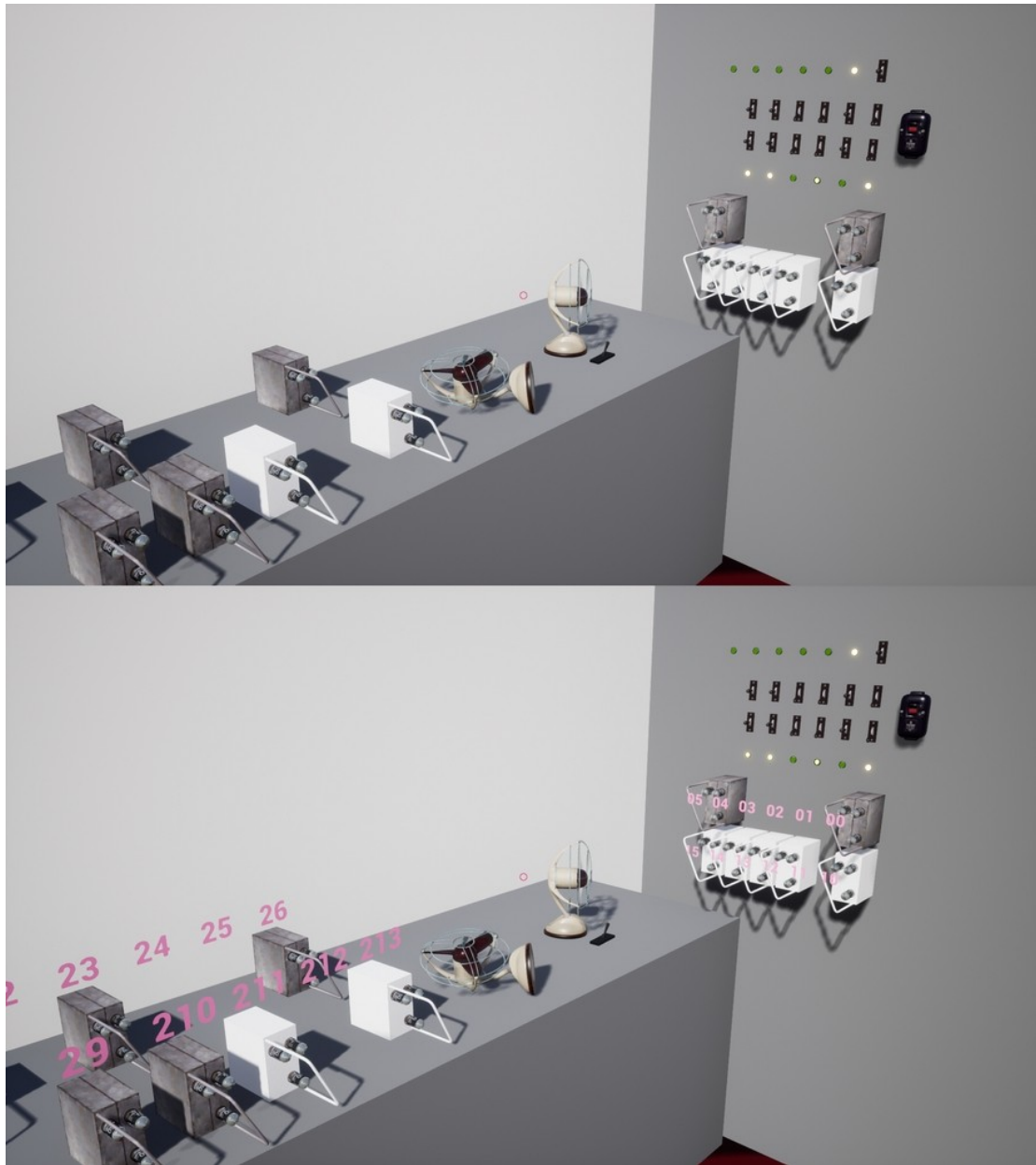


Figura 26. La scena senza e con le etichette.

## 2.5 Lo stile dei menù: criteri e soluzioni

Una buona UI, valida sia dal punto di vista grafico che dell'usabilità, incide positivamente sull'esperienza di gioco (Ng e altri, 2018). L'utente non deve sentirsi



ostacolato: è importante organizzare i menù su livelli che seguono una struttura semplice e ben organizzata (Desurvire e altri 2004). Lo stile adottato deve coinvolgere, non risultare alieno al contesto e soprattutto non dare l'idea di oggetto non curato, come per esempio accade con *Gioventù Ribelle* (v. §1.3.5).

Secondo Csikszentmihalyi (Csikszentmihalyi 1996) il *flow* è uno stato di profonda concentrazione e divertimento che si sperimenta quando si è totalmente assorti in un'attività. Molti studi (Johnson e Wiles 2003; Khalid e altri 2006; Ng 2018) sostengono, riportando esperimenti che hanno coinvolto numerosi giocatori, che il design delle interfacce è uno degli elementi che più contribuisce a innescare e mantenere il flow, sottolineando l'importanza del bilanciamento tra la presentazione grafica e gli aspetti funzionali quali consistenza e leggibilità.

Uno stile conforme al contesto di gioco si presta all'inserimento di particolari che contribuiscono a raccontare l'ambientazione attraverso la costruzione di un immaginario percepito immediatamente anche se non compreso nei dettagli (Malone 1982). Inoltre, come ricordano i fratelli Moldenheuer nel libro *The Art of Cuphead* (Cymet e altri, 2020, p. 233), non importa quanti giocatori saranno in grado di cogliere i dettagli, l'importante è che i dettagli ci siano per il piacere di quelli che li sapranno vedere.

Seguendo questa linea, la tesi ha sfruttato i menù della replica virtuale dell'addizionale per introdurre riferimenti a tecnologie e idee degli albori dell'informatica con uno stile che presta particolare attenzione al funzionamento e alla coerenza meccanica di ogni elemento presente nella UI.

Nei paragrafi successivi è discussa la realizzazione del sistema di menù, sia dal punto di vista funzionale che dello stile e dei suoi riferimenti storici.

### **2.5.1 Interazione nel sistema di menù: classi di interazione**

Per coniugare funzionalità e riferimenti storici è stato definito un "linguaggio meccanico dell'interfaccia" che utilizza come componente semantica alcune categorie di azioni, mentre come componente sintattica l'aspetto grafico.

Le categorie di azioni sono:

1. *navigazione tra pagine di menù*; è la navigazione che si ha nei livelli di un singolo menù;
2. *navigazione tra menù*; è la navigazione che si ha tra due menù diversi (e.g. dal menù di pausa al menù principale);
3. *applicazione di azioni*; sono le azioni che hanno effetto sulla sessione, sulla simulazione o sulle impostazioni del gioco (e.g. pulsante *riprendi*, pulsante *sposta*, pulsante *applica*);
4. *selezione di parametri*; il giocatore può scegliere un parametro da un insieme di parametri definito (e.g. selezione della lingua, scalabilità delle opzioni grafiche);
5. *modifica di parametri in un intervallo*; il giocatore può immettere un valore in un intervallo definito (e.g. temperatura ambiente, livello dei volumi audio);
6. *modifica libera di parametri*; il giocatore può immettere un valore a piacere (e.g. opzioni di simulazione avanzate);
7. *lettura di parametri*; il giocatore può visualizzare i parametri che ha selezionato (e.g. gli indici di posizione dei telaietti da spostare);
8. *navigazione tra immagini*; il giocatore può navigare tra le immagini delle gallerie dell'archivio.

Queste categorie sono state la base per la realizzazione delle classi di *widget* che popolano la UI. Nella terminologia UMG, con il termine *widget* si fa riferimento ai singoli elementi grafici che compongono l'interfaccia. Possono anche essere assemblati tra di loro, il che offre la possibilità di creare *widget* modulari e componibili a seconda delle esigenze.

Le classi di *widget* più significative della replica virtuale dell'addizionatore sono:

1. *zona telegrafica*; è un widget di tipo componente utilizzato negli altri widget per stampare la stringa descrittiva dell'elemento;
2. *pulsante meccanico*; è un widget di tipo componente; costituisce la parte interattiva all'interno di altri widget;
3. *rotella di bachelite*; è un widget di tipo componente utilizzato come componente interattivo nei widget dei contatori e selettori meccanici;

4. *bottonone link*; è il widget utilizzato per la navigazione tra pagine di menù; i suoi moduli sono un widget zona e un widget bottone;
5. *pulsante per la navigazione tra menù*; è utilizzato per la navigazione tra menù comunicanti; è composto da un widget pulsante meccanico (a sinistra) e un widget zona (alla destra del pulsante);
6. *pulsante per l'applicazione di azioni*; è utilizzato per l'applicazione di azioni; i suoi moduli sono un widget zona (a sinistra) e un widget pulsante meccanico (alla destra della zona);
7. *selettore a rotella singola*; è utilizzato per la selezione di parametri; è un widget composto da un display per il parametro e un widget rotella per effettuare la selezione;
8. *contatore a tripla rotella*; è utilizzato per la modifica libera di parametri; è un widget composto da un display per il parametro e tre widget rotella per modificarlo agendo sulle singole cifre;
9. *display meccanico*; è utilizzato per la lettura di parametri; è un widget composto da un display per la visualizzazione del parametro;
10. *slider meccanico*; è utilizzato per la modifica di parametri in un intervallo; consiste in un widget di tipo slider ed è dotato di un display per visualizzare il valore corrente del parametro.

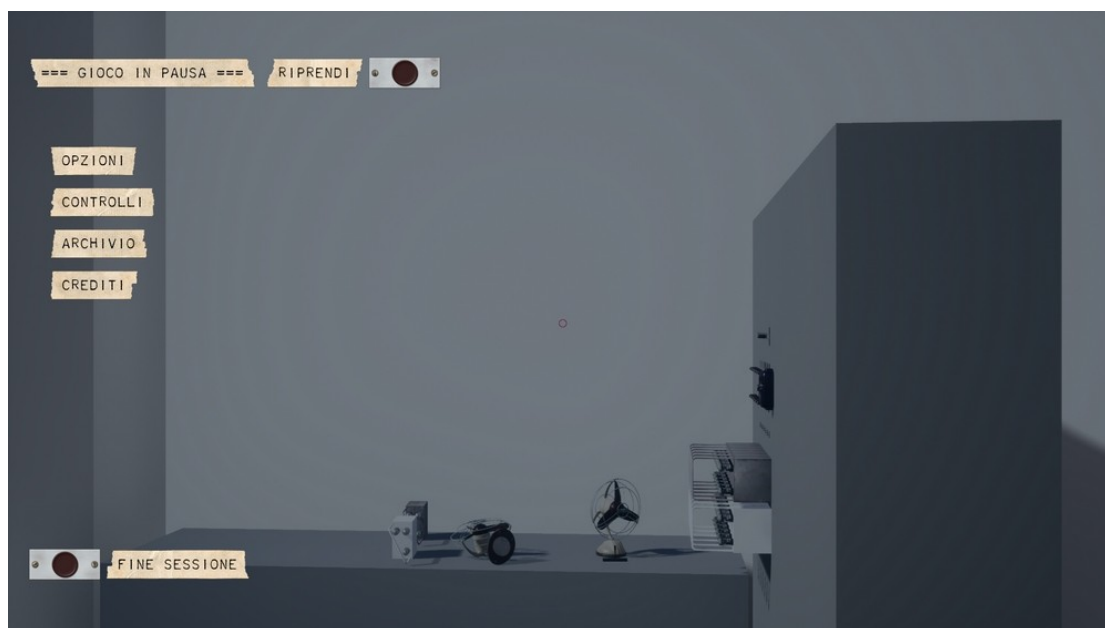


Figura 27. Pagina principale del menù di pausa così come appare nel gioco.

### **2.5.2 Riferimenti storici al mondo telescriventi**

Le telescriventi erano dispositivi utilizzati per le comunicazioni telegrafiche a partire dalla fine del 1800. Adottate dai primi calcolatori elettronici come dispositivi di ingresso/uscita, potevano stampare su foglio o su nastro di carta, detto zona. La stampa su zona offriva il vantaggio pratico di non doversi preoccupare, in fase di trasmissione, di inviare anche i caratteri speciali di accapo e di ritorno carrello. Infatti, nella prassi telegrafica, erano gli operatori che provvedevano a tagliare le zone e incollare manualmente le righe sul modulo del telegramma (v. fig. 28 e 29).

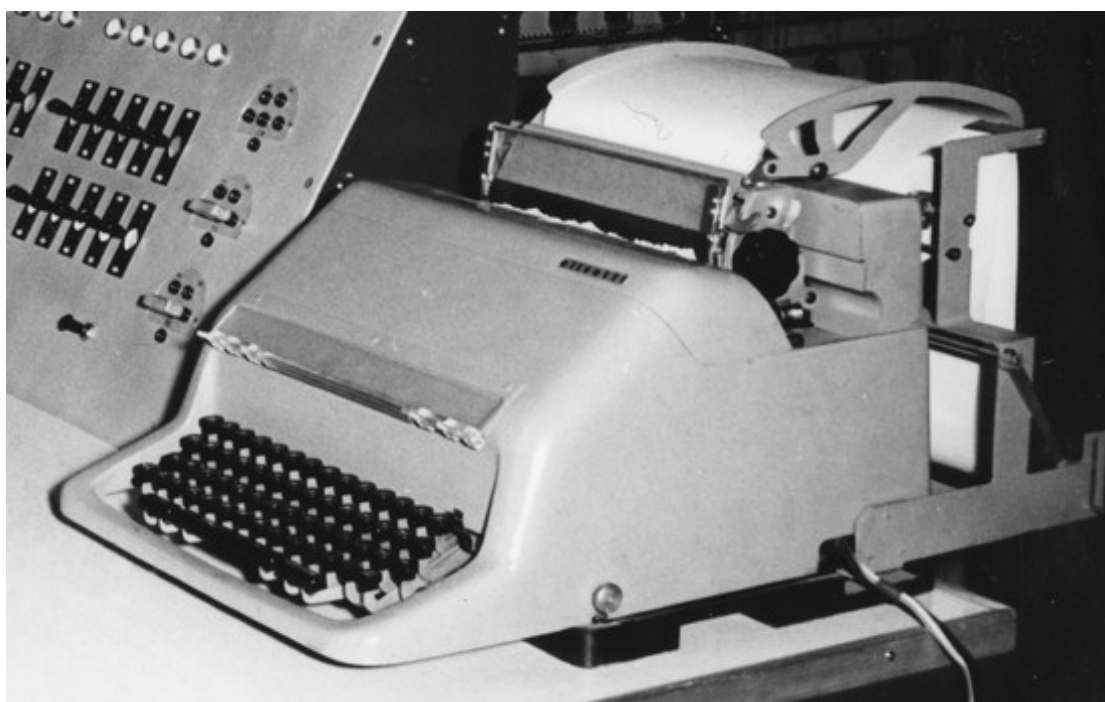


Figura 28. Una telescrivente della prima CEP. (Foto della prima CEP, 1957. Archivio Generale di Ateneo dell'Università di Pisa, dettaglio).



### 2.5.3 Riferimenti storici al memex

Presentato per la prima volta nel luglio 1945 sulla rivista *The Atlantic Monthly* nell'articolo "*As We May Think*" (Bush 1945a), successivamente ripreso da altri periodici in versione più breve e corredata di illustrazioni, il memex (contrazione di "memory expansion") era una macchina elettromeccanica ideata da Vannevar Bush e mai effettivamente realizzata. Consisteva in una scrivania in cui erano integrati tutti gli strumenti per catalogare e recuperare i documenti di un ricercatore in modo automatico. Secondo l'idea di Bush sarebbe anche dovuto essere dotato di una fotocamera che avrebbe permesso di archiviare informazioni scattando foto su microfilm ai documenti cartacei. Il tutto sarebbe stato reso disponibile per la consultazione su due schermi traslucidi posti su dei supporti inclinati rispetto al piano della scrivania (v. fig. 30).

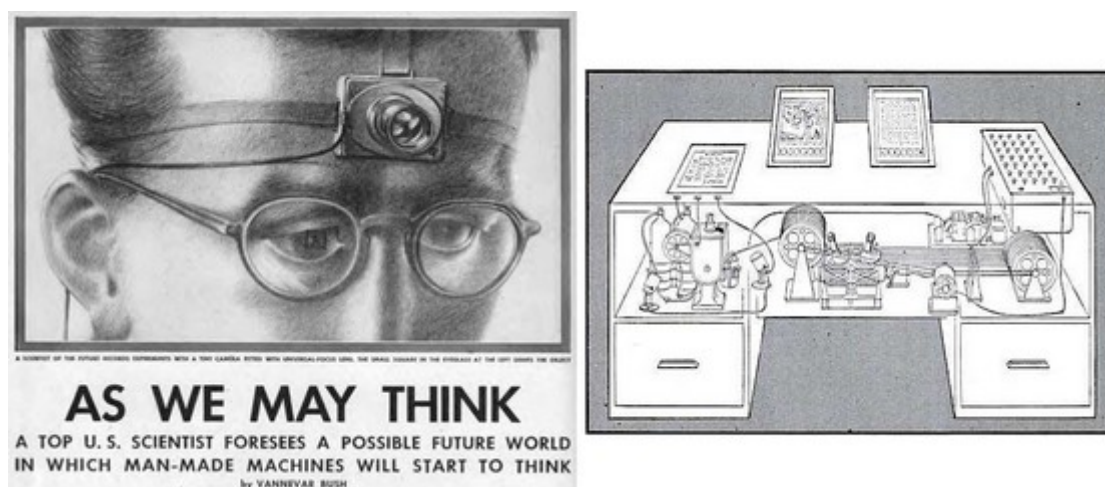


Figura 30. Il particolare della prima pagina dell'articolo e uno schema del memex pubblicati sulle pagine di *Life Magazine* (Bush 1945b) nel settembre 1945.

Il memex è stato l'ispirazione principale per realizzare la pagina dell'archivio del sistema di menù, in cui è possibile notare l'organizzazione "a due schermi ottimizzati" e la stampa dei documenti d'archivio sui fogli di carta tecnicamente archiviati nella scrivania (v. fig. 31).



Figura 31. La pagina dell'archivio così come appare nel gioco.

Nelle altre pagine di menù è possibile notare riferimenti più generici alla storia dell'informatica quali l'utilizzo di materiali come metallo e bachelite.

#### **2.5.4 Stile telegrafico**

Ispirato al mondo delle telescriventi, lo stile telegrafico caratterizza le parti cartacee e testuali dei menù (v. fig. 32).



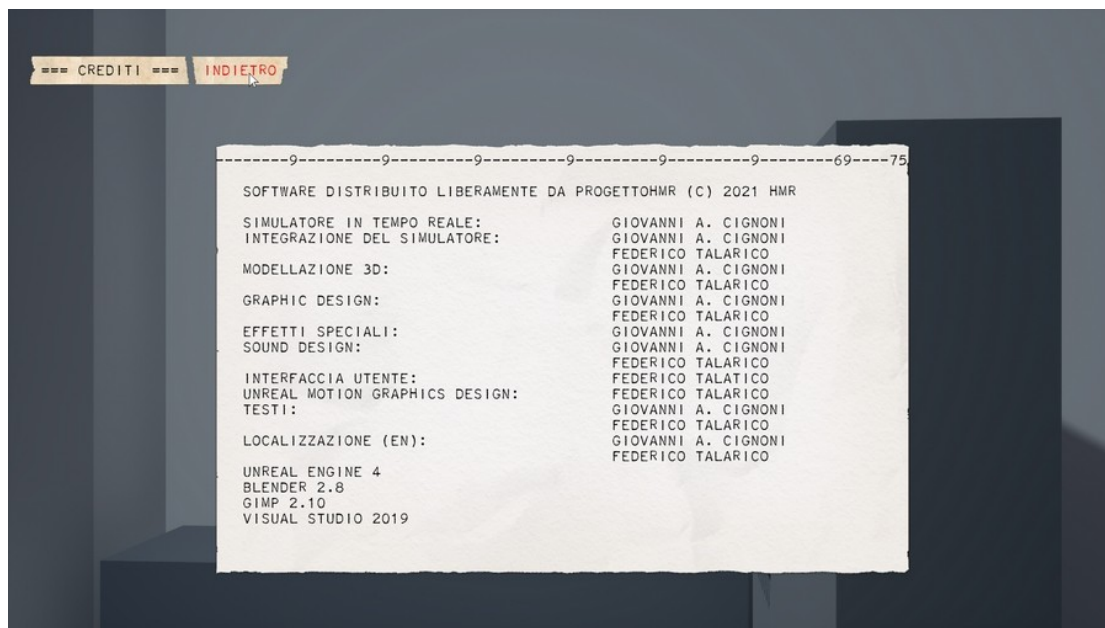


Figura 32. La pagina dei controlli. In alto è possibile notare due widget a zona telegrafica, mentre il contenuto della pagina è “stampato con la telescrivente” su foglio di carta.

I riferimenti storici e la coerenza meccanica sono mantenuti sia nell’aspetto grafico che nelle modalità d’interazione. L’idea di fondo è quella che il sistema di menù sia stato realizzato dai ricercatori del progetto CEP utilizzando le telescriventi:

1. ogni testo stampato su carta o zona telegrafica utilizza il font delle telescriventi;
2. ogni testo ha la stessa dimensione (le telescriventi avevano una sola dimensione del carattere);
3. i *widget zona telegrafica* (v. fig. 32 in alto) sono stati realizzati in modo da sembrare strappati a mano dal nastro;
4. la *pagina dei controlli* (v. fig. 32) è stata stampata su un foglio di carta (la prima CEP era dotata di due telescriventi); il foglio di carta è stato realizzato in modo da sembrare estratto dalla telescrivente dopo essere stato strappato manualmente dal rullo di carta;
5. la dimensione del foglio di carta è di 75 battute, il massimo consentito dal carrello della telescrivente;

6. l'interazione dei widget in stile telegrafico è suggerita con il cambio di colore dell'inchiostro da nero a rosso (v. fig. 32 in alto), i due colori dei nastri inchiostri in dotazione alle macchine Olivetti.

### 2.5.5 Stile meccanico

Liberamente ispirato alle tecnologie del periodo dei primi calcolatori elettronici, lo stile meccanico caratterizza gran parte dei componenti interattivi della UI (v. fig. 33).

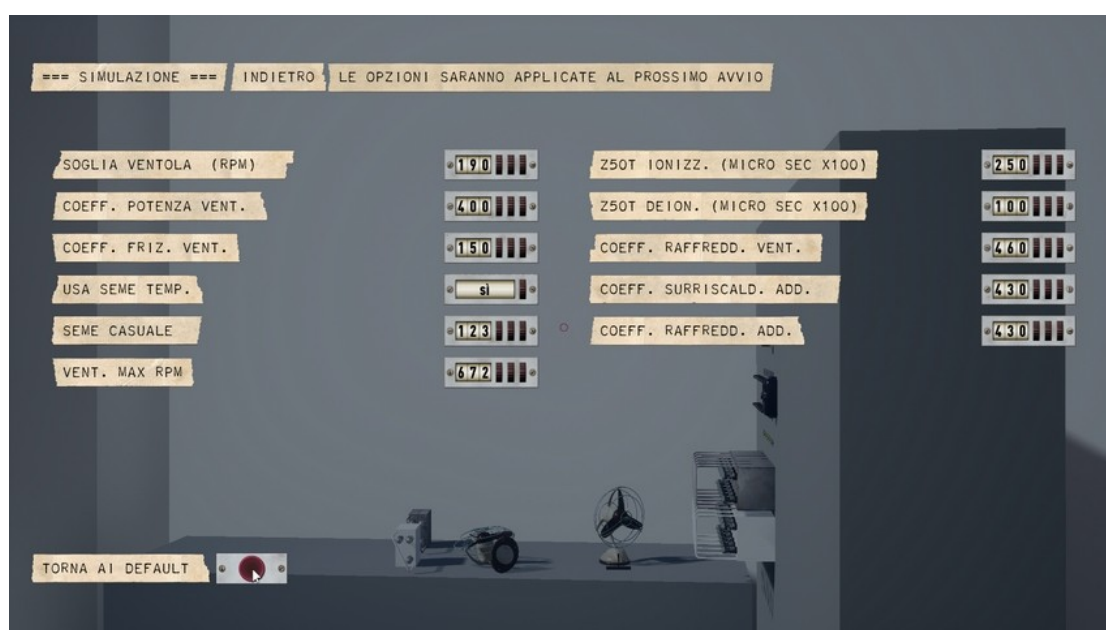


Figura 33. La pagina dei parametri di simulazione. Sono presenti contatori meccanici a rotella tripla, un selettore a rotella singola e un pulsante.

Come per lo stile telegrafico, anche questo stile mantiene la coerenza meccanica sia nell'aspetto grafico che nelle modalità d'interazione, e rimane valida l'idea che "sia stato costruito ad hoc" dai ricercatori del CSCE:

1. ogni widget meccanico è caratterizzato dalla presenza di viti, metallo e bachelite;
2. ogni widget meccanico è stato riutilizzato come modulo prestando attenzione al suo possibile funzionamento e processo di fabbricazione, caratteristica che

- ha determinato dettagli come per esempio la posizione, il numero e la dimensione delle viti;
3. il carattere utilizzato per i testi dei display meccanici è una versione del DIN 1451, il carattere adottato dall'Istituto Tedesco per la standardizzazione a partire dal 1931, ampiamente utilizzato per i display meccanici di calcolatrici e calcolatori;
  4. i componenti interattivi dei widget meccanici vengono segnalati con l'utilizzo della bachelite, che li distingue dalla parte inerte in metallo;
  5. l'interazione viene suggerita con un effetto luminoso di colore *HMR Deep Pink 57*, che si attiva quando il cursore passa sopra un componente interattivo di un widget meccanico (v. fig. 33).

## **2.6 Progettazione del sistema dei menù**

Per gestire i momenti del gioco, esterni e interni alla sessione, è stato necessario definire un “albero dei menù” che delineasse le relazioni e le dipendenze tra menù e tra le pagine interne che li compongono. Le pagine sono organizzate su livelli e ogni pagina raggruppa opzioni, parametri e contenuti affini.

Nei paragrafi seguenti si presentano la struttura del gioco e l'albero dei menù definito con l'utilizzo della notazione UML.

### **2.6.1 L'architettura dell'applicazione**

Il gioco prevede l'utilizzo di due mappe distinte: una per il menù principale e una per la scena 3D.

All'avvio il motore di gioco procede con le inizializzazioni caricando le impostazioni di gioco da un apposito file di configurazione e applicandole. A seguire carica la prima mappa, ovvero quella del menù principale, e lo lancia a tutto schermo. Dal menù principale è possibile uscire dal gioco oppure dare inizio a una nuova sessione caricando la mappa della scena 3D.

Al caricamento della mappa dell'addizionatore, vengono creati e posizionati tutti gli oggetti che compongono il mondo di gioco e vengono svolte le inizializzazioni

relative al funzionamento degli attori e dell'interazione. Contemporaneamente inizia anche la sessione.

Dalla sessione è possibile accedere ai menù in sessione (§2.3.2) e al menù di pausa, dal quale è possibile tornare al menù principale.

## 2.6.2 I diagrammi UML

Il sistema di menù segue la struttura descritta con i diagrammi UML (*Unified Modeling Language*) esposti in questo paragrafo.

La struttura è stata definita partendo dalla parte più esterna del gioco (il menù principale), fino ad arrivare alla parte più interna (l'interazione in sessione). Per convenzione è stata presa in esame la versione inglese del gioco.

Gli stati iniziali sono contrassegnati da un pallino nero, gli stati finali da un pallino nero cerchiato. Ogni altro stato è rappresentato da un rettangolo che ne riporta il nome e il contenuto. Alcuni stati sono compressi per motivi di spazio, come segnalato dalla presenza dei tre puntini, e il loro contenuto viene rappresentato per esteso in alcune apposite tavole.

Nella Tavola 1 il giocatore entra per la prima volta nel gioco. Il motore inizializza le impostazioni e mostra il menù principale, dal quale si può accedere alla sessione o abbandonare il gioco. Nel caso in cui acceda alla sessione, il motore provvede a caricare la nuova mappa e inizializzare la scena. Sulle frecce sono elencate tutte le chiamate di funzione eseguite negli eventi di transizione. Dalla sessione il giocatore può scegliere se accedere al menù di pausa per tornare al menù principale o riprendere le attività nella sessione.

La Tavola 2 è dedicata alla rappresentazione estesa del menù principale (nei diagrammi *game menu*). La tavola descrive la struttura a livelli, mostrando ogni pagina come un sotto-stato dello stato "menù principale". Ogni elemento presente nelle pagine dei menù è identificato dalle stesse stringhe utilizzate nel gioco (nella versione inglese) e il tipo di widget utilizzato. Le frecce rappresentano le possibili transizioni e i tasti per effettuarle. Le pagine di livello 3 (opzioni grafiche e di simulazione) e la pagina d'archivio sono rappresentate in forma compressa per motivi di spazio.

La Tavola 3 è la rappresentazione estesa delle pagine delle opzioni. La rappresentazione segue le convenzioni già introdotte. Sono presenti i cappi che mostrano il funzionamento dei tasti “Apply” e “Reset”.

La Tavola 4 è la rappresentazione estesa della pagina d’archivio. La rappresentazione segue le convenzioni già introdotte. Sono presenti i cappi che mostrano il funzionamento del selettore dei documenti e dei tasti di navigazione della galleria.

La Tavola 5 è dedicata alla rappresentazione estesa della sessione. Lo stato della sessione presenta l’elenco degli input. Sono presenti i cappi che mostrano il funzionamento di ogni tasto mappato. Ha una particolare importanza il menù di montaggio (mounting menu nei diagrammi) che è rappresentato per esteso completo di cappi che mostrano il funzionamento delle funzioni per effettuare gli spostamenti e attivare la pausa.

La Tavola 6 è dedicata alla rappresentazione estesa del menù di pausa (pause menu nei diagrammi). Le uniche differenze con il menù principale sono nel primo livello, nel quale sono presenti i tasti per riprendere o uscire dalla sessione al posto di quelli per entrare o uscire dal gioco.

HMR 6BIT ADDER (game menu)

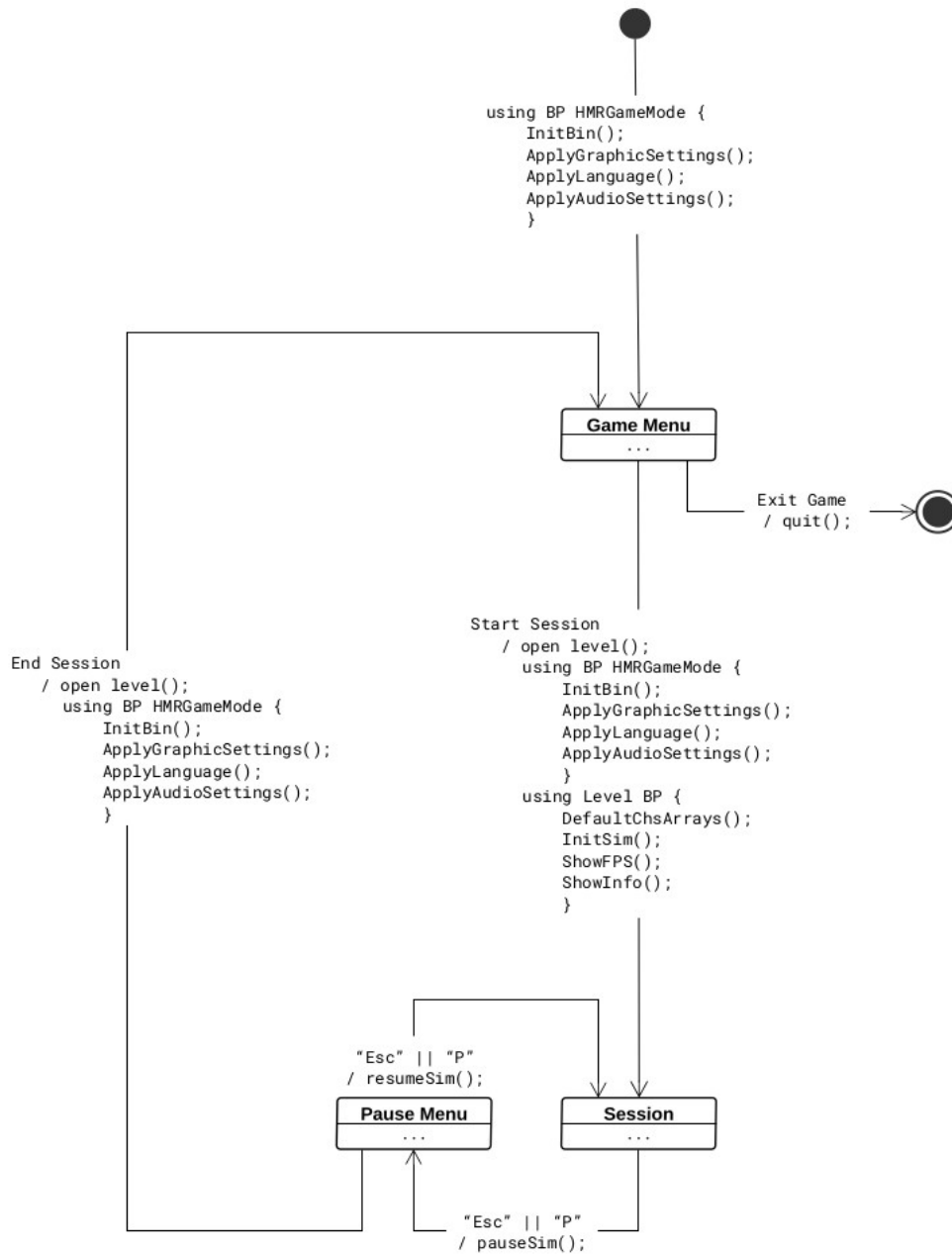


Tavola 1. Vista d'insieme dell'architettura di gioco dall'avvio all'uscita.

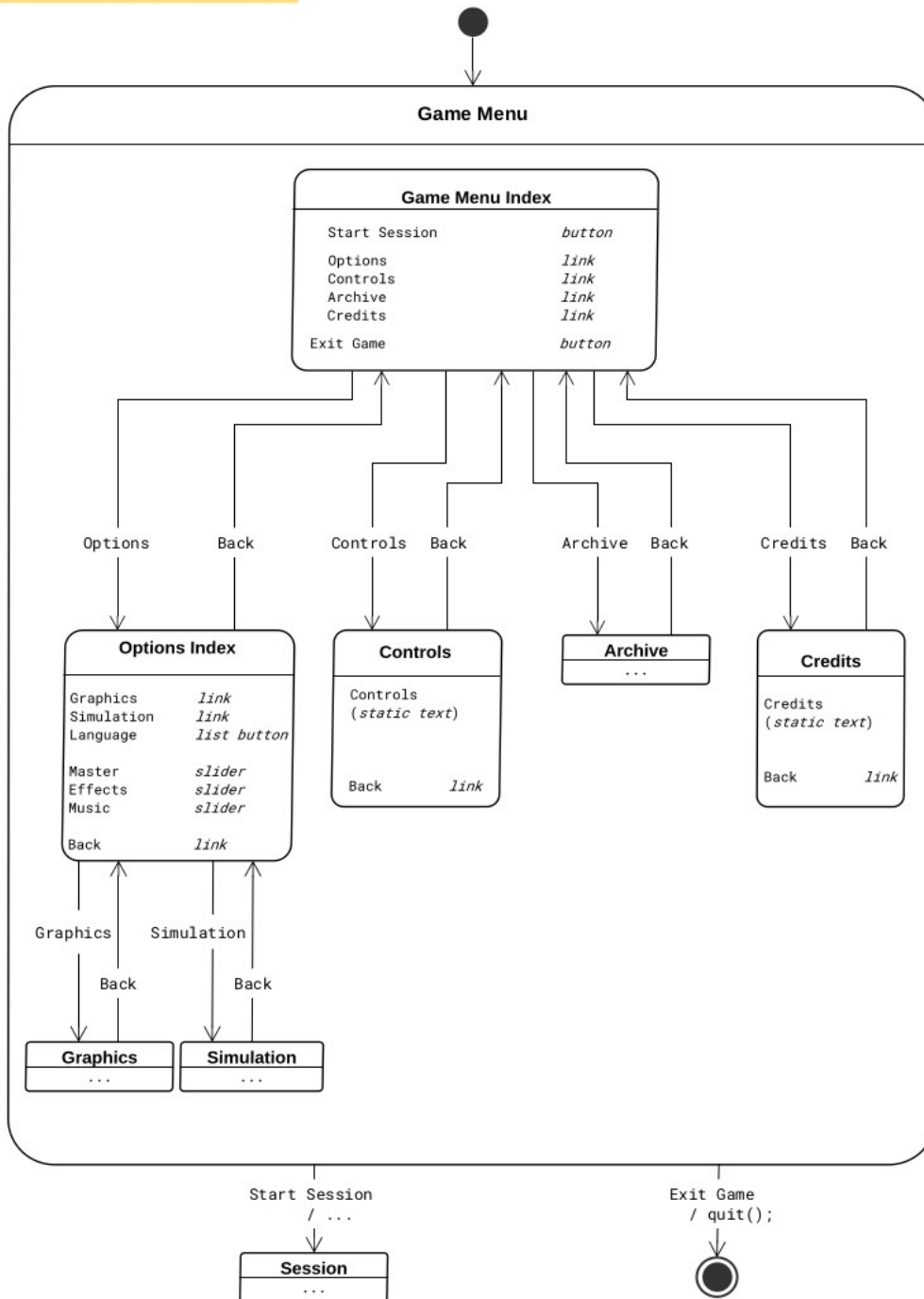


Tavola 2. La struttura del menù principale.

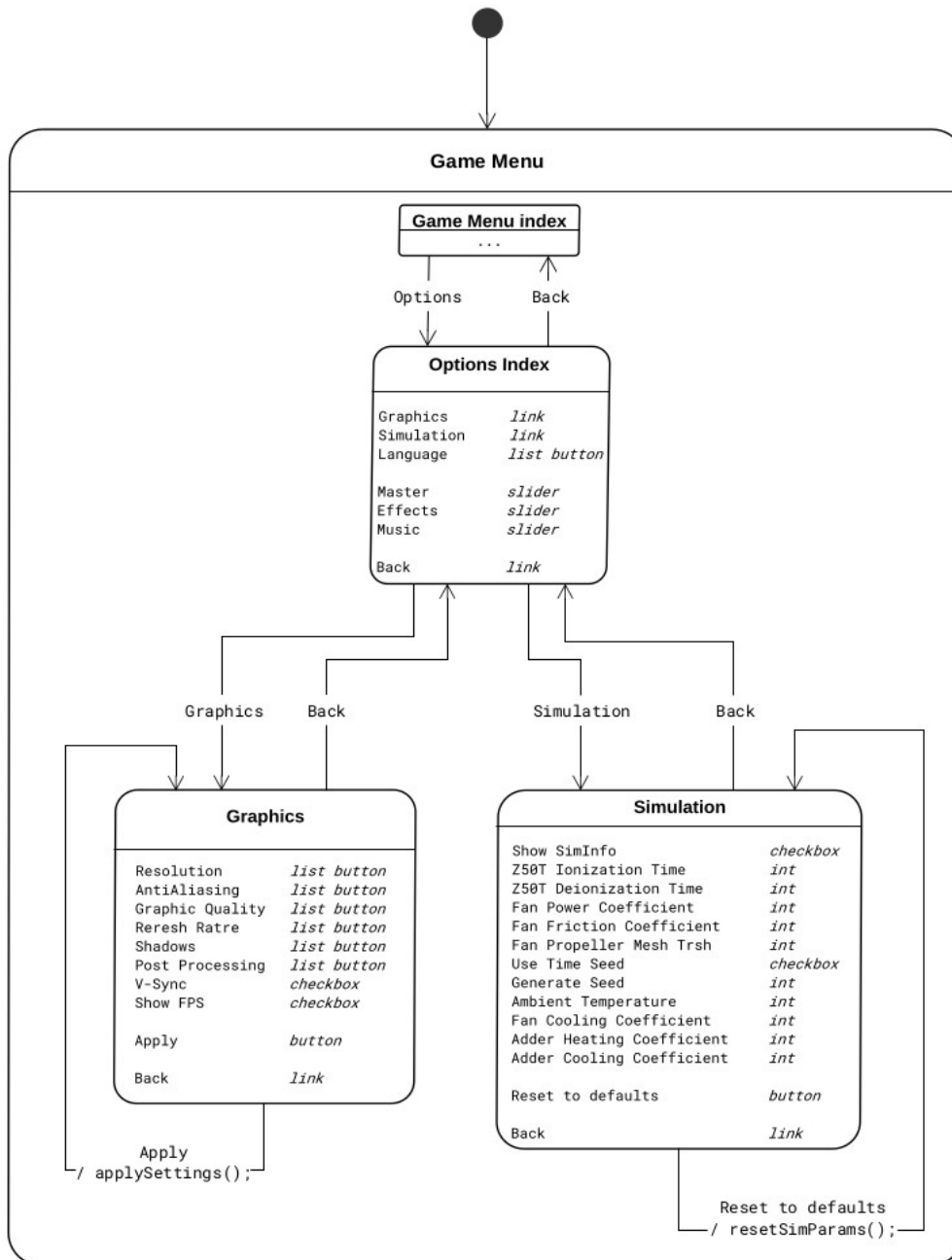


Tavola 3. Rappresentazione estesa della pagina delle opzioni del menù principale.



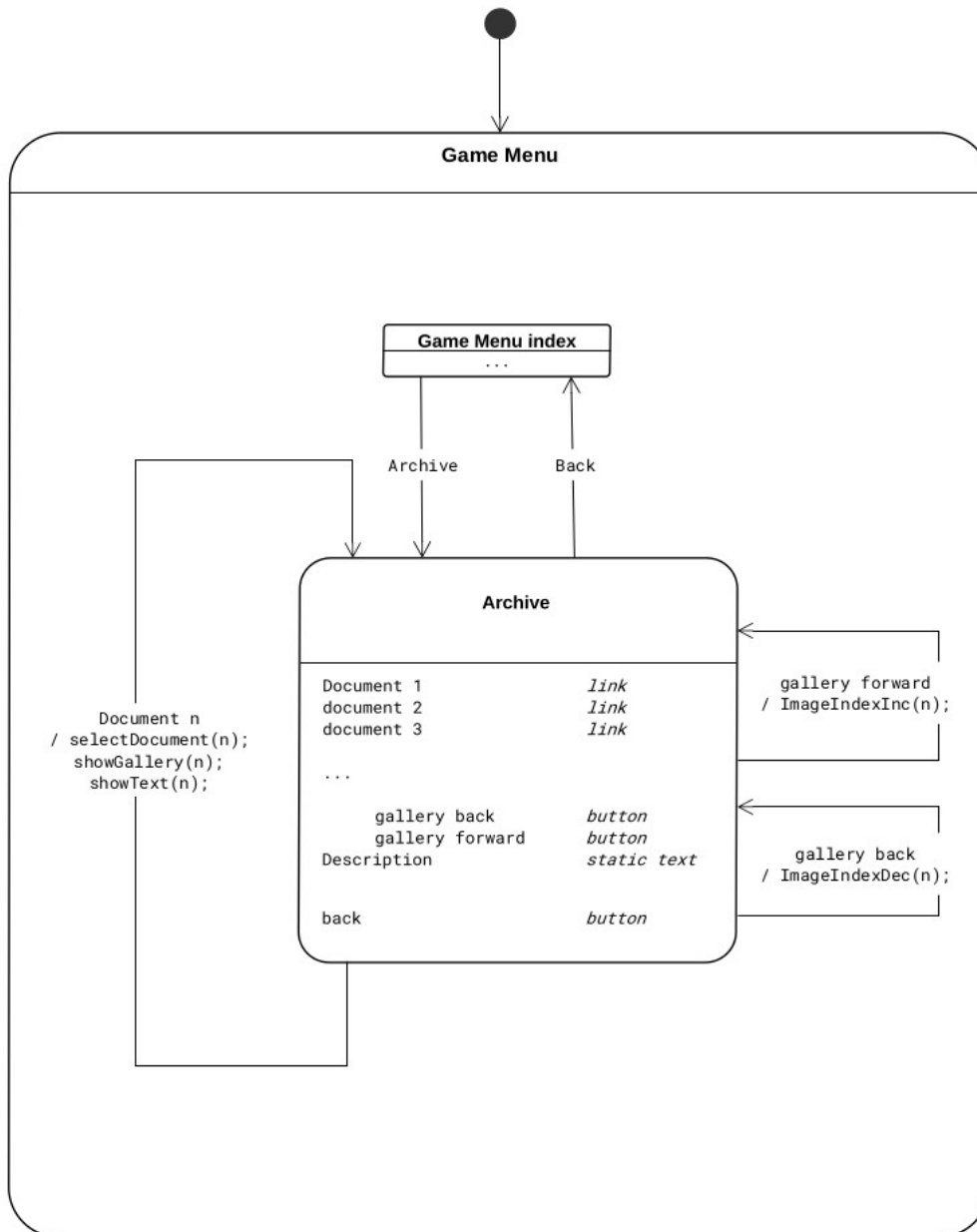


Tavola 4. La pagina dell'archivio del menù principale.

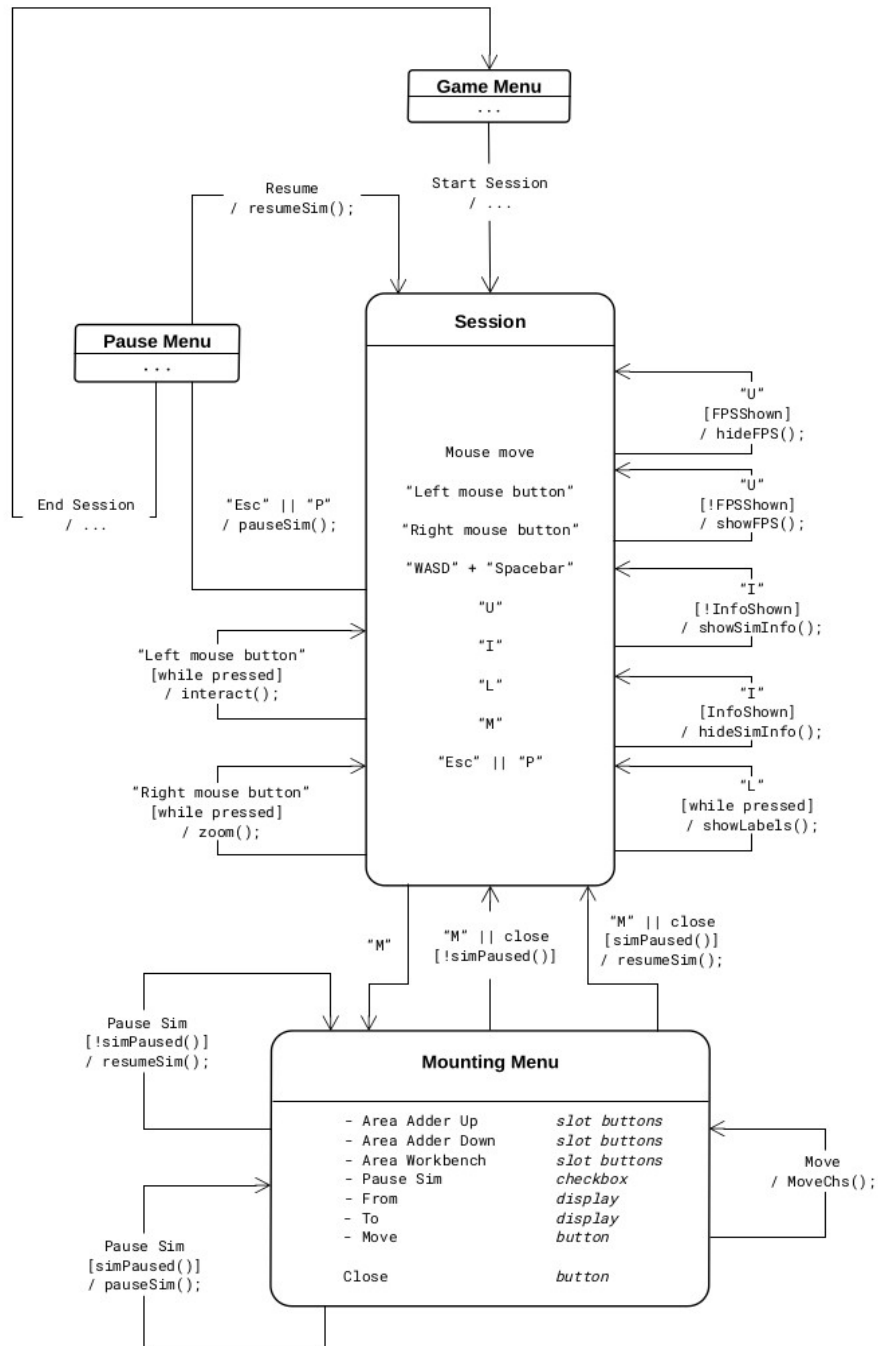


Tavola 5. Rappresentazione estesa della sessione.

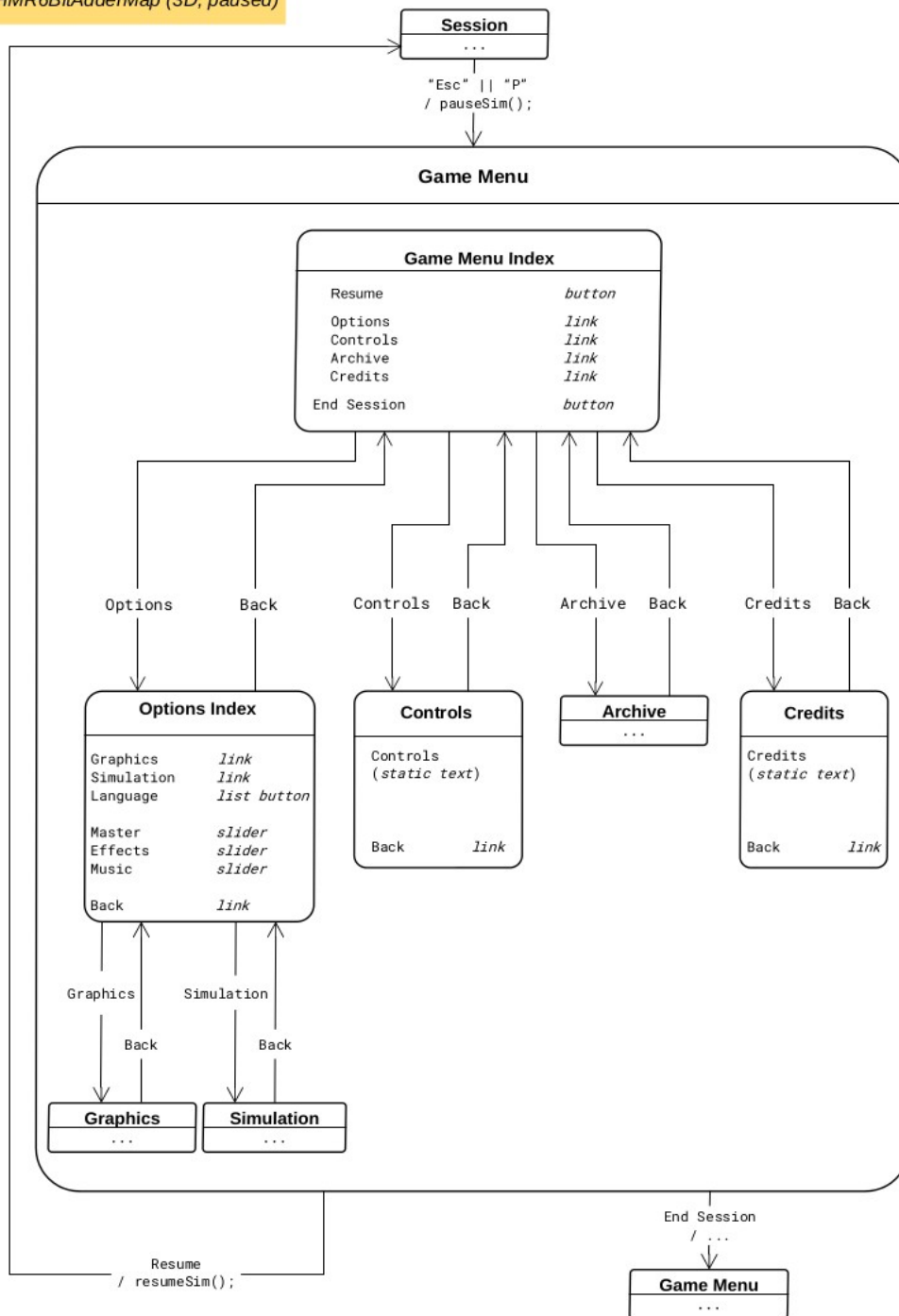


Tavola 6. La struttura del menù di pausa.

## 3. Come lo feci

*Libri e documenti... ma certo! È la biblioteca privata di mio nonno, lo sento! Guardate, guardate questo: “Come lo feci” di Viktor Frankenstein.*

Frederick Frankenstein / Gene Wilder, *Young Frankenstein*, 1974.

Questo capitolo descrive la realizzazione dei componenti del gioco oggetto della tesi. È descritto l'ambiente di sviluppo e per completezza si fa cenno anche a quanto realizzato durante il tirocinio per ottenere il prototipo su cui costruire e verificare il sistema di menù. L'esposizione presenta l'insieme di componenti realizzati e le soluzioni adottate per alcuni problemi specifici quali traduzioni e gestione delle impostazioni.

### 3.1 Lo sviluppo in Unreal, l'esperienza del prototipo

Parte del tirocinio è stata la scelta del motore di gioco. Sono stati presi in considerazione due dei motori più diffusi: *Unreal Engine* (Unreal Engine, sito web) e *Unity* (Unity, sito web). Entrambi offrono una licenza gratuita e possibilità di sviluppo multiplatforma, caratteristiche fondamentali per HMR, che prevede di portare il gioco anche sui dispositivi VR. La scelta è stata determinata dal linguaggio supportato nativamente: Unreal usa il C++, lo stesso linguaggio in cui è scritto il simulatore

dell'addizionatore, caratteristica che ha facilitato l'integrazione e garantito maggiore efficienza. Altre caratteristiche di Unreal quali gli strumenti per gestire il codice, gli elementi della scena e la creazione di interfacce, hanno rafforzato la decisione.

### 3.1.1 Blueprint e User Interface Designer

*Blueprint Visual Scripting* (BP) e *Unreal Motion Graphics User Interface Designer* (UMG, è l'acronimo utilizzato nel gergo Unreal) sono due componenti caratterizzanti l'ambiente di sviluppo di Unreal Engine.

BP è un linguaggio di programmazione grafico che permette di realizzare parti del gioco accedendo a ogni funzionalità del motore scrivendo in un linguaggio più semplice del C++ e senza dover passare da un ambiente di sviluppo C++ esterno, come ad esempio *Visual Studio* (Visual Studio, sito web). Il linguaggio BP adotta il paradigma *object oriented* ed è possibile realizzare classi BP e usarle per istanziare oggetti complessi assemblando più componenti, anche eterogenei: modelli 3D, file audio, fonti di luce, ecc.

I BP sono compilati in C++ ed è diretta l'integrazione con altro codice C++. Data la doppia compilazione il codice BP è tendenzialmente meno efficiente di codice scritto direttamente in C++. Di conseguenza, i BP sono generalmente utilizzati per la comunicazione tra componenti e per le funzionalità di componenti non critici per le prestazioni. Nel caso della replica virtuale il simulatore dell'elettronica e del modello termodinamico, computazionalmente pesante perché deve aggiornare continuamente lo stato dell'addizionatore ogni 10 $\mu$ s, è stato realizzato esternamente in C++; invece la gestione dei menù, computazionalmente semplice e soggetta ai tempi dell'interazione con l'utente, è stata tranquillamente realizzata in BP.

UMG è lo strumento per realizzare le interfacce utente 2D. L'editor consente di definire il design grafico di ogni elemento, creare widget, animazioni e integrare codice BP per gestire gli eventi d'interazione e la comunicazione tra i widget e le altre classi BP definite nel gioco.

### 3.1.2 L'integrazione del simulatore: tra C++ e BP

Il simulatore calcola in tempo reale lo stato dell'addizionatore sia come uscite dipendenti dagli ingressi e dallo stato corrente, sia come temperatura e danneggiamento dei componenti. Rispetto al diagramma in fig. 34, le fasi 1, 2 e 5 sono in carico al motore di gioco che, principalmente, cura la visualizzazione della scena 3D; il simulatore è invece in carico delle fasi 3 e 4: a ogni tick del gioco calcola il tempo effettivamente trascorso e, frazionandolo negli intervalli di integrazione (in senso matematico) di  $10\mu\text{s}$ , calcola il nuovo stato dell'addizionatore. In pratica, assumendo un frame rate di gioco di 60fps, il  $\Delta t$  mediamente è di circa 16.6ms e, a ogni tick, il simulatore esegue circa 1660 cicli di integrazione.

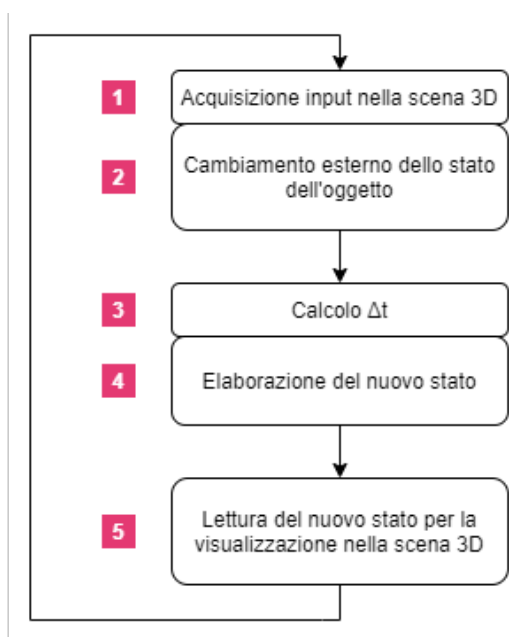


Figura 34. Il modello logico che descrive il ciclo di funzionamento del simulatore.

Il simulatore è uno sviluppo esterno ed è stato acquisito come codice C++ integrabile nel gioco e come applicazione indipendente con interfaccia a linea di comando usata per prove e verifiche.

Anche la comunicazione tra menù, scena e attori coinvolge il simulatore, dal quale passano quasi tutte le interazioni. Per esempio, l'interazione nella scena con

l'interruttore del ventilatore notifica al simulatore il cambio di stato, nei tick successivi la scena chiede al simulatore la posizione delle pale del ventilatore ne riproduce movimento.

Per integrare il codice del simulatore è stato necessario realizzare due *Blueprint Function Library* (BPL), cioè collezioni di funzioni statiche richiamabili dai BP. Le BPL definite sono:

*Proxy to Sim*, che espone l'API C++ del simulatore ai BP; perlopiù si tratta di funzioni *get* e *set* degli stati delle parti dell'addizionatore simulate, sono definite nella BPL come funzioni *blueprint callable*; questa BPL è utilizzata anche per condividere informazioni tra i diversi contesti del gioco, per esempio nel caso del menù di montaggio, tra il menù, dove si scelgono i telaietti da spostare, il simulatore, dove davvero sono spostati, e la scena 3D, nella quale vengono visualizzati spostati.

*General Utils*, che serve per la gestione delle opzioni di gioco; le funzioni di questa libreria tipicamente gestiscono le opzioni fornendo ai widget dei menù le stringhe dei valori selezionabili e al motore i comandi per applicarle e salvarle; in alcuni casi semplificano e personalizzano l'accesso alle API di Unreal.

### **3.1.3 Modelli 3D, texture, materiali, suoni ed effetti**

Per verificare e validare le funzionalità del gioco è stato necessario realizzare un prototipo della scena e degli attori principali, attività che è stata oggetto del tirocinio. Tuttavia durante la tesi sono stati raffinati diversi aspetti. Nel seguito sono descritte le aree di lavoro interessate.

#### **3.1.3.1 Modellazione 3D**

I solidi che rappresentano gli oggetti della scena (in gergo tecnico *mesh*) sono stati modellati in Blender a partire da disegni originali e fotografie; dato che nei videogiochi il motore grafico deve rendere (v. glossario) la scena in tempo reale, per ottimizzare le prestazioni è importante semplificare la geometria degli oggetti riducendo al minimo indispensabile il numero dei vertici e delle facce e affidando i dettagli a texture e materiali. Per poter essere utilizzati in Unreal, i mesh devono essere esportati in formato FBX. Tranne alcuni attori realizzati al di fuori del tirocinio

e della tesi (ventilatore, valvola, interruttore generale e chiave telefonica), i mesh della versione attuale sono poco più che segnaposto.

### **3.1.3.2 Texture e materiali**

Il motore di gioco proietta sulle superfici dei solidi immagini utili a disegnare dettagli quali colore, trasparenza, riflettività, piccole variazioni di altezza della superficie. La mappatura UV è necessaria per riportare le coordinate bidimensionali di un'immagine (in gergo tecnico *texture*) sulle coordinate tridimensionali delle facce dei mesh. Le texture sono disegnate in Gimp, in Blender sono assegnate alle parti del modello tramite i materiali, che permettono di assemblare più texture e modificarne i parametri. I materiali usati in Blender durante la modellazione sono importati in Unreal per realizzare i materiali definitivi verificando nella scena come vengono effettivamente resi dal motore grafico. Unreal permette di realizzare anche *material instance* utilizzabili per creare variazioni di uno stesso materiale padre in modo pratico ed efficiente. Tranne alcuni attori realizzati al di fuori del tirocinio e della tesi (ventilatore, valvola, interruttore generale, chiave telefonica), le texture e i materiali della versione attuale sono giusto abbozzati.

### **3.1.3.3 Suoni**

Ogni attore è provvisto di almeno un suono, il codice BP gestisce l'attivazione, la localizzazione spaziale e i valori di attenuazione. L'audio del gioco è organizzato in classi (generale, effetti speciali e musica) in modo da gestire livelli di volume indipendenti. Non ci sono state attività di generazione e montaggio audio, pertanto i suoni utilizzati sono giusto abbozzati.

### **3.1.3.4 Effetti**

Nei videogiochi effetti speciali quali fumo, fiamme e scintille sono resi con sistemi di particelle. Gli attori dei telaietti hanno effetti che si attivano in base ai danni da surriscaldamento. In Unreal gli effetti sono componenti di tipo *emitter*, definiti come *Niagara particle system*, che utilizzano immagini (in gergo tecnico *sprite*) realizzate in Gimp. Il codice BP gestisce l'attivazione e i parametri degli emitter, coordinando i suoni associati. Gli effetti sono stati realizzati al di fuori del tirocinio e della tesi e sono poco più che abbozzati.



### 3.1.4 Altri problemi affrontati e risolti

Per la realizzazione del prototipo, in gran parte durante il tirocinio, sono stati affrontati i seguenti problemi:

1. *interazione nella scena 3D*: nel tirocinio sono stati definiti i metodi per il suggerimento, l'attuazione e la conferma dell'interazione, mettendo a punto il sistema di controllo sulle collisioni;
2. *definizione dei metodi di collisione*: il metodo principale consiste in un controllo sui canali di collisione mediante l'utilizzo del line tracing; ogni attore è provvisto di una o più hitbox associate al canale *PhysicActor*, che vengono rilevate dal *line tracer* emesso dal giocatore ad ogni tick e corrispondente al centro della sua visuale;
3. *gestione delle inizializzazioni all'avvio*: durante il tirocinio è stato messo a punto il codice BP per l'inizializzazione delle opzioni all'avvio sia del gioco che di ogni sessione, quando è inizializzata la scena, è lanciato il simulatore e sono creati e indicizzati gli oggetti attore.

## 3.2 Il lavoro sui menù: widget e funzionalità

Una delle caratteristiche principali dei menù è la progettazione modulare delle pagine che li compongono, investimento che ha poi reso più semplice sia il lavoro sulla veste grafica che quello sull'integrazione delle funzionalità.

I paragrafi che seguono presentano l'architettura generale dei menù e approfondiscono il lavoro svolto sui widget dei menù dal punto di vista della grafica e della programmazione. Sono descritte anche le soluzioni adottate per gestire il supporto multilingua e le impostazioni del gioco.

### 3.2.1 Architettura object oriented

Ogni pagina di menù raggruppa contenuti affini ed è costituita dai seguenti tipi di classi:

1. *widget layout di pagina* (v. fig. 35 in alto): sono widget a tutto schermo che definiscono la struttura di una categoria di pagine; definiscono la posizione e gli spazi tra i box nei quali possono essere inseriti i widget elemento;
2. *widget elemento* (v. fig. 35 numero 2): popolano i widget layout; sono composti da uno o più widget componente e realizzano l'interazione con le voci e le opzioni dei menù;
3. *widget componente* (v. fig. 35 numero 1): sono i moduli che compongono i widget elemento; sono principalmente utilizzati per le parti comuni.

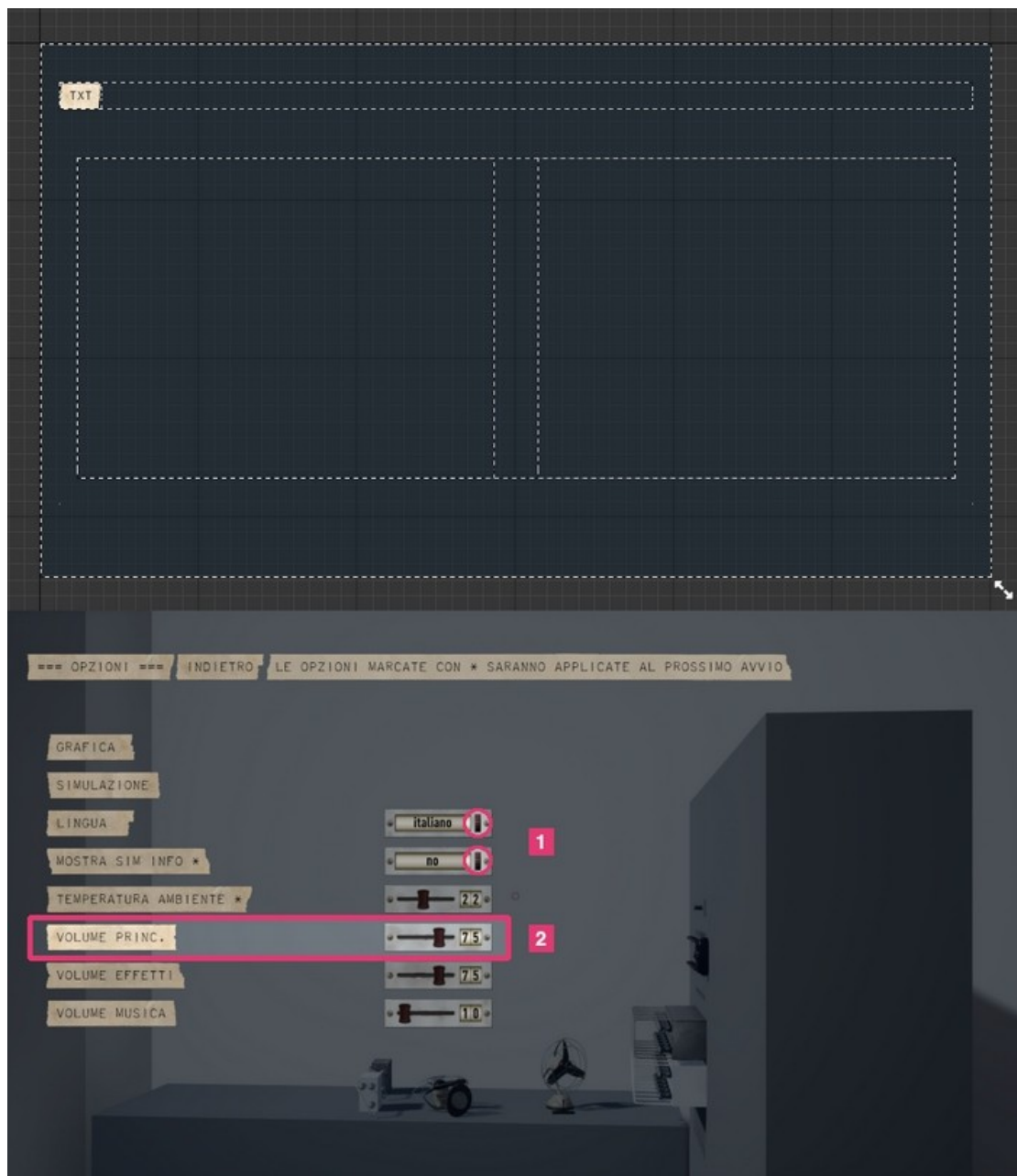


Figura 35. Il layout di una pagina di menù (in alto) e la pagina di menù costruita su di esso (in basso). Con 1 sono evidenziate due istanze di un widget componente (*rotella di bachelite*), mentre con 2 è evidenziato un widget elemento.

Questa soluzione basata su tre livelli di classi di widget permette di apportare modifiche in modo efficiente, consentendo di realizzare nuove pagine come istanze di

classi generali con modifiche di dettaglio. Inoltre i widget possono essere utilizzati anche per realizzare altre parti dell'interfaccia come il menù di montaggio.

Altre caratteristiche del sistema di menù sono il supporto multilingua aperto (per ora sono state realizzate le versioni in italiano e inglese) e il sistema di gestione di inizializzazioni e salvataggio delle opzioni di gioco e di simulazione.

### 3.2.2 Variazioni di bitmap

L'elemento tecnico che caratterizza l'aspetto dei widget sono le *bitmap*, ovvero le immagini che sono utilizzate per connotarli graficamente. È stato quindi necessario realizzare un certo numero di variazioni per ogni bitmap in modo da rendere i menù più "realistici" e non ripetitivi. Quest'ultima caratteristica importante per mantenere la coerenza dello stile il cui scopo è dare l'idea di analogico, usato, e di "fatto a mano": viti avvitate con orientamento diverso, macchie di sporco, ruggine e graffi sulle piastrine metalliche, strappi e lunghezze diverse per le zone telegrafiche e altre imperfezioni.

Per rendere più efficiente il lavoro in Gimp, per ogni elemento grafico è stato adottato un metodo di lavoro basato su livelli e immagini modulari in modo da realizzare molte variazioni rapidamente e da un numero limitato di file sorgente.

### 3.2.3 Widget modulari

Nell'ottica di utilizzare il minor numero di bitmap e fattorizzare il codice BP, le classi di widget che compongono l'interfaccia sono state pensate per essere costituite da elementi assemblabili e riutilizzabili in più contesti. I widget elemento, per esempio, sono generalmente composti da uno o più widget componente (v. fig. 36).



Figura 36. Widget di un'opzione grafica. È composto dai widget componente *zona telegrafica* (a sinistra) e *selettore a rotella singola* (a destra).

Ogni widget è organizzato su uno o più livelli di sovrapposizione: il più basso è quello di sfondo (bitmap), al quale possono essere sovrapposti un livello per i componenti sia modulari (widget componente) che non e altri eventuali livelli superiori.

L'esempio più interessante di modularità e riutilizzo di componenti si ha se si mettono a confronto un *selettore a rotella singola* e un *contatore a tripla rotella* (v. fig. 37), nei quali il widget componente della *rotella di bachelite* viene riutilizzato più volte a seconda delle necessità.



Figura 37. I widget componente *selettore a rotella singola* (a sinistra) e *contatore a tripla rotella* (a destra).

In questo modo per realizzare un eventuale *selettore a cinque rotelle* sarebbe sufficiente riutilizzare cinque volte la *rotella di bachelite* e realizzare una nuova bitmap per la piastra metallica (e le sue variazioni).

### 3.2.4 Widget telegrafici: realizzazione in Unreal

I widget in stile telegrafico sono i widget che, dal punto di vista grafico, si rifanno al mondo delle telescriventi. Possono essere utilizzati da soli oppure come componente nei widget elemento in combinazione con widget meccanici:

*Zona telegrafica* (v. fig. 38): è un widget componente su cui vengono stampate le stringhe descrittive; il codice BP lo predispone per ricevere una bitmap e un testo sia dinamicamente che staticamente; il widget è composto da un livello per lo sfondo al quale viene sovrapposto quello della stringa.



Figura 38. Il widget *zona telegrafica* così come appare nei menù (dettaglio).

*Bottone link*: è un widget elemento composto da un widget zona e un widget bottone; è graficamente identico a una zona telegrafica, ma suggerisce l'interazione cambiando il colore del testo da nero a rosso al passaggio del cursore del mouse (v. fig. 39).



Figura 39. Il widget *bottone link* così come appare nei menù (dettaglio). In questo caso il bottone sta suggerendo l'interazione.

*Foglio di telescrivente*: è il widget utilizzato nei menù per le pagine dei controlli e dei crediti; ha l'aspetto di un foglio strappato dal rotolo di carta di una telescrivente; come nella realtà, la larghezza di questo widget corrisponde a 75 battute (v. fig. 40); il widget è composto da un livello per la bitmap di sfondo al quale viene sovrapposto un livello per il testo.

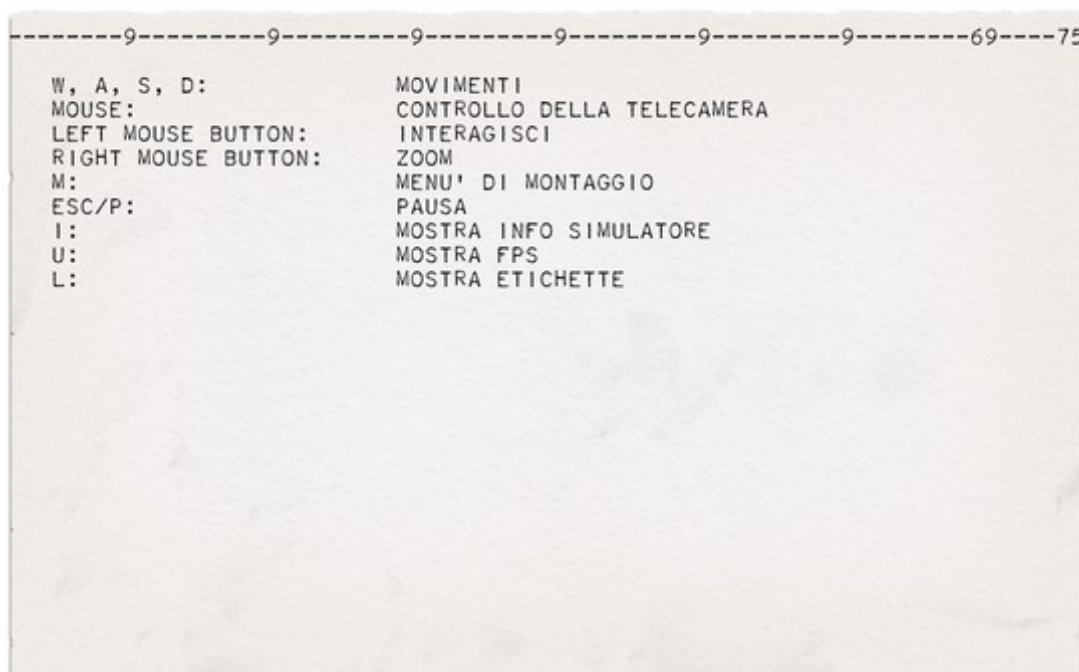


Figura 40. Il widget *foglio di telescrivente* come appare nei menù (dettaglio).

### 3.2.5 Widget telegrafici: realizzazione in Gimp

Le bitmap delle zone telegrafiche sono state create in Gimp. Ogni immagine è composta dai seguenti livelli:

1. *tape*: il nastro telegrafico: la sua lunghezza è determinata dalla lunghezza della stringa che deve ospitare e determina la dimensione dell'intera immagine;
2. *rips*: gli strappi alle estremità: sono sovrapposti al livello *tape* per simulare lo sfilacciamento della carta strappata manualmente;
3. *borders*: i bordi: sono sovrapposti al livello *tape* per simulare i contorni fisici del nastro telegrafico;
4. *shadows*: l'ombreggiatura: è aggiunta sotto il livello *tape* per simulare la tridimensionalità della zona telegrafica, teoricamente incollata sullo sfondo delle pagine dei menù;



Figura 41. Il solo livello *tape* (a sinistra) e il risultato della sovrapposizione di tutti i livelli (a destra).

Il livello *tape* è stato realizzato riutilizzando alcuni moduli preparati a partire da alcune scansioni di telegrammi:

1. *modulo inner*: parte centrale di un nastro telegrafico;
2. *modulo end*: estremità di un nastro telegrafico.

In questo modo per creare zone di lunghezze diverse si utilizza un numero di moduli *inner* a seconda della lunghezza necessaria.

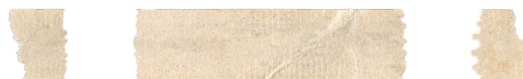


Figura 42. I moduli utilizzati per realizzare i livelli *tape* delle bitmap.

Per evitare che le bitmap subissero deformazioni troppo evidenti a seconda della lunghezza del testo che avrebbero dovuto ospitare, è stata utilizzata un foglio di calcolo (v. tabella 1) per ottenere le dimensioni ottimali delle bitmap in base al numero di caratteri: nella prima colonna è indicato il numero di caratteri della stringa da stampare sulla zona telegrafica, nella seconda colonna è indicato il valore del coefficiente calcolato con la formula  $(colonna1-1)/3$  con approssimazione per difetto, nella terza colonna è indicata la grandezza ottimale della bitmap in pixel calcolata con la formula  $(colonna2+1)*250$ , dove 250 è la risoluzione ottimale per una bitmap che deve contenere da 1 a 3 caratteri. Come è possibile notare dalla tabella, ogni tre caratteri di testo corrispondono a 250px nel file sorgente a dimensione naturale.

Numero Caratteri	INT/3	Bitmap
2	0	250
3	0	250
4	1	500
5	1	500
6	1	500
7	2	750
8	2	750
9	2	750
10	3	1000
...	...	...
60	19	5000

Tabella 1. Foglio di calcolo per ottenere la lunghezza ottimale delle bitmap delle zone telegrafiche nel file sorgente.

Il processo di realizzazione delle bitmap del *foglio di telescrivente* risulta pressoché identico a quello delle zone telegrafiche, ma presenta due livelli in più:

1. *vignette*: effetto vignetta; livello sovrapposto a quello della texture della carta che serve a dare l'idea di consumato e simulare l'occlusione ambientale;



2. *creases*: le pieghe; livello sovrapposto a tutti gli altri che aggiunge imperfezioni per dare l'idea che la carta sia stata maneggiata;

Bordi, strappi e ombre sono stati realizzati nello stesso modo delle zone telegrafiche (v. fig. 43).

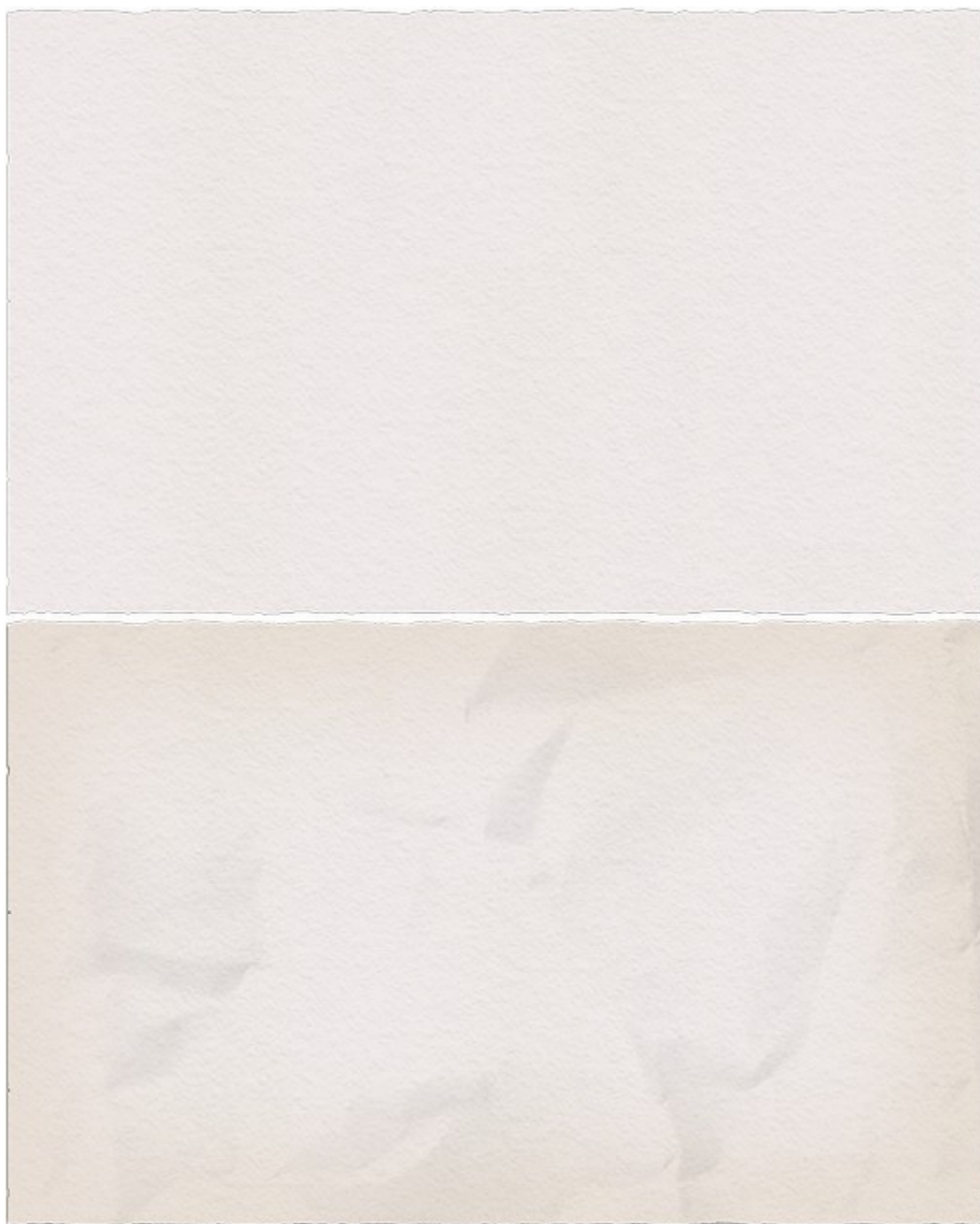


Figura 43. Il livello base del foglio di carta (in alto) e il risultato della sovrapposizione di tutti i livelli (in basso).

### 3.2.6 Widget meccanici: realizzazione in Unreal

I widget in stile meccanico sono liberamente ispirati alle tecnologie dell'informatica degli anni '50 sia nelle forme che nei materiali e nei metodi d'interazione. Possono essere utilizzati da soli oppure come componente nei widget elemento in combinazione con widget telegrafici:

*Pulsante meccanico* (v. fig. 44): widget componente utilizzato per i pulsanti presenti nella UI; è composto da una piastra metallica (bitmap di sfondo) e un widget *pulsante di bachelite*, che costituisce la componente interattiva del widget.

*Pulsante di bachelite* (v. fig. 44): widget componente utilizzato per l'elemento stesso del pulsante nei widget meccanici; è provvisto di due bitmap rispettivamente per la posizione schiacciata e quella alzata; è presente anche una terza bitmap che viene utilizzata per il suggerimento dell'interazione al passaggio del mouse; per i pulsanti della galleria sono state realizzate due bitmap aggiuntive per "stampare" le frecce sulla bachelite.



Figura 44. Il widget *pulsante meccanico* così come appare nei menù (a sinistra), il *pulsante di bachelite* in posizione alzata (al centro) e il *pulsante di bachelite* in posizione premuta (a destra).

*Pulsante per la navigazione tra menù* (v. fig. 45): widget elemento composto da un pulsante meccanico (sinistra) e una zona telegrafica (a destra); la posizione della zona rispetto al pulsante lo rende identificabile come pulsante per la navigazione.



Figura 45. Il widget elemento *pulsante per la navigazione tra menù* così come appare nei menù.

*Pulsante per l'applicazione di azioni* (v. fig. 46): widget elemento composto da una zona telegrafica (sinistra) e un pulsante meccanico (a destra); la posizione della zona rispetto al pulsante lo rende distinguibile dal pulsante per la navigazione.



Figura 46. Il widget elemento *pulsante per l'applicazione di azioni* così come appare nei menù.

*Selettore a rotella singola* (v. fig. 47): widget componente utilizzato per la selezione di parametri; quando appare in un widget elemento insieme a una zona telegrafica, è posizionato alla sua destra; ne esiste anche una versione dimensione più piccola che viene utilizzata nella pagina delle opzioni di simulazione, perlopiù popolata da *contatori a tripla rotella* che hanno una lunghezza ridotta.

*Contatore a tripla rotella* (v. fig. 47): widget componente utilizzato per la modifica libera di parametri; ogni rotella agisce su una singola cifra; quando appare in un widget elemento insieme a una zona telegrafica, è posizionato alla sua destra.

*Rotella di bachelite* (v. fig. 47): widget componente utilizzato per le rotelle dei widget dei selettori e contatori; è composto da una bitmap di sfondo e due bottoni invisibili che definiscono le due aree d'interazione corrispondenti alla rotazione della rotella verso l'alto e verso il basso; per ogni bottone è presente una bitmap che viene utilizzata per il suggerimento dell'interazione al passaggio del mouse.



Figura 47. I widget di una *rotella di bachelite* (a sinistra), un *selettore a rotella singola* (al centro) e un *contatore a tripla rotella* (a destra).

*Display meccanico* (v. fig. 48): widget componente utilizzato per la sola lettura dei parametri; non presenta componenti interattivi.



Figura 48. Il widget *display meccanico* così come appare nei menù.

*Slider meccanico* (v. fig. 49): widget componente utilizzato per modificare parametri in un intervallo; gli estremi e il passo degli scatti dello slider sono impostati dal codice BP.



Figura 49. Il widget elemento *slider meccanico* così come appare nei menù.

I “materiali” utilizzati nelle bitmap segnalano graficamente la presenza d’interazione: gli elementi meccanici con i quali è possibile interagire sono caratterizzati dalla bachelite, caratteristica che dà al giocatore la possibilità di intuire il funzionamento degli elementi dell’interfaccia a partire dal colpo d’occhio. L’interazione viene poi suggerita dall’utilizzo del colore *HMR Deep Pink 57*, che viene utilizzato per evidenziare il contorno del componente interattivo al passaggio del cursore del mouse.

### 3.2.7 Widget meccanici: realizzazione in Gimp

Come per i widget telegrafici, anche le bitmap di questi widget sono state realizzate in Gimp e sono composte dai seguenti livelli:

1. *plate*: la piastra metallica; simula l’effetto di un metallo satinato simile all’alluminio;
2. *plate dirt*: simula lo sporco depositato dalle dita sul metallo durante l’utilizzo;
3. *plate scratches* (solo per gli slider): simula i graffi prodotti dallo sfregamento del pomello degli slider sui bordi della fessura della piastra;
4. *borders*: simula la tridimensionalità della piastra;
5. *borders dirt*: simula lo sporco e la ruggine sui bordi della piastra;

6. *screws*: le viti con le quali il widget è stato “montato” sulle pagine dei menù;
7. *screws shading*: simula la corretta illuminazione sulle viti rendendola conforme a quella degli altri livelli;
8. *screws shadow*: l’ombra che la testa delle viti proietta sulla piastra;
9. *plate shadows*: l’ombra che viene proiettata dalla piastra sullo sfondo delle pagine dei menù.



Figura 50. Il livello della piastra metallica (a sinistra) e il risultato della sovrapposizione di tutti i livelli (a destra).

Le bitmap dei widget che hanno un “display meccanico”, presentano alcuni livelli aggiuntivi:

1. *roller*: il rullo che serve per visualizzare i valori selezionati o le cifre immesse;
2. *roller color*: livello che sovrappone al rullo il colore.



Figura 51. La bitmap di sfondo di un selettore a rotella singola con e senza rullo. Da questa immagine è possibile notare che il testo che appare nei display dei widget è un componente a parte sovrapposto in Unreal.

Inoltre i widget in stile meccanico presentano alcuni componenti modulari come i widget componente *rotella di bachelite* e *pulsante di bachelite*.

Le bitmap della *rotella di bachelite* sono composte dai seguenti livelli:

1. *wheel*: livello che dà la forma alla rotella;
2. *wheel color*: livello che sovrappone alla rotella il colore della bachelite;
3. *wheel shadow*: l’ombreggiatura che la rotella proietta sulla piastra.

Le bitmap del *pulsante di bachelite* sono composte dai seguenti livelli:

1. *button (up/down)*: livello che dà la forma al pulsante;
2. *button color*: livello che sovrappone al pulsante il colore della bachelite;
3. *button glare*: livello che simula l'illuminazione sul pulsante;
4. *button shading*: livello che simula la tridimensionalità del pulsante;
5. *button shadow (up/down)*: l'ombreggiatura del pulsante; è più pronunciata a seconda che il pulsante sia schiacciato oppure no.

Le bitmap dei pulsanti della galleria dell'archivio (v. §3.3) prevedono due livelli aggiuntivi:

1. *button arrow (back/forward)*: la freccia stampata sul pulsante;
2. *button arrow dirt*: livello che aggiunge lo sporco depositato sul pulsante dalle dita durante l'utilizzo.



Figura 52. I vari livelli che compongono le bitmap dei *pulsanti di bachelite*.

Per i componenti interattivi sono presenti anche dei livelli per la bitmap di suggerimento dell'interazione:

1. *outline halo 1*: livello che definisce una sfumatura semitrasparente di base che segue il contorno o un lato del componente in bachelite;
2. *outline halo 2*: livello che definisce una sfumatura semitrasparente più corposa, anche questo segue il contorno o un lato del componente in bachelite;
3. *outline halo highlight*: livello che evidenzia con un colore pieno il contorno o un lato del componente in bachelite.

Il risultato finale genera un effetto che ricorda l'illuminazione di un neon (v. fig. 53).



Figura 53. L'effetto dell'*outline* sul *pulsante di bachelite*, la *rotella di bachelite* e il cursore degli *slider meccanici*.

### 3.2.8 Traduzioni e localizzazione

La localizzazione è l'operazione che consiste nella traduzione e nell'adattamento di un gioco dalla lingua principale a un'altra. Oltre ai testi presenti nel gioco, il processo di adattamento può coinvolgere anche altri componenti quali immagini, tracce audio, modelli 3D e altri asset.

Per realizzare il sistema di menù oggetto della tesi è stato sufficiente tradurre e adattare i testi e le bitmap dei widget. Attualmente il gioco è disponibile in inglese e in italiano, con l'inglese come lingua principale. Questo significa che le tabelle di traduzione del gioco si basano sulla traduzione diretta dall'inglese a qualsiasi altra lingua, per ora solo l'italiano.

Per lavorare alla localizzazione, Unreal mette a disposizione la *Localization Dashboard*, lo strumento che permette di radunare tutti i testi presenti nel gioco e tradurli. I testi presenti nel sistema di menù sono memorizzati nelle variabili di tipo testo utilizzate nei BP e nelle tabelle di stringhe (in inglese *string table*).

Le string table consentono di organizzare i testi localizzabili in tabelle modo da renderli più accessibili ed efficienti. Sono particolarmente vantaggiose per i testi che vengono riutilizzati molte volte: il tasto "indietro" per esempio, invece di generare un *token* (v. glossario) per ogni sua istanza, utilizza quello generato dalla *string table*, evitando di creare inutili duplicati e rendendo le tabelle di traduzione più efficienti.

Localization dashboard offre anche la possibilità di esportare e importare le tabelle di traduzione in formato Portable Object (.PO), uno degli standard più diffusi nel campo della localizzazione, che consente di apportare modifiche alle tabelle utilizzando uno strumento esterno.

Inoltre, per ridurre al minimo il numero di bitmap necessarie per le zone telegrafiche, le parole utilizzate nelle stringhe dei menù sono state scelte cercando di far combaciare il numero di caratteri tra la versione inglese e il corrispettivo italiano. In questo modo è possibile mantenere le stesse bitmap di sfondo per entrambe le lingue.

Nel caso in cui la lunghezza delle stringhe non combaciassero perfettamente, sono stati aggiunti uno o più spazi alla fine della stringa più corta, soluzione che contribuisce anche a rendere più variegate e "imprecise" le zone telegrafiche rafforzando l'idea di



stile del casuale e del “fatto a mano”. Seguendo la tabella che definisce le dimensioni ottimali in base al numero di caratteri (v. §3.2.5), sono state calcolate la quantità di zone necessarie in base alle frequenze della lunghezza delle stringhe e le misure delle bitmap in pixel.

### **3.2.9 Gestione delle impostazioni (GameUserSettings + MoneyBin)**

Le impostazioni di gioco sono gestite da due file di configurazione: *GameUserSettings* per le opzioni grafiche, e *MoneyBin* per le opzioni di simulazione. Quando il gioco viene lanciato, le opzioni sono inizializzate in base all’ultima versione dei file di configurazione; qualora i file non esistessero (e.g. al primo avvio), vengono creati utilizzando i valori di default. A seguire Unreal inizializza il *MoneyBin* e successivamente utilizza il BP *HMRGameMode* per inizializzare il gioco con le funzioni della BPL *General Utils*.

Le funzioni della BPL Proxy to Sim agiscono direttamente sui valori dei parametri di simulazione salvati nel *MoneyBin*. Quelle della BPL *General Utils*, che leggono i valori salvati in *GameUserSettings*, funzionano in modo diverso: ogni funzione restituisce uno o più array e l’indice corrispondente al valore selezionato; l’indice viene poi passato al widget dedicato, grazie al quale è possibile selezionare un nuovo valore tra quelli disponibili, e quindi un indice diverso.

## **3.3 L’archivio e i suoi contenuti**

L’archivio presenta una veste grafica ispirata al memex di Vannevar Bush (v. §2.5.3). La pagina infatti propone una reinterpretazione dell’organizzazione “a due schermi ottimizzati”, una delle caratteristiche distintive del memex.

Le cornici metalliche che supportano gli schermi sono state realizzate a partire da moduli componibili, con i quali è possibile costruire una nutrita varietà di cornici in base alle esigenze dei menù. I moduli creati sono:

1. *angolo sinistro/destro superiore/inferiore;*
2. *giunzione centrale sinistra/destra con alloggio per il pulsante;*
3. *lato verticale sinistro/destro;*

4. lato verticale sinistro/destro con vite (inutilizzato);
5. lato orizzontale superiore/inferiore;
6. lato orizzontale superiore/inferiore con vite;
7. selettore a rotella singola integrato.

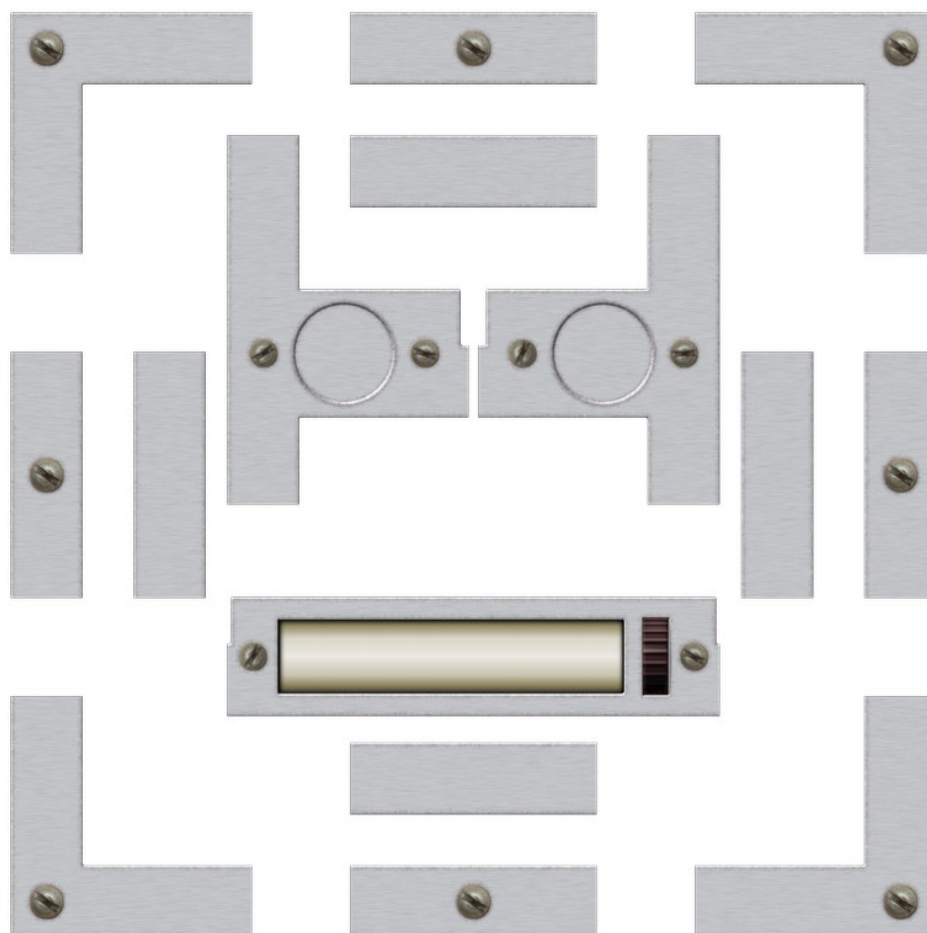


Figura 54. I moduli utilizzati per realizzare le cornici metalliche in stile memex.

Ognuno di questi moduli è a sua volta composto dagli stessi livelli utilizzati anche negli altri widget meccanici:

1. *plate*;
2. *plate dirt*;
3. *borders*;
4. *borders dirt*;

5. *screws*;
6. *screws shading*;
7. *screws shadow*;
8. *plate shadows*.

I documenti, selezionabili dallo schermo di sinistra che funge da indice, sono organizzati secondo la seguente logica:

1. i primi tre documenti contestualizzano l'oggetto: *Progetto HMR, La prima CEP, L'addizionatore a 6 bit, La replica dell'addizionatore*;
2. i successivi quattro documenti sono dedicati principali componenti elettronici e meccanici della scena: *Le chiavi telefoniche, I tubi Z50T, I tubi E88CC, Il ventilatorino*;
3. gli altri documenti forniscono approfondimenti sulle ispirazioni grafiche dei menù e il contesto storico: *La Ercole Marelli S.A., Il memex, Telescriventi, Zona telegrafica, Il carattere OliT2*.

La stesura dei testi è stata soggetta ad alcuni limiti imposti dallo spazio a disposizione sullo schermo: 60 caratteri per le didascalie delle immagini e 528 caratteri, distribuiti su 8 righe da 66 caratteri, per il paragrafo descrittivo.

A livello di codice BP, ogni documento viene costruito dinamicamente a partire da una riga di *data table*, una struttura dati che permette di creare una tabella di oggetti eterogenei in Unreal. Ogni riga presenta i seguenti campi:

1. *Row Name*: identificatore della riga, viene utilizzato per selezionare la riga corrispondente a un determinato documento;
2. *Title*: variabile di tipo testo corrispondente al titolo visualizzato nello schermo dell'indice per un determinato documento;
3. *Text*: variabile di tipo testo corrispondente al paragrafo che descrive l'oggetto a cui è dedicato il documento selezionato;
4. *Images*: array di immagini utilizzato per popolare la galleria di un determinato documento;
5. *Descriptions*: array di testi utilizzato per le didascalie delle immagini della galleria;

6. *MemexPaper*: variabile di tipo texture 2D che definisce la bitmap di foglio di carta utilizzata come sfondo sul quale “stampare il documento”.

Ognuna di queste informazioni viene utilizzata per sostituire i segnaposto di default dei widget che compongono la pagina. L’archivio viene sempre inizializzato sul primo documento e sulla prima immagine della sua galleria.

La navigazione tra documenti è resa possibile da un *selettore a rotella singola*, che permette di selezionare le varie pagine dell’indice, mentre la navigazione tra le immagini della galleria utilizza due *pulsanti di bachelite*, per i quali sono utilizzate le bitmap con i livelli aggiuntivi per le frecce (v. fig. 52).

## Conclusioni

La tesi si inserisce nel sottoprogetto MR-VR di HMR per realizzare simulatori di vecchi strumenti per il calcolo con interfacce utenti in realtà virtuale. In particolare, ha realizzato un componente della replica virtuale dell'addizionatore: il sistema di menù. La tesi prosegue l'esperienza del tirocinio durante il quale è stata realizzata l'architettura di gioco ed è stato integrato il simulatore C++. Usando il prototipo come base di lavoro, la tesi ha costruito il sistema di menu:

1. *studiando il contesto*: è stata condotta un'attività di ricerca e documentazione riguardo l'impiego dei videogiochi in ambito didattico e il rapporto che essi hanno con la storia e la sua rappresentazione; sono stati analizzati, sia in generale che rispetto ai menù obiettivo della tesi, l'attuale stato dell'arte, gli errori più comuni, le criticità e le buone pratiche da seguire nello sviluppo di un videogioco didattico;
2. *definendo uno stile grafico ispirato alla storia dell'informatica*: i menù sono graficamente ispirati alle tecnologie, alle forme e ai materiali dell'informatica degli anni '50; i menù contribuiscono all'ambientazione accogliendo, fin dal primo contatto con il gioco, l'utente nell'atmosfera degli anni della sperimentazione e realizzazione dei primi calcolatori con riferimenti sia puntuali (e.g. telescriventi) che liberamente ispirati (e.g. widget meccanici in metallo e bachelite);
3. *definendo le specifiche funzionali con diagrammi UML*: la struttura del sistema di menù è stata documentata utilizzando una delle notazioni più diffuse nell'ambito della modellazione software; sono quindi state descritte le specifiche e le funzionalità dei menù e i loro rapporti di dipendenza;
4. *realizzando il sistema di menù*: i menù gestiscono le opzioni di gioco e di simulazione, e alcuni aspetti d'interazione durante la sessione; i menù sono stati progettati in modo modulare minimizzando il numero di file sorgente necessari e rendendo più efficienti gli interventi di modifica e applicazione

degli stili grafici; tutte le funzioni definite per aggiungere funzionalità o semplificare le API di Unreal sono state documentate (vedi appendice).

## **Strumenti di sviluppo e dimensione del lavoro**

Lo sviluppo dell'applicazione ha permesso lo studio e l'approfondimento di strumenti conosciuti come Gimp e Visual Studio, ma anche di strumenti mai utilizzati quali Unreal Engine e Blender. Tra questi, gli strumenti utilizzati maggiormente nella tesi sono stati Unreal e Gimp.

Per il sistema di menù sono state realizzate 42 classi di widget e 4 blueprint generali per la gestione di input, integrazioni e inizializzazioni. A queste si aggiungono 12 classi di attori BP realizzate nel tirocinio. Per l'integrazione del simulatore sono state prodotte 1215 righe di codice C++.

In Gimp sono stati creati 23 progetti, utilizzati come sorgenti per generare le 164 bitmap utilizzate nei menù; sempre in Gimp sono state anche realizzate 26 texture e immagini provvisorie presenti nel gioco.

Il gioco è attualmente disponibile in versione demo per i PC Windows sul sito del Progetto HMR sia in versione direttamente eseguibile, sia in versione sorgente all'indirizzo [www.progettoHMR.it/MR-VR/](http://www.progettoHMR.it/MR-VR/).

## **Sviluppi futuri**

Facendo parte di un progetto in divenire, la tesi rappresenta una parte del lavoro che porterà alla realizzazione del videogioco completo. Alcune tesi si stanno già occupando dell'evoluzione dell'ambiente 3D e dell'illuminazione, altre tesi e tirocini si occuperanno di effetti speciali, suoni, del *porting* su altre piattaforme, della distribuzione su negozi digitali come, per esempio, Steam...

Non è escluso che si possono espandere e perfezionare anche i menù. Alcune idee già ipotizzate sono:

- *sostituire i bottoni link*: per quanto pratico, l'utilizzo delle zone telegrafiche per la navigazione tra pagine di menù è “moderna” e di fatto espone una certa

incoerenza meccanica; una futura revisione dello stile dei menù potrebbe estendere la metafora del memex, utilizzata nella pagina d'archivio, anche alle altre pagine;

- *aggiungere animazioni ai widget*: allo stato attuale tutte le classi di widget cambiano stato in un passo, aggiungere animazioni più fluide contribuirebbe a far diventare ancora più interessanti i menù e l'interazione con essi;
- *migliorare il suggerimento dell'interazione 2D*: il suggerimento dell'interazione avviene attraverso la comparsa di *outline* molto semplici che evidenziano il contorno dei componenti interattivi; in futuro queste outline potrebbero essere raffinate migliorando la simulazione dell'effetto di illuminazione e aggiungendo effetti di dissolvenza, oppure introducendo l'utilizzo di nuovi componenti meccanici, per esempio dei tubi Z50T;
- *realizzare un sistema di achievement* (v. glossario) *per i documenti*: l'archivio potrebbe contenere un certo numero di "documenti base", sempre presenti e accessibili, e un numero di documenti sbloccati che diventerebbero accessibili in base ad alcune azioni specifiche del giocatore in sessione (e.g. il giocatore interagisce per la prima volta con un oggetto);
- *sviluppare una modalità "ispeziona"*: attualmente il giocatore può esaminare gli oggetti utilizzando lo zoom con il tasto destro; la modalità ispeziona consentirebbe all'utente di esaminare, tramite un apposito menù, un modello ad alta risoluzione dell'oggetto desiderato.

L'impegno nello sviluppo del gioco proseguirà come tesi magistrale centrata sulla amministrazione del progetto e l'integrazione del lavoro degli altri tesisti e tirocinanti, con l'obiettivo di sperimentare se l'informatico umanista sia anche in grado di condurre ruoli di coordinamento come lo *scrum master* di progetti condotti con metodi agili (Schwaber 1997; Scrum Guides, sito web).

# Appendici

## Glossario

Questa sezione di glossario contiene termini appartenenti al gergo dello sviluppo dei videogiochi, al linguaggio tecnico di Unreal e alcune diciture specifiche del progetto CEP.

**Achievement.** In gergo videoludico con achievement (lett. conquista, compimento di un'impresa) si intende un obiettivo esterno al gioco che non ha effetti sulla partita. Gli achievement si riferiscono ai successi del giocatore in quanto videogiocatore e non personaggio del gioco, per esempio “parla con tutti i personaggi”, “trova tutti gli oggetti collezionabili”. Gli achievement sono conosciuti anche come trofei, medaglie, premi, distintivi e sfide.

**API.** *Application programming interface.* Interfaccia per accedere da programma alle funzionalità di un'applicazione. Una buona prassi di ingegneria del software è costruire le applicazioni come componenti *core* che espongono API a diverse interfacce d'uso.

**AR.** *Augmented reality,* realtà aumentata. Arricchimento di un'esperienza che si svolge nel mondo reale utilizzando dispositivi mobili, vetri di veicoli e visori per sovrapporre all'ambiente reale oggetti virtuali e informazioni.

**Asset.** Modelli, texture, animazioni, suoni, effetti, script, widget eventualmente organizzati in componenti più complessi, utilizzati dal motore di gioco per costruire il mondo di gioco.

**Bitmap.** Tipo di formato grafico in cui l'immagine è definita pixel per pixel con informazioni di colore e trasparenza. Nella tesi sono state utilizzate per realizzare i componenti 2D dell'interfaccia.

**BP.** Abbreviazione per Blueprint, inteso come linguaggio, codice e oggetto in Unreal.

**BPL.** Abbreviazione per Blueprint function Library, ovvero una libreria di funzioni utilizzabili nei blueprint.



**Controller.** Periferica di gioco solitamente dotata di tasti, levette e sensori che serve a raccogliere gli input del giocatore.

**Fps.** Fotogrammi (o *frame*) per secondo, il numero di fotogrammi al secondo visualizzati sullo schermo.

**Frame rate.** Frequenza dei fotogrammi, è la frequenza di riproduzione dei fotogrammi che appaiono sullo schermo. Viene misurata in hertz (Hz) ed espressa in termini di fps.

**Gameplay.** Termine tecnico con il quale si fa riferimento all'esperienza che il videogioco offre al giocatore (lett. "giocare il gioco").

**HUD.** *Head-up-display*, di origine militare, il termine viene utilizzato nel mondo dei videogiochi per riferirsi alla parte di UI sempre presente in sovrimpressione durante il gioco.

**Mesh.** Insieme di poligoni che definiscono un oggetto nello spazio 3D.

**Rendere.** In inglese *render*, il *rendering* è il processo di creazione di un'immagine in computer grafica.

**Run 'n' gun.** Sottogenere degli sparatutto a scorrimento (*shoot 'em up*), si riferisce a quei giochi in cui il giocatore "corre e spara" per farsi strada verso la fine del livello.

**Sandbox.** Genere di videogiochi in cui vengono messi a disposizione del giocatore molti strumenti per modificare liberamente il mondo di gioco dando sfogo alla propria creatività. In questo tipo di giochi spesso non c'è un particolare obiettivo da raggiungere.

**Sparatutto in prima persona.** In inglese *first-person shooter (FPS)*, si riferisce ai giochi con visuale in prima persona in cui il giocatore spara a tutto ciò che è ostile.

**Sprite.** Immagine utilizzata per rappresentare uno o più frame di un'animazione 2D.

**Texture.** Immagine proiettata sulle facce dei poligoni di un mesh per aggiungere particolari quali colore, trasparenza, opacità, piccole variazioni di altezza della superficie e altri dettagli grafici.

**Tick.** Termine che si riferisce sia l'evento di aggiornamento che l'intervallo di tempo fra due aggiornamenti successivi durante l'esecuzione di un gioco.

**Token.** In linguistica computazionale il token è l'unità minima di analisi di un testo, in Unreal il token corrisponde al contenuto localizzabile delle variabili di testo.

**UI.** *User interface*, interfaccia utente.

**UMG.** Acronimo per Unreal Motion Graphics UI Designer, lo strumento che Unreal mette a disposizione per realizzare le interfacce.

**VR.** *Virtual reality*, realtà virtuale. Simulazione della realtà che si svolge interamente all'interno di un mondo virtuale. L'interazione avviene attraverso appositi dispositivi pensati per aumentare il livello di coinvolgimento sia per la presentazione della scena (visori, cuffie) sia per l'acquisizione dei comandi tramite sensori (visori, guanti, controller).

**Widget.** Singolo elemento grafico che compone l'interfaccia. Il termine può essere utilizzato sia per indicare l'unità minima, che per indicare widget complessi, per esempio una pagina di menù.

## Codici

Questa sezione è dedicata alla descrizione delle funzioni presenti nelle BPL utilizzate per integrare il simulatore e le altre funzionalità. L'intero codice del gioco, compresi i BP, è incluso nelle distribuzioni sorgenti del gioco.

BPL General Utils:

- *void: MySetMstrVolume(float)*; imposta il livello audio del volume generale; converte il valore in percentuale e semplifica l'API verso la *SoundMixClass* di Unreal;
- *void: MySetSfxVolume(float)*: imposta il livello audio del volume degli effetti speciali; converte il valore in percentuale e semplifica l'API verso la *SoundMixClass* di Unreal;
- *void: MySetMscVolume(float)*: imposta il livello audio della musica; converte il valore in percentuale e semplifica l'API verso la *SoundMixClass* di Unreal;
- *{intpoint[], text[], int}: MyGetResolution()*: seleziona le risoluzioni in 16:9 e simili supportate dal gioco nella macchina su cui è in esecuzione; restituisce l'array delle risoluzioni supportate, quello delle stringhe corrispondenti e l'indice della risoluzione corrente; per impostare la risoluzione viene utilizzata una funzione dell'API standard di Unreal;

- *{intpoint[], text[], int}: MyGetFullscreenMode()*: legge dal file di configurazione (*GameUserSettings*) la modalità della finestra; restituisce l'array delle modalità, quello delle stringhe corrispondenti e l'indice della modalità selezionata;
- *{text[], int}: MyGetAA()*: legge dal file di configurazione (*GameUserSettings*) il livello di anti aliasing selezionato; restituisce l'array delle stringhe corrispondenti a tutti i livelli supportati e l'indice di quello selezionato;
- *void: MySetAA(int)*: imposta il livello di anti aliasing desiderato immettendo un comando di console;
- *{float[], text[], int}: MyGetFPS()*: legge dal file di configurazione (*GameUserSettings*) il refresh rate dello schermo; restituisce l'array delle soglie di refresh rate supportate, quello delle stringhe corrispondenti e l'indice della soglia selezionata; per impostare il refresh rate viene utilizzata una funzione dell'API standard di Unreal;
- *{text[], int}: MyGetPP()*: legge dal file di configurazione (*GameUserSettings*) il livello di *post processing* selezionato; restituisce un array delle stringhe corrispondenti a tutti i livelli supportati e l'indice di quello selezionato;
- *void: MySetPP(int)*: funzione set che imposta il livello di *post processing* desiderato immettendo un comando di console;
- *{text[], int}: MyGetShadows()*: legge dal file di configurazione (*GameUserSettings*) il livello di qualità delle ombre selezionato; restituisce l'array delle stringhe corrispondenti ai livelli di qualità selezionabili, e l'indice del livello selezionato;
- *void: MySetShadows(int)*: imposta il livello di qualità delle ombre desiderato immettendo un comando di console;
- *{text[], int}: MyGetViewDistance()*: legge dal file di configurazione (*GameUserSettings*) la distanza della visibilità del campo visivo; restituisce l'array delle stringhe corrispondenti ai livelli selezionabili e l'indice di quello selezionato; per impostare la *view distance* viene utilizzata una funzione dell'API standard di Unreal;

- *{text[], int}: MyGetVisualFX()*: legge dal file di configurazione (*GameUserSettings*) il livello di dettaglio degli effetti speciali grafici; restituisce l'array delle stringhe corrispondenti ai livelli di scalabilità selezionabili e l'indice di quello selezionato; per impostare il livello degli effetti viene utilizzata una funzione dell'API standard di Unreal;
- *{text[], int}: MyGetVSync()*: legge dal file di configurazione (*GameUserSettings*) la sincronia verticale; restituisce l'array delle stringhe corrispondenti agli stati di attivazione dell'opzione e l'indice di quello selezionato; per impostare la sincronia verticale viene utilizzata una funzione dell'API standard di Unreal;
- *{text[], int}: MyGetShowInfo()*: legge dal file di configurazione (*MoneyBin*) lo stato di attivazione del menù in sovrapposizione *SimInfo*; restituisce l'array delle stringhe corrispondenti agli stati di attivazione dell'opzione e l'indice di quello selezionato; per impostare lo stato del menù viene utilizzata una funzione della BPL Proxy To Sim Core;
- *{text[], int}: MyGetShowFPS()*: legge dal file di configurazione (*MoneyBin*) lo stato di attivazione del contatore degli fps; restituisce un array delle stringhe corrispondenti agli stati di attivazione dell'opzione e l'indice di quello selezionato; per impostare lo stato del menù viene utilizzata una funzione della BPL Proxy To Sim Core;
- *{text[], int}: MyGetUseTimeSeed()*: legge dal file di configurazione (*MoneyBin*) lo stato di utilizzo del *time seed*; restituisce un array delle stringhe corrispondenti agli stati di attivazione dell'opzione e l'indice di quello selezionato; per impostare il *time seed* viene utilizzata una funzione della BPL Proxy To Sim Core;
- *{text[], int}: MyGetLanguage()*: legge dal file di configurazione (*GameUserSettings*) la lingua selezionata; restituisce l'array delle stringhe corrispondenti alle lingue supportate e l'indice di quella selezionata;
- *void: MySetLanguage(int)*: imposta la lingua selezionata semplificando l'API verso il supporto per la localizzazione di Unreal.

#### BPL Proxy To Sim Core:

- *void: initBin()*: inizializza il *MoneyBin*;
- *void: initSim()*: inizializza il simulatore;
- *void: resetToBinDefault()*: riporta i valori ai default del *MoneyBin*;
- *void: resetGrpOptsToBinDefault()*: riporta i valori delle opzioni grafiche ai default del *MoneyBin*;
- *void: resetSimOptsToBinDefault()*: riporta i valori delle opzioni di simulazione ai default del *MoneyBin*;
- *void: resetGmeOptsToBinDefault()*: riporta i valori delle opzioni di gioco ai default del *MoneyBin*;
- *void: saveToBinFile()*: salva i valori correnti nel file di configurazione (*MoneyBin*);
- *void: readFromBinFile()*: legge i valori salvati nel file di configurazione (*MoneyBin*);
- *void: setBinUseTimeSeed(bool)*: imposta il flag relativo all'utilizzo del *time seed*;
- *bool: getBinUseTimeSeed()*: legge dal file di configurazione (*MoneyBin*) il valore del flag relativo all'utilizzo del *time seed*;
- *void: setBinRandGenSeed(int)*: imposta il valore del seme per la generazione dei valori pseudocasuali;
- *int: getBinRandGenSeed()*: legge dal file di configurazione (*MoneyBin*) il valore del seme per la generazione dei valori pseudocasuali;
- *void: setBinZ50TiT(int)*: imposta il tempo in  $\mu\text{s}$  di ionizzazione dei tubi Z50T;
- *int: getBinZ50TiT()*: legge dal file di configurazione (*MoneyBin*) il tempo in  $\mu\text{s}$  di ionizzazione dei tubi Z50T;
- *void: setBinZ50TdT(int)*: imposta il tempo in  $\mu\text{s}$  di deionizzazione dei tubi Z50T;
- *int: getBinZ50TdT()*: legge dal file di configurazione (*MoneyBin*) il tempo in  $\mu\text{s}$  di deionizzazione dei tubi Z50T;
- *void: setBinFPCf(int)*: imposta il coefficiente della potenza della ventilatore;

- *int: getBinFPCf()*: legge dal file di configurazione (*MoneyBin*) il coefficiente della potenza del ventilatore;
- *void: setBinFMRPM(int)*: imposta il numero massimo di giri al minuto del ventilatore;
- *int: getBinFMRPM()*: legge dal file di configurazione (*MoneyBin*) il numero massimo di giri al minuto del ventilatore;
- *void: setBinFFCf(int)*: imposta il coefficiente di frizione delle pale del ventilatore;
- *int getBinFFCf()*: legge dal file di configurazione (*MoneyBin*) il coefficiente di frizione delle pale del ventilatore;
- *void: setBinFanMeshSwTrsh(int)*: imposta il valore della soglia di giri al minuto oltre il quale il mesh delle pale viene sostituito dal solido di rotazione semitrasparente;
- *int: getBinFanMeshSwTrsh*: legge dal file di configurazione (*MoneyBin*) il valore della soglia di giri al minuto oltre il quale il mesh delle pale viene sostituito dal solido di rotazione semitrasparente;
- *void: setBinAmbCurTemp(int)*: imposta la temperatura ambiente della stanza;
- *int: getBinAmbCurTemp()*: legge dal file di configurazione (*MoneyBin*) la temperatura ambiente della stanza;
- *void: setBinFanClCf(int)*: imposta il coefficiente di raffreddamento del ventilatore;
- *int: getBinFanClCf()*: legge dal file di configurazione (*MoneyBin*) il valore del coefficiente di raffreddamento del ventilatore;
- *void: setBinAdderHCf(int)*: imposta il coefficiente di surriscaldamento dell'addizionale;
- *int: getBinAdderHCf()*: legge dal file di configurazione (*MoneyBin*) il valore del coefficiente di surriscaldamento dell'addizionale;
- *void: setBinAdderCCf(int)*: imposta il coefficiente di raffreddamento dell'addizionale;
- *int: getBinAdderCCf()*: legge dal file di configurazione (*MoneyBin*) il valore del coefficiente di raffreddamento dell'addizionale;

- *void: strtSim()*: avvia il simulatore;
- *void: updateSim()*: aggiorna lo stato del simulatore;
- *void: stopSim()*: arresta il simulatore;
- *void: toggleSimSwAdder()*: gestisce accensione e spegnimento dell'interruttore generale dell'addizionatore;
- *void: toggleSimSwFan()*: gestisce accensione e spegnimento dell'interruttore del ventilatore;
- *void: setSimBit(int addend, int pos, int val)*: imposta nel simulatore lo stato del bit specificato come addendo e posizione; l'addendo può essere un valore fra 0 e 2, indicando rispettivamente il riporto (solo posizione 0) o il primo o secondo addendo, con posizione da 0 a 5.
- *int: getSimBit(int result, int bit)*: interroga il simulatore per ottenere lo stato del bit desiderato;
- *int: getSimFanAngle()*: interroga il simulatore per ottenere l'angolo di rotazione delle pale del ventilatore;
- *int: getSimFanRPM()*: interroga il simulatore per ottenere il numero di giri al minuto del ventilatore;
- *FString: getSimShowStr()*: interroga il simulatore per ottenere la stringa di informazione dello stato del simulatore in versione ridotta; restituisce una FString;
- *FString: getSimFullShowStr()*: interroga il simulatore per ottenere la stringa di informazione completa dello stato del simulatore;
- *int: getSimChsDmgLvl(int tp, int sn)*: interroga il simulatore per ottenere il livello di danno del telaietto desiderato, specificato utilizzando il tipo e il numero di serie;
- *int: getSimChsResLvl(int tp, int sn)*: interroga il simulatore per ottenere il livello di accensione delle resistenze del telaietto desiderato, specificato utilizzando il tipo e il numero di serie;
- *int: getSimChsTemp(int tp, int sn)*: interroga il simulatore per ottenere la temperatura del telaietto desiderato, specificato utilizzando il tipo e il numero di serie;

- *int: getSimChsTypeByPos(int ar, int sl)*: interroga il simulatore per ottenere il tipo di un telaietto in base alla sua posizione corrente, specificata in termini di area e slot; l'area può essere un valore da 0 a 2, rispettivamente l'area dei telaietti di riporto, l'area dei telaietti di somma e il bancone da lavoro, con slot che vanno da 0 a 5 per le aree dell'addizionatore e da 0 a 13 per l'area del bancone;
- *int: getSimChsSNumByPos(int ar, int sl)*: interroga il simulatore per ottenere il numero di serie di un telaietto in base alla sua posizione corrente, specificata in termini di area e slot;
- *int: getSimDmgStateByPos(int ar, int sl)*: interroga il simulatore per ottenere lo stato di danneggiamento di un telaietto in base alla sua posizione corrente, specificata in termini di area e slot;
- *int: getSimResStateByPos(int ar, int sl)*: interroga il simulatore per ottenere il livello di accensione delle resistenze di un telaietto in base alla sua posizione corrente, specificata in termini di area e slot;
- *int: simMove(int aF, int sF, int aT, int sT)*: sposta un telaietto dalla sua posizione corrente a quella desiderata, definita in termini di area e slot di provenienza (aF e sF) e area e slot di destinazione (aT e sT);
- *void: setChsMoved()*: utilizzata per ricordare che è stato effettuato uno spostamento di telaietti;
- *bool: getChsMoved()*: utilizzata per sapere se è avvenuto uno spostamento di telaietti; utilizzata per l'aggiornamento della scena 3D;
- *int: getLastMoveCType()*: utilizzata per ottenere il tipo dell'ultimo telaietto spostato; utilizzata per l'aggiornamento della scena 3D;
- *int: getLastMoveCSNum()*: utilizzata per ottenere il numero di serie dell'ultimo telaietto spostato;
- *int: getLastMoveAreaT()*: utilizzata per ottenere l'area dell'ultimo telaietto spostato;
- *int: getLastMoveSlotT()*: utilizzata per ottenere lo slot dell'ultimo telaietto spostato;



- *void: setBinShowFPS(bool)*: funzione set che imposta lo stato di visualizzazione del contatore degli FPS;
- *bool: getBinShowFPS()*: utilizzata per ottenere lo stato di visualizzazione del contatore degli FPS;
- *void: setBinShowInfo(bool)*: salva nel file di configurazione (*MoneyBin*) lo stato di visualizzazione del menù *SimInfo* per la nuova sessione;
- *bool: getBinShowInfo()*: legge dal file di configurazione (*MoneyBin*) lo stato di visualizzazione del menù *SimInfo*;
- *void: setMstrVolume(float)*: imposta il livello audio del volume generale;
- *float: getMstrVolume()*: legge dal file di configurazione (*MoneyBin*) il livello audio del volume generale;
- *void: setSfxVolume(float)*: imposta il livello audio del volume degli effetti speciali;
- *float: getSfxVolume()*: legge dal file di configurazione (*MoneyBin*) il livello audio del volume degli effetti speciali;
- *void: setMscVolume(float)*: imposta il livello audio della musica;
- *float: getMscVolume()*: legge dal file di configurazione (*MoneyBin*) il livello audio della musica;

## Bibliografia

- Ahl, David Hollerith. 1978. *Basic Computer Games, Microcomputer Edition*. New York: Workman Publishing.
- Al-Rawi Ahmed. 2018. "Video games, terrorism, and ISIS's Jihad 3.0". *Terrorism and Political Violence* 30, no. 4 (luglio-agosto): 740-760.  
<https://doi.org/10.1080/09546553.2016.1207633>.
- Anthes, Christoph, Rubén Jesús García-Hernández, Markus Wiedemann e Dieter Kranzlmüller. 2016. "State of the art of virtual technology". IEEE Aerospace Conference, pp. 1-19.  
<https://doi.org/10.1109/AERO.2016.7500674>.
- Azuma, Ronald. 1997. "A Survey of Augmented Reality", *Presence: Teleoperators and Virtual Environments* 6, no. 4 (agosto): 355-385.  
<https://doi.org/10.1162/pres.1997.6.4.355>.
- Backus, John Warner. 1956. *The Fortran Automatic coding System for the IBM 704 EDPM*. New York: IBM Corp. Acceduto via ProgettoHMR (precedentemente su The Fortran Company) giugno 2021.  
[https://www.progettohmr.it/Documentazione/Archivio/Misc/1956\\_FortranForTheIBM704.pdf](https://www.progettohmr.it/Documentazione/Archivio/Misc/1956_FortranForTheIBM704.pdf).
- Billinghurst, Mark, Adrian Clark e Gun Lee. 2015. "A Survey of Augmented Reality". *Foundations and Trends® in Human-Computer Interaction* 8, no. 2-3: 73-272.  
<http://doi.org/10.1561/11000000049>.
- Bodrato, Stefano, Fabrizio Caruso e Giovanni A. Cignoni. 2019. "Discovering Eastern Europe PCs by Hacking them... Today". *Histories of Computing in Eastern Europe*, atti del IFIP WG 9.7 Int. Workshop on the History of Computing, Poznań, 19–21 settembre 2018. IFIP AICT (a cura di C. Leslie, M. Schmitt) no. 549: 279-294. Springer. ISBN: 978-3-030-29159-4. Acceduto via ProgettoHMR giugno 2021.

- [https://www.progettohmr.it/Documenti/HMR\\_2019s\\_SBFCGC-IFIP97Conf18Proc.pdf](https://www.progettohmr.it/Documenti/HMR_2019s_SBFCGC-IFIP97Conf18Proc.pdf).
- Bondioli, Mariasole. 2020. “Developing technological solutions to assist children with ASD with application to real-life and diagnostic scenarios”. Tesi di dottorato di ricerca. Università di Pisa.  
<https://etd.adm.unipi.it/theses/available/etd-02282020-163424/>.
- Bush, Vannevar. 1945a. “As we may think”. *The Atlantic Monthly*, luglio, 1945.  
<https://www.theatlantic.com/magazine/archive/1945/07/as-we-may-think/303881/>.
- Bush, Vannevar. 1945b. “As we may think”. *Life Magazine*, 10 settembre, 1945.  
<https://oldlifemagazine.com/september-10-1945-life-magazine.html>.
- Cairns, Paul, Anna Cox e A. Imran Nordin. 2014. “Immersion in Digital Games: Review of Gaming Experience Research”. In *Handbook of Digital Games*, a cura di Marios C. Angelides e Harry Agius, 339–361. John Wiley & Sons, Inc.  
<https://doi.org/10.1002/9781118796443.ch12>.
- Centro Studi Calcolatrici Elettroniche. 1956. *Relazione sulle attività del CSCE dal 23 dicembre 1955 al 31 luglio 1956*. Archivio Generale di Ateneo dell’Università di Pisa. Acceduto via ProgettoHMR giugno 2021.  
[https://www.progettohmr.it/Documentazione/Archivio/DocCEP/1956\\_0731\\_RelazCSCE.pdf](https://www.progettohmr.it/Documentazione/Archivio/DocCEP/1956_0731_RelazCSCE.pdf).
- Cignoni, Alessandro, Giovanni A. Cignoni, Giuliano Pacini e Daniele Ronco. 2020. “RiBau: il CANE torna a correre, un calcolatore didattico del 1970”. *Mondo Digitale* 89 (dicembre): 1-14. Acceduto via ProgettoHMR giugno 2021.  
[https://www.progettohmr.it/Documenti/HMR\\_2020s\\_ACGCGPDR-CANE-MondoDigPubb.pdf](https://www.progettohmr.it/Documenti/HMR_2020s_ACGCGPDR-CANE-MondoDigPubb.pdf).
- Cignoni, Giovanni A. e Cinzia Colosimo. 2014. “Raccontare il calcolo, senza fare conti”, intervento a Contact Zone - XXIV Congresso dell'Associazione Nazionale dei Musei Scientifici, Livorno, 11-13 novembre 2014.  
[https://www.progettohmr.it/Documenti/HMR\\_2014s\\_GCCC-ANMS-ab.pdf](https://www.progettohmr.it/Documenti/HMR_2014s_GCCC-ANMS-ab.pdf).

- Cignoni, Giovanni A. e Fabio Gadducci. 2013. “La storia dell'informatica al Museo degli Strumenti per il Calcolo di Pisa”, atti di Pianeta Galileo 2012, Reg. Toscana. ISBN: 978-88-89365-36-6. Acceduto via ProgettoHMR giugno 2021.  
[https://www.progettohmr.it/Documenti/HMR\\_2013s\\_GCFG-PiGal-tx.pdf](https://www.progettohmr.it/Documenti/HMR_2013s_GCFG-PiGal-tx.pdf).
- Cignoni, Giovanni A. e Nicola Pratelli. 2018. “Raccontare la storia dell'informatica giorno per giorno”, intervento a Metti la Storia al lavoro - II Conferenza Italiana di Public History, Pisa, 11-15 giugno 2018. Acceduto via ProgettoHMR giugno 2021.  
[https://www.progettohmr.it/Documenti/HMR\\_2018s\\_GCNP-AIPHabs.pdf](https://www.progettohmr.it/Documenti/HMR_2018s_GCNP-AIPHabs.pdf).
- Cignoni, Giovanni A. e Stefano Paci. 2012. “UML Modelling and Code Generation For Agent-based, Discrete Events Simulation”, atti di International Workshop on Applied Modeling and Simulation, Roma, 24-27 settembre 2012. ISBN: 978-88-97999-06-5. Acceduto via ProgettoHMR giugno 2021.  
[https://www.progettohmr.it/Documenti/HMR\\_2012s\\_GCSP-WAMS-cr.pdf](https://www.progettohmr.it/Documenti/HMR_2012s_GCSP-WAMS-cr.pdf).
- Cignoni, Giovanni A., Diego Ceccarelli e Claudio Imbrenda. 2009. “Il “restauro” del software di sistema della Macchina Ridotta del 1956”, atti del 47mo Congresso Nazionale AICA, Roma, 5 novembre 2009. ISBN: 978-88-9016-208-4. Acceduto via ProgettoHMR giugno 2021.  
[https://www.progettohmr.it/Documenti/HMR\\_2009s\\_GCDCCI-AICA-cr.pdf](https://www.progettohmr.it/Documenti/HMR_2009s_GCDCCI-AICA-cr.pdf).
- Cignoni, Giovanni A., e Simone Masoni, “GeneSim: modellazione e generazione di codice per simulatori di sistemi dinamici”, atti del 44mo Congresso Nazionale AICA, Cesena, 2006, ISBN: 88-6055-075-0. Acceduto via ProgettoHMR giugno 2021.  
[https://www.progettohmr.it/Documenti/HMR\\_2006s\\_GCSM-AICA-cr.pdf](https://www.progettohmr.it/Documenti/HMR_2006s_GCSM-AICA-cr.pdf).
- Cignoni, Giovanni A., Fabio Gadducci e Daniele Ronco. 2013. “I documenti raccontano le storie delle CEP”. In *Quaderni della Fondazione Galileo Galilei*, a cura di Fabio Gadducci, 119-146. Pisa: Pisa University Press. Acceduto via ProgettoHMR giugno 2021.  
[https://www.progettohmr.it/Documenti/HMR\\_2013s\\_GCFGDR-DocCEP.pdf](https://www.progettohmr.it/Documenti/HMR_2013s_GCFGDR-DocCEP.pdf).

- Cignoni, Giovanni A., Fabio Gadducci e Stefano Paci. 2015. “A Virtual Experience on the Very First Italian Computer”. *Journal on Computing and Cultural Heritage* 7, no. 4 (febbraio): articolo no. 21, 1-23.  
<https://doi.org/10.1145/2629484>.
- Cignoni, Giovanni A., Fabio Gadducci. 2020. “Pisa, 1954–1961: Assessing Key Stages of a Seminal Italian Project”. *IEEE Annals of the History of Computing* 42, no. 2 (aprile-giugno): 6-19. Acceduto via ProgettoHMR giugno 2021.  
[https://www.progettohmr.it/Documenti/HMR\\_2020s\\_GCFCG-CEP-IEEEAnnals.pdf](https://www.progettohmr.it/Documenti/HMR_2020s_GCFCG-CEP-IEEEAnnals.pdf).
- Cignoni, Giovanni A., Nicolò Pratelli e Maria Serena Papa. 2019. “Raccontare sul luogo: le Calcolatrici Elettroniche Pisane”, *Museologia Scientifica Memorie*, no. 19: 212-215. ISBN: 978-88-908819-2-3.  
[https://www.progettohmr.it/Biblio/2019/HMR\\_2019s\\_GCNPMP-MusScLuoghiCEP.pdf](https://www.progettohmr.it/Biblio/2019/HMR_2019s_GCNPMP-MusScLuoghiCEP.pdf).
- Cignoni, Giovanni A., Tommaso Mongelli e Leonora Cappellini. 2015. “Games, from Engaging to Understanding: a Perspective from a Museum of Computing Machinery”, atti della 14th International Conference on Entertainment Computing, Trondheim, 30 settembre - 2 ottobre 2015. Acceduto via ProgettoHMR giugno 2021.  
[https://www.progettohmr.it/Biblio/2015/HMR\\_2015s\\_GCLCTM-ICEC.pdf](https://www.progettohmr.it/Biblio/2015/HMR_2015s_GCLCTM-ICEC.pdf).
- Cignoni, Giovanni A.. 2013. “Il carattere delle telescriventi Olivetti T2”. Acceduto via ProgettoHMR giugno 2021.  
<https://www.progettohmr.it/Documenti/HMR-OliT2-04.pdf>.
- Cignoni, Giovanni A.. 2014. “CEP, storie delle — Cap. 2 Com’era bello il sincrotrone”. *PaginaQ* no. 15 (agosto), 2014.  
[https://www.progettohmr.it/PaginaQ/HMR\\_2014d\\_GC\\_4CCsC-15.pdf](https://www.progettohmr.it/PaginaQ/HMR_2014d_GC_4CCsC-15.pdf).
- Cignoni, Giovanni A.. 2021. “Virtuale sì, virtuale no”. *Museologia Scientifica Memorie* 2, giugno 2021: 12-16.  
[http://www.anms.it/riviste/dettaglio\\_rivista/41](http://www.anms.it/riviste/dettaglio_rivista/41).

- Conversi, Marcello. 24 luglio 1957. *Circolare ai colleghi*. Archivio Generale di Ateneo dell'Università di Pisa. Acceduto via ProgettoHMR giugno 2021. [https://www.progettohmr.it/Biblio/1957/1957\\_0724\\_LettConversi.pdf](https://www.progettohmr.it/Biblio/1957/1957_0724_LettConversi.pdf).
- Cooper, Alan, Robert Reinmann, David Cronin, Chris Noessel. 2007 *About Face: The Essentials of Interaction Design*. Indianapolis, IN, USA: Wiley Publishing (prima edizione pubblicata nel 1995). ISBN: 0470084111.
- Csikszentmihalyi, Mihaly. 1996. *Flow and the Psychology of Discovery and Invention*. New York: Harper Collins. ISBN: 978-0-06-228325-2.
- Cymet, Eli, Tyler Moldenhauer, Chad Moldenhauer e Jared Moldenhauer. 2020. *The Art of Cuphead*. Milwaukee: Dark Horse Books. ISBN: 978-1-50671-320-5.
- D.W.F. Van Krevelen e R. Poelman. 2010. "A survey of augmented reality technologies, applications and limitations". *The International Journal of Virtual Reality* 9, no. 2 (1 gennaio): 1-20. <https://doi.org/10.20870/IJVR.2010.9.2.2767>.
- De Smale, Stephanie, Martijn J. L. Kors e Alyea M. Sandoval. 2019. "The Case of This War of Mine: A Production Studies Perspective on Moral Game Design". *Games and Culture* 14, no. 4 (giugno): 387–409. <https://doi.org/10.1177/1555412017725996>.
- Desurvire, Heather, Martin Caplan e Jozsef A. Toth. 2004. "Using heuristics to evaluate the playability of games", CHI '04 Extended Abstracts on Human Factors in Computing Systems (CHI EA '04). Association for Computing Machinery, New York, NY, USA, 1509–1512. <https://doi.org/10.1145/985921.986102>.
- Deterding Sebastian, Dan Dixon, Rilla Khaled e Lennart Nacke. 2011. "From game design elements to gamefulness: defining "gamification" ", Proceedings of the 15th International Academic MindTrek Conference: Envisioning Future Media Environments (MindTrek '11). Association for Computing Machinery, New York, NY, USA, 9–15. <https://doi.org/10.1145/2181037.2181040>.
- Di Serio Ángela, María Blanca Ibáñez e Carlos Delgado Kloos. 2013. "Impact of an AR system on students' motivation for a visual art course". *Computers &*

- Education* 68 (ottobre): 586-596.  
<https://doi.org/10.1016/j.compedu.2012.03.002>.
- Di Tore, Stefano, Maristella Fulgione e Maurizio Sibilio. 2014. “Dislessia e Videogames: Il Potenziale Didattico dei Videogiochi”. *Mediterranean Journal of Social Sciences* 5, no. 23 (novembre): 1165-1171.  
<http://dx.doi.org/10.5901/mjss.2014.v5n23p1165>.
- Feurzeig, Wallace, S. Papert, M. Bloom, R. Grant e C. Solomon. 30 novembre 1969. *Programming-Languages as a Conceptual Framework for Teaching Mathematics. Final Report on the First Fifteen Months of the LOGO Project*. Acceduto via Eric giugno 2021.  
<https://eric.ed.gov/?id=ED038034>.
- Frerich, Sulamith, Daniel Kruse, Marcus Petermann e Andreas Kilzer. 2014. “Virtual Labs and Remote Labs: Practical experience for everyone”. IEEE Global Engineering Education Conference (EDUCON), 312-314.  
doi: 10.1109/EDUCON.2014.6826109.
- Johnson, Daniel, e Janet Wiles. 2003. “Effective affective user interface design in games”. *Ergonomics* 46, no. 13-14 (20 ottobre-5 novembre): 1332–1345.  
<https://doi.org/10.1080/00140130310001610865>.
- Khalid, Halimahtun M. 2006. “Embracing diversiti in user needs for affective design”. *Applied Ergonomics* 37, no. 4 (luglio): 409–418.  
<https://doi.org/10.1016/j.apergo.2006.04.005>.
- Lieberman, Debra A. 2012. “Video Games for Diabetes Self-Management: Examples and Design Strategies”. *Journal of Diabetes Science and Technology* 6, no. 4 (luglio): 802–6.  
<https://doi.org/10.1177/193229681200600410>.
- Malkin, G. 1996. “RFC1983: Internet Users’ Glossary”. RFC Editor, USA.  
<https://doi.org/10.17487/RFC1983>.
- Malone, Thomas W. 1982. “Heuristics for designing enjoyable user interfaces: lessons from computer games”, atti della Conference on Human Factors in Computing Systems (CHI '82), New York, marzo 1982. Association for

- Computing Machinery, New York, NY, USA, 63–68.  
<https://doi.org/10.1145/800049.801756>.
- McMichael, Andrew. 2007. “PC Games and the Teaching of History”. *The History Teacher* 40, no. 2 (febbraio): 203-218.  
<https://doi.org/10.2307/30036988>.
- Nagendran, Myura, Kurinchi Selvan Gurusamy, Rajesh Aggarwal, Marilena Loizidou e Brian R. Davidson. 2013. “Virtual reality training for surgical trainees in laparoscopic surgery”. *Cochrane Database of Systematic Reviews* 8 (27 agosto). Acceduto via Cochrane Library giugno 2021.  
<https://doi.org/10.1002/14651858.cd006575.pub3>.
- Ng, Yiing Y’ng, Chee Weng Khong e Robert Jeyakumar Nathan. 2018. “Evaluating Affective User-Centered Design of Video Games Using Qualitative Methods”. *International Journal of Computer Games Technology* vol. 2018 (giugno), ID articolo 3757083.  
<https://doi.org/10.1155/2018/3757083>.
- Resnick, Mitchel, John Maloney, Andrés Monroy-Hernández, Natalie Rusk, Evelyn Eastmond, Karen Brennan, Amon Millner, Eric Rosenbaum, Jay Silver, Brian Silverman e Yasmin Kafai. 2009. “Scratch: programming for all. Commun”. *Communications of the ACM* 52, no. 11 (novembre): 60–67.  
<https://doi.org/10.1145/1592761.1592779>.
- Ross, Harold. 1953. “The Arithmetic Element of the IBM Type 701 Computer”. *Proceedings of the IRE* 41, no. 10: 1287-1294. Acceduto via ProgettoHMR giugno 2021.  
[https://www.progettohmr.it/Biblio/1953/1953\\_ProcIRE-IBM701Arith.pdf](https://www.progettohmr.it/Biblio/1953/1953_ProcIRE-IBM701Arith.pdf).
- Rughiniş, Răzvan e Ştefania Matei. 2015. “Play to Remember: The Rhetoric of Time in Memorial Video Games”. *Human-Computer Interaction: Interaction Technologies*, a cura di Masaaki Kurosu, 628-639. HCI 2015. Lecture Notes in Computer Science, vol 9170. Cham: Springer.  
[https://doi.org/10.1007/978-3-319-20916-6\\_58](https://doi.org/10.1007/978-3-319-20916-6_58).
- Schrier, Karen. 2014. “Using digital games to teach history and historical thinking”. In *Learning, Education and Games*, a cura di Karen Schrier, 73–91.



- Pittsburgh: ETC Press.  
<https://dl.acm.org/doi/abs/10.5555/2811147.2811152>.
- Schut, Kevin. 2007 “Strategic Simulations and Our Past: The Bias of Computer Games in the Presentation of History”. *Games and Culture* 2, no. 3 (luglio): 213–35.  
<https://doi.org/10.1177/1555412007306202>.
- Schutte, Nicola S., e Emma J. Stilinović. 2017. “Facilitating empathy through virtual reality”. *Motivation and Emotion* 41, no. 3 (giugno): 708–712.  
<https://doi.org/10.1007/s11031-017-9641-7>.
- Schwaber, Ken. 1997. “SCRUM Development Process”. In *Business Object Design and Implementation*, a cura di Sutherland, Jeff, Casanave C., Miller J., Patel P. e Hollowell G., 117-134. Londra: Springer.  
[https://doi.org/10.1007/978-1-4471-0947-1\\_11](https://doi.org/10.1007/978-1-4471-0947-1_11).
- Università di Pisa. 1959. *Informazioni generali sul Centro di Studi sulle Calcolatrici Elettroniche*. Acceduto via ProgettoHMR giugno 2021.  
[https://www.progettohmr.it/Documentazione/Archivio/DocCEP/1959\\_InfoGenCSCE.pdf](https://www.progettohmr.it/Documentazione/Archivio/DocCEP/1959_InfoGenCSCE.pdf).
- Uricchio, William. 2005. “Simulation, history and computer games”. In *Handbook of Computer Game Studies*, a cura di Jeffrey Goldstein e Joost Raessans, 327-338. Cambridge: MIT Press.
- Zeldman, Jeffrey (@zeldman). 2008. “Content precedes design. Design in the absence of content is not design, it's decoration.”. Twitter, 5 maggio, 2008.  
<https://twitter.com/zeldman/status/804159148>.

## Bibliografia videoludica

- 11 bit studios. This War of Mine. 11 bit studios. PC/Mac/Linux/PlayStation 4/Xbox One/Nintendo Switch/iOS/Android. 2014.
- Alfieri, Marco. Call of Salveenee. PC. 2015.
- Berretta, Michele. L'area dei Lungarni di Pisa nel tardo Medioevo (XIV-XV secolo). un tentativo di ricostruzione in 3D. PC. 2012.
- Creative Assembly. Napoleon: Total War. Sega, Typhoon Games e Feral Interactive. PC/Mac. 2010.
- Creative Assembly. Shogun: Total War. Electronic Arts e Sold Out. PC. 2000.
- Creative Assembly. Total War: Rome II. Sega. PC/Mac. 2013.
- Creative Assembly. Total war: Warhammer. Sega. PC/Mac/Linux. 2016.
- Croteam. Serious Sam 3: BFE. Devolver Digital. PC/Mac/Linux/PlayStation 3/PlayStation 4/Xbox 360/Xbox One/Nintendo Switch. 2011.
- Crytek. Crysis 2. Electronic Arts. PC/PlayStation 3/Xbox 360. 2011.
- Danger Close Games e EA DICE. Medal of Honor. Electronic Arts. PC/PlayStation 3/Xbox 360. 2010.
- DICE. Battlefield 1. Electronic Arts. PC/PlayStation4/Xbox One. 2016.
- Firaxis Games. Sid Meier's Civilization III. Infogrames Entertainment e MacSoft. PC/Mac. 2001.
- FromSoftware. Dark Souls. Namco Bandai Games. PC/PlayStation 3/PlayStation 4/Xbox 360/Xbox One/Nintendo Switch. 2011.
- Global Islamic Media Front. Quest for Bush. PC. 2005.
- Gruppo di Filiera dei Produttori Italiani di Videogiochi. Gioventù ribelle. Dipartimento per le politiche giovanili e il servizio civile universale. PC. 2011.
- Innersloth. Among Us. Innersloth. PC/PlayStation 3/PlayStation 4/PlayStation 5/Xbox One/Xbox Series X/S/Nintendo Switch/iOS/Android. 2018.
- Irrational Games. Bioshock Infinite. 2K Games PC/Mac/Linux/PlayStation 3/PlayStation 4/Xbox/Xbox 360/Xbox One/Nintendo Switch. 2013.

Irrational Games. Bioshock. 2K Games. PC/Mac/PlayStation 3/ PlayStation 4/Xbox/Xbox 360/Xbox One/Nintendo Switch/iOS. 2007.

Mojang. Minecraft. Mojang, Microsoft Studios e Sony Computer. PC/Mac/Linux/PlayStation 3/PlayStation 4/PlayStation Vita/Xbox 360/Xbox One/Nintendo Wii U/Nintendo 3DS/Nintendo Switch/iOS/Android/Windows Phone. 2009.

Paradox Development. Europa Universalis II. Strategy First. PAN Vision e Ubisoft. PC/Mac. 2001.

Pietroni, Eva e Leonardo Rescic. Difendiamo le Mura. PC. 2015.

Rare. Kinect Sports Rivals. Xbox One. 2014.

Rockstar North. Grand Theft Auto IV. Rockstar Games. PC/PlayStation 3/Xbox 360/Xbox One. 2008.

Rockstar North. Grand Theft Auto V. Rockstar Games. PC/PlayStation 3/PlayStation 4/PlayStation 5/Xbox 360/Xbox One/Xbox Series X/S. 2013.

Rockstar North. Grand Theft Auto: San Andreas. Rockstar Games. PC/Mac/PlayStation 2/PlayStation 3/Xbox/Xbox 360/iOS/Android/Windows Phone/Fire OS. 2004.

Rockstar North. Grand Theft Auto: Vice City. Rockstar Games. PC/Mac/PlayStation 2/Xbox/iOS/Android/Fire OS. 2002.

Rockstar Studios. Red Dead Redemption II. Rockstar Games. PC/PlayStation 4/Xbox One/Stadia. 2019.

Schrier, Karen. Reliving the Revolution. Schrier, Karen. PC. 2005.

Sinclair, Finn. The VR Museum of Fine Art. Sinclair, Finn. PC/Oculus Rift/Valve Index/HTC Vive. 2016.

Sledgehammer Games. Call of Duty: WWII. Activision. PC/PlayStation 4/Xbox One. 2017.

Sonic team. Sonic the Hedgehog. Sega. Sega Genesis. 1991.

Square. Final Fantasy VII. Square Enix. 1997. PC/PlayStation/PlayStation4/Xbox One/Nintendo Switch/iOS/Android.

Studio MDHR. Cuphead. Studio MDHR. PC/Mac/PlayStation 4/Xbox One/Nintendo Switch. 2017.

Techland. Call of Juarez: Gunslinger. Ubisoft. PC/PlayStation 3/Xbox 360/Nintendo Switch. 2013.

U.S. Army. America's Army: Proving Grounds. U.S. Army. PC/Mac/Linux/PlayStation 4. 2015.

Ubisoft Montpellier. Valiant Hearts: The Great War. PC/PlayStation 3/PlayStation 4/Xbox 360/Xbox One/Nintendo Switch/iOS/Android. 2014.

Ubisoft Montréal. Assassin's Creed II. PC/Mac/PlayStation 3/PlayStation 4/Xbox 360/Xbox One. 2009.

Ubisoft Montreal. Assassin's Creed: Brotherhood. Ubisoft. PC/Mac/PlayStation 3/PlayStation 4/Xbox 360/Xbox One. 2010.

Ubisoft. Rabbids Coding!. Ubisoft. PC. 2019.

Valve. Half-Life 2. Valve. PC/Mac/Linux/PlayStation 3/Xbox/Xbox 360/Android. 2004.

Valve. Half-Life Alyx. Valve. PC/Linux. 2020.

WNET Thirteen. Mission US. WNET Thirteen. PC. 2009.

## Sitografia e altri media

- Blender. n. d. Acceduto giugno 2021. <https://www.blender.org/>.
- Brooks, Mel, regista. 1974. *Young Frankenstein*. USA: 20th Century Fox.
- Gimp, n. d. Acceduto giugno 2021. <https://www.gimp.org/>.
- Grünwald, Sebastian. 2012. “VCFe - Vintage Computer Festival Europe, Munich 2012”. Registrato nel 2012 al Vintage Computer Festival Europa 2012, Monaco di Baviera, DE. Video.  
<https://www.youtube.com/watch?v=-hwafAJTd3A>.
- Mission US. 2009. Acceduto giugno 2021. <https://www.mission-us.org/>.
- ProgettoHMR. n. d. “Quattro chiacchiere sul calcolo, senza fare conti”. Acceduto giugno 2021.  
<https://www.progettohmr.it/PaginaQ/>.
- ProgettoHMR. n. d. Acceduto giugno 2021. <https://www.progettohmr.it/>.
- Raymond, Eric Steven. 2004. “The Jargon File, version 4.4.8”. Acceduto giugno 2021. <http://catb.org/jargon/>.
- Schwaber, Ken e Jeff Sutherland. 2020. “La Guida Scrum”. Ultima modifica novembre 2020.  
<https://scrumguides.org/docs/scrumguide/v2020/2020-Scrum-Guide-US.pdf#zoom=100>.
- Scratch. 2009. Acceduto giugno 2021. <https://scratch.mit.edu/>.
- Scrum Guides. 2020. Ultima modifica novembre 2020. <https://scrumguides.org/>.
- Sharpsteen, Ben, regista. 1937. *Don Donald*. USA: Walt Disney Productions.  
Acceduto via YouTube giugno 2021.  
[https://www.youtube.com/watch?v=Z\\_gtYcYU5oc](https://www.youtube.com/watch?v=Z_gtYcYU5oc).
- The National Museum of Computing. n. d. “3D Virtual Tour”. Acceduto giugno 2021.  
<https://www.tnmoc.org/3d-virtual-tour>.
- Unreal Engine. n. d. Acceduto giugno 2021. <https://www.unrealengine.com/en-US/>.
- Visual Studio, sito web. n. d. Acceduto giugno 2021.  
<https://visualstudio.microsoft.com/it/>.