

UNIVERSITÀ DEGLI STUDI DI MILANO
FACOLTÀ DI SCIENZE E TECNOLOGIE

DIPARTIMENTO DI INFORMATICA
GIOVANNI DEGLI ANTONI



Corso di Laurea Magistrale in Informatica

PIATTAFORMA DI CLOUD GAMING PER GIOCHI ARCADE

Relatore: Prof. Dario Maggiorini
Correlatore: Prof. Davide Gadia

Tesi di Laurea di:
Michele Maione
Matr. Nr. 931468

Anno Accademico 2020-2021

i

“Se non così, come? E se non ora, quando?”

—Primo Levi

Ringraziamenti

Rivolgo il primo ringraziamento al prof. Dario Maggiorini per il suo continuo supporto e per avermi proposto questo progetto con la codebase di C/C++ più grande e ben strutturata che abbia mai compilato e modificato. È stata un'esperienza entusiasmante e altamente formativa.

Ringrazio il prof. Davide Gadia per la sua guida utile durante i miei sforzi verso il completamento della presente tesi.

Milano, luglio 2021

Sommario

Negli ultimi anni sono apparse molte piattaforme che sfruttano il paradigma del cloud computing per offrire servizi accessibili su richiesta e da remoto per archiviare file, utilizzare le suite per l'ufficio, vedere film e serie TV, ascoltare musica e a partire dal 2011 anche giocare.

Il cloud gaming è un servizio che unisce il cloud computing e il live streaming per rendere possibile giocare in remoto senza scaricare o installare il gioco sul device dell'utente, in pratica consente di archiviare ed eseguire i videogiochi su un server remoto e trasmettere l'output audio-video all'utente sul proprio dispositivo.

Per far conoscere alle nuove generazioni i videogiochi che hanno fatto la storia e dare la possibilità di poter giocare ancora a macchine che ormai hanno cessato di funzionare per motivi di obsolescenza, sfruttando due tecnologie entrate a far parte della quotidianità, il live streaming e il cloud computing, in questo lavoro si propone la creazione di una piattaforma di cloud gaming. La piattaforma permetterà lo streaming audio-video, direttamente e su richiesta, dei videogiochi da un server remoto ad un client (computer, console e telefono). Il gioco è archiviato, eseguito e renderizzato su un server remoto; l'input (tastiera e gamepad) viene inviato dal client al server e lì processato. Il cloud gaming permette di iniziare a giocare immediatamente poiché il gioco è già installato sul server offrendo agli utenti un rapido accesso indipendentemente dal sistema operativo e dalle capacità hardware del client utilizzato. Infine la piattaforma, indirettamente, garantisce la gestione dei diritti digitali (DRM) per gli editori. Per questo progetto verrà ampliato il software MAME (rilasciato sotto licenza GNU-GPL) che è in grado di emulare oltre 7.000 giochi arcade, in modo che possa fungere da server di cloud gaming e comunicare con un front-end HTML, rimanendo sempre indipendente dal sistema operativo, così da rendere più agevole l'installazione di uno stand per il retro-gaming.

Indice

Ringraziamenti	ii
Sommario	iii
Introduzione	vi
1 Stato dell'arte	1
1.1 La nascita dei videogiochi	1
1.2 Cloud gaming	2
1.2.1 Tecnologie per lo streaming audio-video	4
1.2.2 Storia del cloud gaming	5
2 Architettura del sistema	11
2.1 Sistema proposto	11
2.2 MAME	12
2.2.1 Rendering	14
2.2.2 Missaggio audio	16
2.2.3 Gestione input	17
3 Implementazione	20
3.1 Cattura	20
3.1.1 Video	20
3.1.2 Audio	21
3.2 Codifica	21
3.2.1 MPEG	21
3.2.2 FFmpeg	22
3.3 Trasmissione	23
3.3.1 Web APIs	23
3.4 Decodifica	23
3.4.1 Web APIs	23
3.4.2 Librerie JavaScript	24
3.5 Gestione input	24
3.5.1 Librerie JavaScript	24
3.5.2 SDL input	24

4 Prestazioni	25
4.1 Qualità audio-video	25
4.1.1 Peak signal-to-noise ratio	26
4.1.2 Structural Similarity Index Method	26
4.2 Bit-rate	27
4.3 Latenza	28
Direzioni future di ricerca e conclusioni	29
A Listato	viii
A.1 Modifiche classi SDL	viii
A.1.1 Main	viii
A.1.2 Missaggio audio	ix
A.1.3 Rendering	ix
A.1.4 Gestione input	xi
A.2 Moduli nuovi	xi
A.2.1 Codifica	xi
A.2.2 Server	xv
B Manuale utente	xviii
B.1 Configurazione	xviii
B.2 Esecuzione	xviii
Bibliografia	xix
Sitografia	xx

Introduzione

Nel 1972 la società Atari pubblicava il primo videogioco della storia, Pong, vendendo 19.000 cabinati e presto molte altre società seguirono l'esempio. Alla fine del decennio iniziò l'epoca d'oro dei videogiochi arcade e la nascita delle console [William Acke et al. 2020]. I videogiochi uniscono narrativa, animazione e musica all'interattività, ed è grazie a quest'ultima che riescono ad esercitare un potenziale d'immersione e attrazione che gli altri media non hanno, tanto da diventare un fenomeno culturale di massa con centinaia di milioni di persone che giocano regolarmente ogni giorno, rendendoli attori dominanti nel settore dell'intrattenimento. L'importanza economica dei videogiochi arcade, negli ultimi vent'anni, è notevolmente diminuita a favore dei videogiochi per personal computer, console e più recentemente per mobile.

Il cloud computing è un paradigma a cui siamo ormai abituati e ci risulterebbe difficile abbandonare servizi come Dropbox, Office 365, Spotify e Netflix per tornare alle loro versioni "precedenti": i rullini fotografici, i documenti aziendali negli archivi, i CD audio ed i DVD a noleggio. Dalla nascita dei primi servizi di cloud computing nel 2006 alcuni oggetti sono stati sostituiti con la loro controparte informatica portando al fallimento di aziende storiche come Blockbuster (nel 2013), Kodak (nel 2012) e Borders¹ (nel 2011) [Andrea Pitzozzi 2019]. Il cloud computing consiste nella distribuzione on-demand delle risorse IT tramite internet su tre livelli di servizio che sono: l'accesso all'infrastruttura hardware tramite API (IaaS), la piattaforma software inclusa di sistemi di sviluppo (PaaS) e le applicazioni (SaaS). Con il cloud computing la potenza della macchina, sia essa fisica o virtuale, aumenta automaticamente all'esigenza permettendo di gestire i picchi di utilizzo; svincola gli utenti dal dover acquistare, manutenere e gestire fisicamente le infrastrutture IT; fornisce l'accesso alle risorse informatiche da qualsiasi device, da qualsiasi luogo e in modo collaborativo.

Unendo il paradigma del cloud computing con lo streaming nasce un nuovo tipo di servizio dedicato ai videogiochi: il cloud gaming. Questo servizio rende possibile giocare in remoto senza scaricare o installare il gioco sul device dell'utente. Con il cloud gaming i videogiochi sono archiviati ed eseguiti su un server remoto e l'output audio-video trasmesso al dispositivo dell'utente, permettendo di iniziare a giocare immediatamente, indipendentemente dal sistema operativo e dalle capacità hardware del dispositivo utilizzato. Infine, indirettamente, viene garantita la gestione dei diritti digitali (DRM) per gli editori. Il cloud gaming è l'unione di due modelli del cloud computing: il modello SaaS e il modello PaaS sia perché il videogioco viene offerto al giocatore come "applicazione" sia perché allo sviluppatore viene offerto il sistema operativo e i vari SDK, questo paradigma implementato dal cloud gaming è definito "gioco come servizio" (GaaS) che si divide in tre instanze: rendering remoto (RR-GaaS), rendering locale (LR-GaaS), allocazione delle risorse cognitive (CRA-GaaS) [D'Angelo, Ferretti e Marzolla 2015].

Lo scopo di questa tesi è creare una piattaforma di cloud gaming per far conoscere alle nuove generazioni i videogiochi che hanno fatto la storia e dare la possibilità di poter giocare ancora a macchine che ormai hanno cessato di funzionare per motivi di obsolescenza. Per far ciò verrà

¹Borders Group era un rivenditore americano di libri e musica.

ampliato il software MAME (rilasciato sotto licenza GNU-GPL) che è in grado di emulare oltre 7.000 giochi arcade, in modo che possa fungere da server di cloud gaming e comunicare con un front-end HTML, rimanendo sempre indipendente dal sistema operativo, rendendo più agevole l'installazione di uno stand per il retro-gaming.

Per la realizzazione del progetto sono state prese in considerazione le analisi fatte sulle piattaforme delle multinazionali come GeForce Now, Stadia e PlayStation Now in [Domenico et al. 2020] e [Maggiorini et al. 2016], come Amazon Luna in [Kyle Orland 2020], ma anche progetti open source come "Games on Demand" di [Karachristos, Apostolatos e Metafas 2008] che propone una piattaforma basata sul hooking di funzioni DirectX, codifica in MPEG2 e trasmissione tramite UDP; "GamingAnywhere" di [Huang et al. 2014] che è un progetto multipiattaforma che trasmette tramite protocollo RTP ed esegue la cattura audio-video utilizzando la libreria SDL tramite polling.

Il sistema proposto è stato progettato con un'ottica incentrata sull'utilizzo in LAN con l'utenza connessa tramite WiFi, ad esempio in stand di retro-gaming ad eventi di informatica e videogiochi, in aziende come servizio di svago per i clienti in sala d'attesa e per i dipendenti durante la pausa, ecc..., è importante ricordare che i videogiochi, nonostante siano stati pensati come fonte d'intrattenimento, migliorano diversi tipi di abilità chiave: abilità sociali e intellettuali, riflessi e concentrazione [Suznjevic e Homen 2020]; per questo motivo la piattaforma può essere installata anche nelle scuole.

Per ampliare il progetto MAME, lato server ho modificato le funzionalità di rendering video e missaggio audio per convogliare il loro output, che viene codificato in MPEG-TS, ad un modulo che esegue lo streaming tramite il protocollo WebSocket ad una pagina HTML. Lato client ho creato un modulo JavaScript per gestire l'input utente e decodificare il filmato MPEG-TS. La piattaforma utilizza un bit-rate tra 0.5 Mbps e 2.2 Mbps ed offre una risoluzione di 480p; per valutarne la latenza è stata testata su rete locale e su rete internet [Popovic et al. 2016]; mentre la qualità audio-video è stata classificata tramite "peak signal-to-noise ratio" e "structural similarity index method" [Shea et al. 2013].

La tesi è strutturata nel modo seguente:

- il capitolo 1 fornisce un'introduzione sulla nascita dei videogiochi e dei ricavi globali dell'industria videoludica, dà una definizione di cloud computing e di cloud gaming, fa una panoramica delle piattaforme di gioco che si sono susseguite nel tempo e delle proiezioni di mercato del settore del cloud gaming;
- nel capitolo 2 verrà descritto il sistema proposto, il MAME e le sue funzioni di: rendering, missaggio audio e gestione dell'input utente;
- nel capitolo 3 verranno descritte le cinque fasi aggiuntive del cloud gaming e la loro implementazione in C++ come nuovi moduli del MAME: la cattura audio-video, la codifica, la trasmissione, la decodifica e la gestione dell'input utente;
- il capitolo 4 analizza le prestazioni del progetto relativamente ai tre difetti intrinseci del cloud gaming: riduzione della qualità audio-video, bit-rate richiesto e il problema della latenza;
- nelle conclusioni si riassumono gli scopi, le valutazioni di questi e le prospettive future;
- nell'appendice A si riporta la documentazione del progetto logico. Il listato con l'autodocumentazione relativa è riportato nell'appendice B. Infine nell'appendice C si trova il manuale utente.

Capitolo 1

Stato dell'arte

Il capitolo che apre questa tesi fornisce un'introduzione sulla nascita dei videogiochi e dei ricavi globali dell'industria videoludica, dà una definizione di cloud computing e di cloud gaming, fà una panoramica delle piattaforme di gioco che si sono susseguite nel tempo e delle proiezioni di mercato del settore del cloud gaming.

1.1 La nascita dei videogiochi

Nel 1952 nei laboratori dell'Università di Cambridge, come esempio a corredo di una tesi di dottorato sull'interazione uomo-macchina, fu creato OXO, la trasposizione del tris come gioco per computer. OXO è considerato tecnicamente il primo videogioco. Nel 1958 un professore di fisica del Brookhaven National Laboratory creò un gioco, Tennis for Two, che aveva il compito di simulare le leggi fisiche relative ad una partita di tennis, lo strumento utilizzato era un oscilloscopio.

Nel 1961, sei giovani scienziati del Massachusetts Institute of Technology su un PDP-1¹ crearono il primo videogioco a scopo di intrattenimento: Spacewar!.

Due mesi dopo due ingegneri elettrici, N. Bushnell e T. Dabney, terminarono la loro versione di Spacewar! su larga scala (1.500 copie), ma il gioco non ebbe un grande successo a causa dell'elevata difficoltà. Bushnell, dopo l'esperimento non particolarmente riuscito, decise però di insistere nel settore dando così vita alla società Atari. Il primo gioco arcade di Atari fu il primo grande successo del settore: Pong. Pubblicato alla fine del 1972, è un gioco che riproduce approssimativamente la meccanica del ping pong. Atari vendette 19.000 cabinati di Pong e presto molte altre società seguirono l'esempio [William Acke et al. 2020]. Alla fine del decennio iniziò l'epoca d'oro dei videogiochi arcade e la nascita delle console (console che hanno fatto la storia in Fig. 1.1).



Figura 1.1: Console iconiche, fino alla nona generazione

¹PDP-1: Programmed Data Processor-1, era un computer della Digital Equipment Corporation del 1959.

I videogiochi sono un mezzo di intrattenimento unico che combina le diverse forme d’arte, quali musica, narrativa e animazione, all’interattività. Ed è proprio questa caratteristica, l’interattività, che permette loro di esercitare un potenziale d’immersione e attrazione che altri media non hanno. Sono ormai diventati un fenomeno culturale di massa con centinaia di milioni di persone che giocano regolarmente ogni giorno, il che li rende attori dominanti nel settore dell’intrattenimento, settore in continua crescita che non ha mai subito interruzioni nel corso degli anni come mostrato in Fig. 1.2. Negli ultimi vent’anni l’importanza economica dei videogiochi arcade è notevolmente diminuita² (in viola nella figura) a favore dei videogiochi per personal computer, console e più recentemente per mobile.

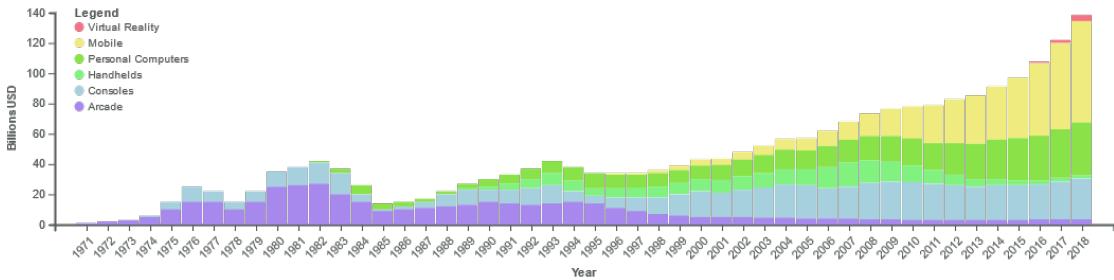


Figura 1.2: Ricavi globali dell’industria dei videogiochi dal 1971 al 2018 (non adeguati all’inflazione). Fonte: wikipedia.org

1.2 Cloud gaming

Negli ultimi anni sono apparsi molti tipi di servizi che sfruttano il paradigma del cloud computing: archiviazione (Dropbox, Drive, OneDrive), musica (Spotify, Amazon Music), cinematografia (Prime Video, Netflix), documenti (Google Workspace, Office 365) e più recentemente il cloud gaming. Il cloud gaming è un servizio che unisce il cloud computing e il live streaming per rendere possibile giocare in remoto senza scaricare o installare il gioco sul device dell’utente, in pratica consente di archiviare ed eseguire i videogiochi su un server remoto e trasmettere l’output audio-video all’utente sul proprio dispositivo.

Il cloud computing è un paradigma in cui un provider offre risorse fisiche e software accessibili da remoto, tramite la sottoscrizione di un abbonamento mensile/annuale oppure “pay-as-you go” (calcolato su: spazio di archiviazione utilizzato, tempo di utilizzo, cicli di CPU/GPU, ecc...), le cui caratteristiche essenziali sono:

- la capacità di fornire risorse hardware (processori, memoria, spazio di archiviazione, ecc...) automaticamente in base alle necessità software;
- l’accesso alle risorse attraverso la rete tramite protocolli standard;
- le risorse fisiche e virtuali possono essere distribuite dinamicamente agli utenti in base alle loro richieste;
- dal punto di vista dell’utente le risorse sono illimitate e possono essere acquistate in qualsiasi quantità ed in qualunque momento.

²Giappone, Cina e Corea mantengono una forte industria arcade ai giorni nostri.

- l'uso delle risorse e dei servizi è ottimizzato tramite il modello "pay-per-use" ed è monitorato e controllato in modo trasparente sia dal provider che dall'utente; alcune società offrono anche abbonamenti mensili e annuali, che invece limitano le risorse ad un tetto massimo.

Il cloud computing offre tre modelli di servizio: *Infrastructure as a Service* (IaaS), *Platform as a Service* (PaaS) e *Software as a Service* (SaaS). Il cloud gaming è l'unione del modello SaaS poiché i videogiochi sono offerti come un servizio dal punto di vista del giocatore, e del modello PaaS dal punto di vista dello sviluppatore che necessita del sistema operativo, delle librerie e dei kit di sviluppo; per cui il paradigma implementato dal cloud gaming è definito *Gaming as a Service* (GaaS) e si divide in tre istanze [D'Angelo, Ferretti e Marzolla 2015], come mostrato in Fig. 1.3, che sono:

- *Remote rendering* (RR-GaaS): il gioco viene eseguito e codificato sul server ed inviato all'utente come un filmato;
- *Local rendering* (LR-GaaS): il gioco viene eseguito, codificato sul server ed inviato all'utente sotto forma di istruzioni di rendering. Il client invia le istruzioni di rendering alla scheda grafica dell'utente;
- *Cognitive resource allocation* (CRA-GaaS): il client riceve moduli eseguibili del gioco che vengono eseguiti sul dispositivo dell'utente.

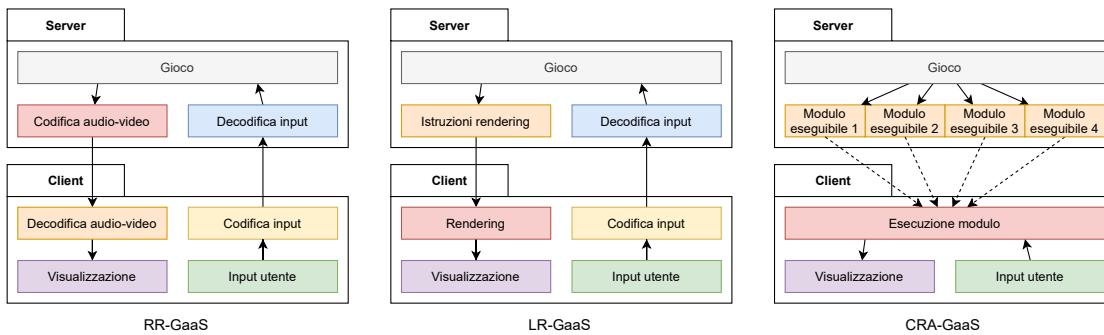


Figura 1.3: Instanze di GaaS

Il rendering remoto (RR-GaaS) è attualmente l'istanza più utilizzata nelle soluzioni di cloud gaming sul mercato essendo la soluzione che ha tutta la computazione a carico del server ed offre all'utente la possibilità di usare videogiochi computazionalmente esosi su qualsiasi tipo di device. Dei tre protocolli è però quello che richiede il bit-rate maggiore. Alcuni esempi sono Stadia³, PlayStation Now⁴.

Il rendering locale (LR-GaaS) è l'istanza che richiede il bit-rate minore ma, come nel caso dell'allocazione delle risorse, il rendering è eseguito sul device dell'utente, richiedendo così particolari caratteristiche hardware. Attualmente non ci sono importanti piattaforme sul mercato che offrono questo tipo d'istanza.

Alcune società propongono una versione dell'allocazione delle risorse cognitive (CRA-GaaS) mista al rendering remoto, in cui il gioco viene inizialmente servito come rendering remoto ed in contemporanea viene eseguito il download dei moduli eseguibili; al completamento del

³Stadia è una piattaforma di cloud gaming di Google.

⁴PlayStation Now è il cloud gaming targato Sony.

download dei moduli che riguardano l'attuale stato del gioco, lo streaming si interrompe e il gioco riprende dal device dell'utente; questa è una delle funzionalità che Project Atlas⁵ dovrebbe offrire. Un'altra variante, che viene chiamata commercialmente come "progressive download", si basa sul concetto di scaricare inizialmente i moduli principali del gioco, solitamente il menù e il primo livello; gli store Origin⁶ e Ubisoft Connect⁷ implementano questa funzionalità.

Nel prossimo paragrafo vedremo le tecnologie per lo streaming maggiormente usati.

1.2.1 Tecnologie per lo streaming audio-video

I protocolli di comunicazione a livello di trasporto su cui sono costruiti servizi, tecnologie e Web API [Mozilla 2021a], schematizzati in Fig. 1.4, sono UDP, TCP e SCTP, le cui caratteristiche principali sono riassunte in Tabella 1.1.

Caratteristica	UDP	TCP	SCTP
Dimensione header	8 byte	20-60 byte	12 byte
Entità del pacchetto	datagramma	segmento	datagramma
Orientato alla connessione	no	sì	sì
Trasporto affidabile	no	sì	sì
Consegna ordinata	no	sì	sì/no
Controllo del flusso	no	sì	sì
Controllo della congestione	no	sì	sì
Flussi multipli	no	no	sì
Multihoming ⁸	no	no	sì

Tabella 1.1: Comparazione tra protocolli di trasporto

Di questi solo quattro possono essere utilizzati per lo streaming utilizzando il browser web [Ilya Grigorik 2013]:

- Dynamic Adaptive Streaming over HTTP (DASH) è una tecnica di streaming con bit-rate adattivo del Moving Picture Experts Group (MPEG), che consente lo streaming di alta qualità di contenuti multimediali su protocollo HTTP;
- HTTP Live Streaming (HLS) è il protocollo di streaming ad alta latenza più popolare su HTTP per video on demand (video preregistrato) sviluppato da Apple;
- WebSocket è un protocollo di comunicazione che fornisce un canale full-duplex su una singola connessione TCP, con una latenza inferiore rispetto ad HLS e DASH;
- Web Real-Time Communication (WebRTC) è un progetto per la comunicazione in tempo reale basato sul protocollo RTP (Real-time Transport Protocol).

⁵Project Atlas è un progetto di una piattaforma cloud gaming mista ad intelligenza artificiale della Electronic Arts.

⁶Origin è una piattaforma di distribuzione digitale di videogiochi sviluppata da Electronic Arts.

⁷Ubisoft Connect è un servizio di distribuzione digitale della società Ubisoft.

⁸Il multihoming è la funzionalità di poter connettere un host a più di una rete.

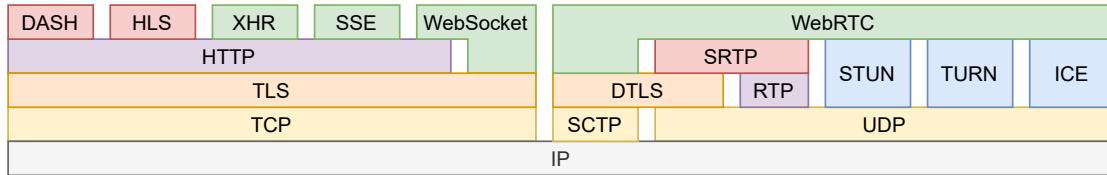


Figura 1.4: Streaming con perdita di pacchetti dell'8%.

Nel paradigma del cloud gaming la codifica audio-video avviene in real-time e il buffering non è usabile perché aumenterebbe la latenza, per questi motivi non tutte le tecnologie citate sono adatte. Come vedremo nelle caratteristiche delle piattaforme di streaming commerciali, descritte nel prossimo paragrafo, solo TCP, UDP ed RTP sono stati effettivamente usati. Solitamente la comunicazione tramite TCP è relegata alla parte di autenticazione e gestione dell'input utente. Infatti da test condotti in [Bielievtsov et al. 2018] sul frame rate in caso di perdita di pacchetti durante lo streaming, in Fig. 1.5 con una perdita di pacchetti dell'8%, si nota che con RTSP⁹, schema (a) e (b), il frame rate è costante ma con alcune perturbazioni; invece HLS, schema (c), risulta essere la tecnologia che risente maggiormente della perdita dei pacchetti; mentre con RTMP¹⁰, schema (d), la riproduzione si interrompe e il video si blocca per alcuni secondi.

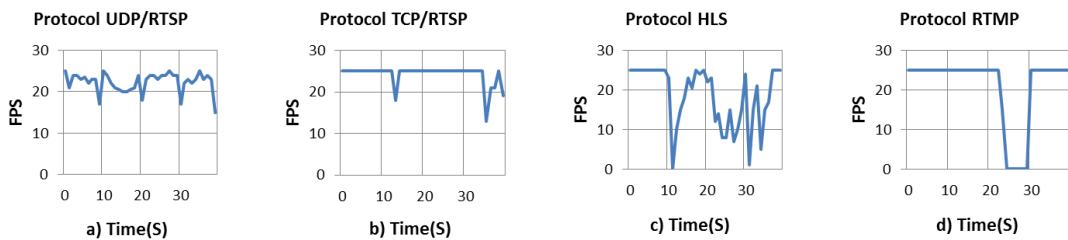


Figura 1.5: API, protocolli e servizi di rete del browser

Nel prossimo paragrafo viene fatta un'introduzione, in ordine cronologico, delle piattaforme di cloud gaming sul mercato.

1.2.2 Storia del cloud gaming

Una delle prime piattaforme di cloud gaming è stata OnLive di OL2, presentata alla GDC¹¹ 2009 e poi lanciata sul mercato a giugno 2010 negli Stati Uniti e a settembre 2011 nel Regno Unito. I giocatori, previo pagamento di un abbonamento mensile di 15\$, potevano acquistare o noleggiare giochi sulla piattaforma oppure utilizzare quelli precedentemente acquistati su Steam¹². Il servizio era ospitato su 5 data center situati sul suolo americano che servivano gli utenti per vicinanza geografica; il bit-rate richiesto era di 1,5 Mbps per la qualità video SD¹³ e 5 Mbps per la Stan-

⁹RTSP: Real Time Streaming Protocol serve a stabilire e gestire la sessione di streaming, la trasmissione dei dati avviene tramite RTP e la raccolta delle statistiche sulla qualità del servizio è affidata al Real-time Transport Control Protocol (RTCP).

¹⁰RTMP: Real Time Messaging Protocol era un protocollo sviluppato da Macromedia per il Flash player, ad oggi non più supportato dai browser.

¹¹GDC: Game Developers Conference, una conferenza annuale per gli sviluppatori di videogiochi.

¹²Steam è un servizio di distribuzione digitale di videogiochi della società Valve.

¹³SD: Standard Definition include i formati video con rapporto 4:3 e 16:9 con 480 linee di risoluzione (in alcuni casi 576).

dard HD¹⁴. I protocolli di comunicazione utilizzati erano: TCP per il test della velocità verso i 5 data center, successivamente tramite TLS su TCP veniva fatta l'autenticazione al servizio; l'input utente veniva inviato tramite UDP e lo streaming, codificato tramite H.264¹⁵, trasmesso tramite protocollo RTP [Manzano et al. 2014]. Era disponibile, oltre ad un client per Windows, macOS ed Android, anche una micro console da collegare alla TV. Il servizio fu acquistato ad aprile 2015 da Sony [JP Mangalindan 2020].

Al GDC 2010 Gaikai ha presentato il suo omonimo servizio di cloud gaming; A febbraio 2012 era distribuito su 24 data center e disponibile in 12 nazioni. La società si è concentrata principalmente su due modelli di business: sull'utilizzo del cloud gaming come forma di pubblicità online per i videogiochi, fornendo agli utenti la possibilità di accedere alle demo dei videogiochi sponsorizzati; fornire ad altre aziende l'infrastruttura per il cloud gaming, come per Wikipad¹⁶, Electronic Arts e Samsung [Gaikai 2012]. La piattaforma era accessibile tramite browser (utilizzando il plugin di streaming disponibile in Adobe Flash, Java o Google Native Client), la qualità video HD richiedeva un bit-rate minimo di 5 Mbps. A luglio 2012 la società fu acquisita da Sony per integrare la loro tecnologia di streaming nella piattaforma di cloud gaming PlayStation Now [Sean Hollister 2010].

Beijing Cloud Union al CES¹⁷ 2012 ha presentato il suo omonimo servizio di cloud gaming con 64 videogiochi tra cui alcuni titoli per PlayStation 3, accessibile tramite browser sul PC e tramite app per smartphone. Il servizio era disponibile solo in Cina ed arrivò a 20 milioni di utenti chiudendo definitivamente a dicembre 2018 [Beijing Cloud Union 2018].

PlayStation Now è un servizio di cloud gaming basato sulla tecnologia cloud di Gaikai. È stato presentato durante il CES 2014 ed è stato reso disponibile a partire da gennaio 2015 in Nord America, da settembre in Giappone e Regno Unito ed ha iniziato a coprire il mercato europeo gradualmente a partire da agosto 2017. La piattaforma consente all'utente di giocare ai titoli PlayStation (attualmente 800 dal catalogo giochi della PS2, PS3 e PS4) su PS4, PS5 e Windows (tramite l'installazione del client "PS Now app" e l'uso di un controller compatibile). Per quanto riguarda i giochi PS2 e PS3 Sony ha costruito una scheda madre contenente la componentistica miniaturizzata di otto PS3, ognuna indipendentemente controllabile, inoltre l'hardware contiene un codificatore video H.264; lo stesso è avvenuto per i giochi PS4, utilizzando schede madri PS4 modificate ad hoc [Ryan Whitwam 2014]. Questo ha permesso di ridurre ulteriormente la latenza di cattura e codifica (argomento trattato nel Paragrafo 4.3). La risoluzione offerta è Standard HD con un bit-rate minimo richiesto di 5 Mbps e la tecnologia di trasmissione è basata su UDP [Domenico et al. 2020]. Gli abbonamenti proposti sono da 10\$, 25\$ o 60\$ per 1, 3 o 12 mesi di utilizzo [Sony 2021].

GeForce Now è il servizio di cloud gaming di Nvidia lanciato in beta a gennaio 2017 e ufficialmente a febbraio 2020. I data center sono collocati negli Stati Uniti, Europa, Australia e Canada, e sono dotati di GPU Nvidia P40. GeForce Now consente agli utenti di accedere da remoto (tramite streaming) a un computer virtuale, dove possono installare giochi (attualmente il catalogo consta di 800 giochi) acquistati su Steam, Ubisoft Connect o Epic Games Store¹⁸. Il servizio può essere utilizzato su Windows, macOS, iOS, Android o Nvidia Shield TV¹⁹. Nvidia ha scelto di usare RTP come protocollo per lo streaming utilizzando il codec H.264, UDP per il test della velocità e l'invio dell'input dell'utente e TLS tramite TCP per l'autenticazione [Domenico

¹⁴Standard High Definition (chiamata anche HD ready) è un formato video 16:9 (disponibile anche con rapporto 4:3) con 720 linee di risoluzione.

¹⁵H.264 è un formato di compressione video dello standard MPEG-4.

¹⁶Oggi Gamevice, è un produttore di tablet e periferiche specializzato in prodotti per il gaming.

¹⁷CES: Consumer Electronics Show è un evento annuale che ospita presentazioni di nuovi prodotti e tecnologie nel settore dell'elettronica di consumo.

¹⁸Epic Games Store è un negozio di videogiochi digitali gestito da Epic Games.

¹⁹Nvidia Shield TV è un lettore multimediale digitale basato su Android.

et al. 2020]. L'abbonamento mensile è di 10\$ e le risoluzioni video offerte sono Standard HD e Full HD²⁰ con, rispettivamente, 15 e 25 Mbps di bit-rate richiesto [Nvidia 2021].

A maggio 2018 Electronic Arts ha svelato Project Atlas che va ad espandere il concetto di cloud gaming. La piattaforma mira a fornire un'esperienza di gioco nuova grazie al supporto dell'intelligenza artificiale, offrendo universi di gioco che cambiano con il passare del tempo, con l'interazione con altri giocatori e sotto l'influenza del mondo esterno. Dal punto di vista degli sviluppatori il progetto punta a far confluire il motore di gioco Frostbite²¹, i servizi di gioco e l'intelligenza artificiale in una nuova piattaforma di sviluppo [Electronic Arts 2018].

Vortex della RemoteMyApp è un servizio di cloud gaming lanciato a novembre 2018, è disponibile per Android, Windows e macOS e offre tre piani mensili (12\$, 23\$ e 34\$) che consentono all'utente di giocare per un massimo di 140 ore al mese ad un catalogo di 170 giochi. Sfortunatamente, alcuni giochi possono essere riprodotti solo acquistando la licenza del gioco. La società utilizza 13 data center con: processori Intel Xeon, 512GB di memoria RAM e schede grafiche NVIDIA. Il bit-rate richiesto è 10 Mbps per giocare in Standard HD [RemoteMyApp 2021].

Microsoft ha anticipato Xbox Cloud Gaming all'E3²² 2018. La piattaforma è disponibile per gli abbonati a Xbox Game Pass Ultimate da settembre 2020 ed offre sia la libreria esistente di giochi per Xbox che per Xbox Series X (attualmente una selezione di 250 giochi). La piattaforma è ospitata su 54 data center Azure²³ che coprono 140 paesi [Adam Bankhurst 2018] mentre l'hardware è basato su schede madri Xbox Series X ridisegnate ad hoc [Tom Warren 2020b]. Il servizio è progettato per funzionare con gli smartphone (attualmente solo Android), con controlli touchscreen o usando un controller Bluetooth compatibile, ad una risoluzione Standard HD ed è richiesto un bit-rate di 10 Mbps, l'abbonamento ha un costo mensile di 10\$ [Microsoft 2021].

Google Stadia è una piattaforma di cloud gaming rilasciata a novembre 2019, ma è disponibile solo in Europa e negli Stati Uniti. Il servizio è ospitato sui data center Google che montano GPU personalizzate di AMD [Google 2019a] con supporto alle API Vulkan²⁴ in grado di sviluppare una potenza di oltre 10 teraflops [Google 2019b]. La piattaforma è eseguita su una versione personalizzata di Debian²⁵ e l'utilizzo dello Stadia SDK²⁶ è necessario per gli sviluppatori. Stadia è stata la prima piattaforma a sfruttare completamente WebRTC che fornisce: l'autenticazione tramite DTLS²⁷ e STUN²⁸, l'invio dell'input utente tramite DTLS e lo streaming tramite RTP, utilizzando uno dei seguenti codec video: AV1²⁹, VP9³⁰ e H.264 [Domenico et al. 2020]. Il costo mensile del servizio è di 10\$ ed è accessibile tramite app su Android, tramite web app su iOS, su Chromecast³¹ utilizzando il controller Stadia³² e su computer tramite browser Chrome si può giocare con mouse e tastiera oppure usando un controller compatibile. Le risoluzioni video offerte sono Standard HD, Full HD e UHD³³, con bit-rate richiesti di 10, 25 e 35 Mbps. La piattaforma offre le seguenti funzionalità: live streaming su YouTube del proprio gameplay; "Crowd Play" che consente agli spettatori di unirsi ad una sessione di gioco in live stream (se invitati dall'host); "Stream Connect" che consente all'utente di condividere la schermata di gioco con altri giocatori

²⁰Full HD è un formato video 16:9 con 1080 linee di risoluzione.

²¹Frostbite è un motore di gioco sviluppato da DICE, una società sussidiaria di Electronic Arts.

²²E3: Electronic Entertainment Expo, un evento commerciale per l'industria dei videogiochi.

²³Microsoft Azure è una piattaforma di cloud computing.

²⁴Vulkan è un API per la grafica real-time 3D.

²⁵Debian è una distribuzione Linux composta interamente da software libero.

²⁶Il progetto Stadia è rilasciato sotto licenza GPL 2.0 ed è disponibile su Github.

²⁷DTLS è un protocollo crittografico per UDP basato su TLS.

²⁸STUN è un protocollo per il NAT traversal per comunicazioni in tempo reale.

²⁹AV1 è un formato di codifica video open-source della Alliance for Open Media.

³⁰Google VP9 è un formato di codifica video open-source.

³¹Google Chromecast è un lettore multimediale digitale per contenuti audiovisivi in streaming su Internet.

³²Controller WiFi di Google con connessione diretta a Stadia.

³³UHD: Ultra High Definition è un formato video 16:9 con 2160 linee di risoluzione.

nella stessa partita; "Condivisione dello stato" che consente di condividere il proprio salvataggio di gioco con gli amici [Google 2021].

Amazon Luna è stata annunciata a settembre 2020, con "accesso anticipato" a partire da ottobre 2020. Per motivi di compatibilità il S.O. scelto per la piattaforma è Windows ed è in esecuzione su istanze EC2 G4³⁴ in grado di sviluppare una potenza di 8,1 teraflops [Tom Warren 2020a]. Il catalogo giochi proposto consta di più di 100 giochi ed il servizio offre l'integrazione con Twitch ed una partnership con Ubisoft che da accesso ai loro titoli al momento del rilascio. Si può accedere alla piattaforma utilizzando tramite PC, Fire TV³⁵ e smartphone, utilizzando controller Luna³⁶, Xbox o PS. Come Stadia anche Luna utilizza WebRTC per l'autenticazione, la gestione dell'input utente e lo streaming [Kyle Orland 2020]. L'abbonamento è di 6\$ al mese (15\$ per la versione in partnership con Ubisoft) con una risoluzione Full HD e una banda richiesta di 10 Mbps [Amazon 2021].

La società Playkey ha realizzato un omonima piattaforma di cloud gaming distribuito, attualmente in alpha testing. Il sistema distribuito è formato da un server centrale che gestisce l'infrastruttura e dai computer dei cosiddetti "minatori", coloro che mettono a disposizione il proprio computer come unità di calcolo del sistema distribuito, su cui viene eseguito il gioco, la codifica e lo stream tramite protocollo UDP. La piattaforma offre la Standard HD come risoluzione ed è richiesto un bit-rate di 10 Mbps. L'abbonamento è di 23\$ mensili mentre i "minatori" guadagnano 10\$ al giorno. Playkey da la possibilità di giocare solo i titoli precedentemente acquistati da Steam, Ubisoft Connect, Origin e Battle.net³⁷ [Playkey 2021].

Il caso Apple

A metà del 2020 Apple aveva cercato di bloccare le app di cloud gaming sull'App Store, ma a settembre 2020 decise di consentire il cloud gaming con alcune restrizioni: che i giochi offerti nel servizio dovessero essere scaricati direttamente dall'App Store e non da un'app all-in-one. I produttori di app sono autorizzati a rilasciare una cosiddetta "app catalogo" che si collega ad altri giochi nel servizio, ma ogni gioco dovrà essere una singola app e tutti i giochi e le "app catalogo" devono offrire l'acquisto solo tramite il sistema di elaborazione dei pagamenti "in-app purchases" di Apple, in base al quale la Apple ha un guadagno del 30% sugli acquisti fatti dall'utente [Kif Leswing 2020].

Progetti a scopo didattico

Per la realizzazione di questo progetto sono stati presi in considerazione due progetti di cloud gaming a scopo didattico.

"Games on Demand" di [Karachristos, Apostolatos e Metafas 2008] è una piattaforma del 2007 per Windows basata sul hooking di funzioni DirectX tramite la libreria Taksi³⁸. Lo streaming avviene tramite UDP utilizzando una codifica video in MPEG2, mentre l'audio viene omesso.

GamingAnywhere di [Huang et al. 2014] è un progetto multipiattaforma del 2013 per Windows, Linux, macOS ed Android. La cattura audio-video avviene utilizzando la libreria SDL tramite polling settando un frame-rate per il video e uno per l'audio; se la velocità di aggiornamento dell'output video è maggiore del frame-rate alcuni frame intermedi vengono scartati;

³⁴EC2 G4 è una istanza del cloud computing di Amazon progettata per il rendering e il machine learning. Le schede video installate sono le Nvidia T4.

³⁵Fire TV è una linea di media center di Amazon.

³⁶Controller WiFi di Amazon con connessione diretta a Luna.

³⁷Battle.net è una piattaforma di distribuzione digitale e di gestione dei diritti digitali sviluppata da Blizzard Entertainment.

³⁸Taksi è una libreria open-source per la cattura video di applicazioni che utilizzano DirectX, OpenGL o GDI.

invece la cattura audio ha un altro problema, nel caso in cui il gioco non emetta nessun suono, il programma deve generare dei frame audio di silenzio. Questi due problemi in MAME CGP³⁹ sono stati affrontati in modo differente e verranno illustrati nel paragrafo 3.1.2. Lo streaming avviene tramite protocollo RTP, utilizzando la libreria LIVE555⁴⁰; la codifica usata è VP8 con una risoluzione video Standard HD ed un bit-rate richiesto di 3 Mbps.

In Tabella 1.2 è riportato il riepilogo delle piattaforme disponibili ad oggi con caratteristiche, requisiti e protocollo di streaming utilizzato.

Piattaforma	Tipo	Risoluzione ⁴¹	Bit-rate ⁴²	Protocollo
Amazon Luna	Cloud gaming	1080	10	RTP
Games on Demand	Cloud gaming	?	?	UDP
GamingAnywhere	Cloud gaming	720	3	RTP
GeForce Now	Computer virtuale	1080	25	RTP
Google Stadia	Cloud gaming	2160	35	RTP
MAME CGP	Cloud gaming	480	2	WebSocket
Playkey	Computer virtuale	720	10	UDP
PlayStation Now	Cloud gaming	720	5	UDP
Vortex	Cloud gaming	720	10	UDP
Xbox Cloud Gaming	Cloud gaming	720	10	UDP

Tabella 1.2: Piattaforme disponibili

Nel prossimo paragrafo verranno illustrate le proiezioni di mercato relative al 2023.

Proiezioni di mercato

Secondo una ricerca di Newzoo⁴³ sull’industria dei videogiochi [Newzoo 2020], come mostrato in Fig. 1.6, nel 2020 il mercato del cloud gaming ha generato quasi 584,7 milioni di USD di entrate, di cui il 39% e il 29% in Nord America e in Europa, e si prevede una crescita fino a 4,8 miliardi di USD entro il 2023, se non maggiore. Per questo sono entrate nel mercato del cloud gaming anche aziende che non sono editrici o produttrici di videogiochi come Google e Amazon, come vedremo nel paragrafo 1.2.2.

Queste previsioni sono conseguenza sia di connessioni di rete, domestiche e mobili, sempre più veloci sia del paradigma del live streaming a cui oggi (soprattutto le nuove generazioni) siamo abituati. Dal punto di vista del giocatore il cloud gaming offre molti vantaggi tra cui: rendere il gioco facilmente accessibile senza la necessità di scaricarlo e installarlo localmente; compatibilità con computer, smartphone e anche con smart TV (se utilizzato con un gamepad WiFi) senza dover badare ai requisiti hardware; diverse modalità di pagamento tra cui l’acquisto di un gioco su richiesta e l’abbonamento mensile/annuale per l’utilizzo di tutta (o una parte) della libreria videoludica; funzionalità aggiuntive per sfruttare al meglio questo modello, come lo streaming della sessione di gioco, che siamo già abituati a vedere, con la possibilità di far entrare uno spettatore nella propria partita, funzionalità avanzate per il multiplayer come la condivisione della visuale di gioco e dei salvataggi di gioco, ecc.... Ci sono vantaggi anche per gli sviluppatori

³⁹MAME CGP (Cloud Gaming Platform) è il nome che ho dato a questa versione modificata del MAME in grado di fungere da piattaforma di cloud gaming.

⁴⁰LIVE555 Streaming Media è un set di librerie open-source per lo streaming multimediale.

⁴¹Risoluzione video misurata in linee verticali.

⁴²Bit-rate misurato in Mbps.

⁴³Società di analisi statistica del settore videoludico.

perché il cloud gaming riduce i costi di produzione limitando lo sviluppo e il testing ad una sola piattaforma ed inoltre risolve definitivamente un problema che esiste dai tempi delle audiocassette e dei floppy disk, la pirateria. Infine per i fornitori di servizi si viene a creare un nuovo modello di business su contenuti già esistenti. Tuttavia ci sono anche degli svantaggi, di cui parleremo nel capitolo 4: perdita della qualità audio-video a causa della compressione, bit-rate richiesto non soddisfacile dall'utente e l'ineliminabile latenza.

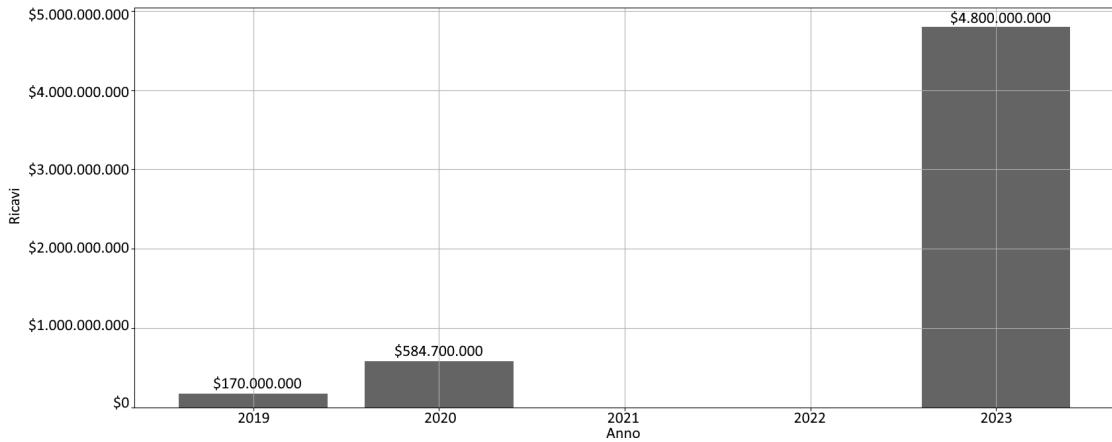


Figura 1.6: Previsioni per il mercato globale del cloud gaming (in dollari americani). Fonte: newzoo.com/global-cloud-gaming-report

Capitolo 2

Architettura del sistema

In questo capitolo verrà descritto il sistema proposto, il MAME e le sue funzioni di rendering, missaggio audio e gestione dell'input utente.

2.1 Sistema proposto

L'esigenza per la quale nasce questo progetto è far conoscere alle nuove generazioni i videogiochi che hanno fatto la storia e dare la possibilità di poter giocare ancora a macchine che ormai hanno cessato di funzionare per motivi di obsolescenza, sfruttando due tecnologie entrate a far parte della quotidianità, lo streaming e il cloud computing. In questo lavoro si propone la creazione di una piattaforma di cloud gaming, che permette lo streaming audio-video direttamente e su richiesta dei videogiochi, da un server remoto, ad un client (computer, console, telefono). Per far ciò verrà ampliato il software MAME (rilasciato sotto licenza GNU-GPL) che è in grado di emulare oltre 7.000 giochi arcade. Le caratteristiche principali del progetto, che sono state vincolanti nella scelta delle tecnologie da utilizzare, sono la portabilità e la possibilità di utilizzare il sistema senza dover installare software aggiuntivi; per questi vincoli, lato client, la scelta è ricaduta sul browser web.

Il sistema è stato progettato con un'ottica incentrata sull'utilizzo in LAN con l'utenza connessa tramite WiFi, ad esempio in stand di retro-gaming ad eventi di informatica e videogiochi, in aziende come servizio di svago per i clienti in sala d'attesa e per i dipendenti durante la pausa, nelle scuole, ecc...; infatti nonostante siano stati pensati come fonte d'intrattenimento, i videogiochi migliorano diversi tipi di abilità chiave: abilità sociali e intellettuali, riflessi e concentrazione [Suznjevic e Homen 2020]. La tecnologia di streaming scelta è stata WebSocket poiché in questo contesto la differenza di velocità tra TCP e RTP può essere trascurata, è un protocollo di comunicazione standardizzato dal 2011, è pienamente supportato da tutti i browser moderni, ha una latenza inferiore rispetto ad HLS e DASH, è semplice da instanziare e non richiede l'utilizzo di protocolli aggiuntivi o configurazioni complesse a differenza di WebRTC.

Come mostrato in Fig. 2.1 il sistema è costituito dal server di gioco (Linux, macOS o Windows), su cui è installato il MAME CGP con le rom dei giochi ed una pagina HTML5 che funge da front-end. Il programma è in ascolto per connessioni WebSocket con parametri (per es.: il nome del gioco, l'ID del player, l'ID della partita, ecc...). Una volta stabilita la connessione, il server invia informazioni sulla risoluzione di rendering e avvia il gioco. Il rendering e il missaggio audio

del gioco vengono generati utilizzando la libreria SDL, codificati e pacchettizzati nel contenitore MPEG-TS¹ usando la libreria FFmpeg² e inviati tramite WebSocket al client.

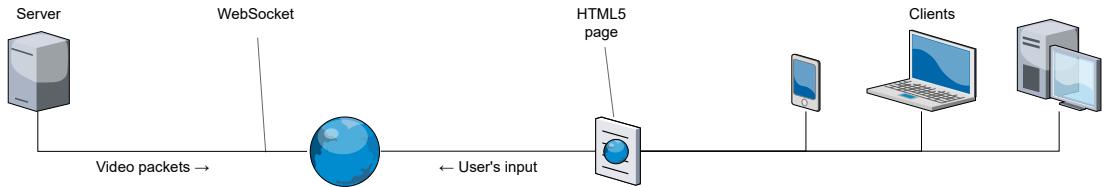


Figura 2.1: Panoramica del sistema

Lato client vari script si occupano di decodificare i dati audio-video ricevuti, catturare e inviare l'input dell'utente (sia dalla tastiera che dal gamepad) al server tramite WebSocket.

Nel prossima sezione introdurremo il software MAME su cui si basa questo progetto.

2.2 MAME

Multiple Arcade Machine Emulator (MAME) è un progetto open-source (GNU-GPL) di Nicola Salmoria. La prima versione del MAME risale al febbraio 1997 ed è attualmente supportato da una vasta comunità di sviluppatori in tutto il mondo. Il suo scopo principale è quello di essere un riferimento al funzionamento interno delle macchine emulate, sia per scopi educativi che per scopi di conservazione, al fine di evitare che il software storico scompaia per motivi di obsolescenza. Il progetto MAME è stato realizzato in C e C++, inizialmente usando solamente la libreria standard e poi successivamente, negli anni, sono state aggiunte al progetto varie librerie open-source per estenderne le funzionalità. Originariamente era disponibile solo per MS-DOS ma grazie alla vasta comunità di sviluppatori è stato compilato anche per i sistemi Windows e Unix-like [MAME Team 2021]. Logicalmente è suddiviso in quattro macro categorie, come mostrato in Fig. 2.2:

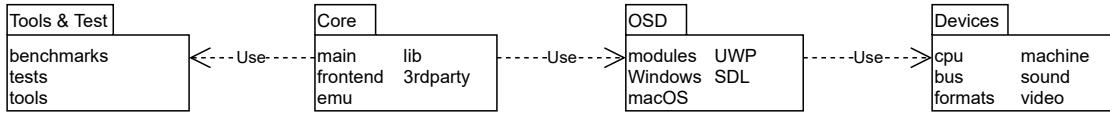


Figura 2.2: MAME, diagramma dei packages

- "Core" in cui ci sono i sotto-progetti indipendenti dal sistema e dal device emulato tra cui il programma principale (progetto main), il front-end grafico, il motore di emulazione (progetto emu), le librerie comuni (lib) ed i sorgenti delle librerie esterne (3rdparty);
- "OSD" contenente le funzionalità dipendenti dal sistema operativo tra cui macOS, Windows, UWP³ e SDLMAME, ed i moduli (progetto modules) di input, audio e video dipendenti da altre librerie come OpenGL, DirectX, SDL, CoreAudio, XAudio, XInput, ecc...;

¹MPEG-TS: MPEG transport stream, è un contenitore digitale per la trasmissione e l'archiviazione audio-video.

²FFmpeg è una suite open-source di librerie e programmi per la gestione di video, audio, e altri file multimediali e stream.

³UWP: Universal Windows Platform è un'architettura applicativa della Microsoft per sviluppare applicazioni eseguibili su Windows 10, Xbox One e Hololens.

- "Devices" che contiene per ogni device emulato (ad esempio il Capcom CP System III) le classi che gestiscono le informazioni della macchina ed emulano cpu, bus, schede video, schede audio e i dischi (progetto formats);
- "Tools & Test" che contiene varie utility per la gestione delle rom e per la fase di testing e performance.

Il progetto ha una struttura modulare, schematizzata in Fig. 2.3, formata da un nucleo centrale che dirige le operazioni, gestisce l'interfaccia utente e mette a disposizione dei driver un buon numero di funzioni d'uso comune. Il nucleo delega a moduli esterni l'emulazione dei vari tipi di CPU e di chip audio supportati. In pratica il nucleo fornisce un ambiente operativo specializzato nell'emulazione di videogiochi arcade, che i driver possono sfruttare con poco codice aggiuntivo. Spesso la maggior parte del contenuto di un driver è costituita da strutture dati, gestite direttamente dal nucleo. Il driver deve fornire codice solo per alcuni compiti specifici, ad esempio l'aggiornamento del video [Nicola Salmoria 2002].

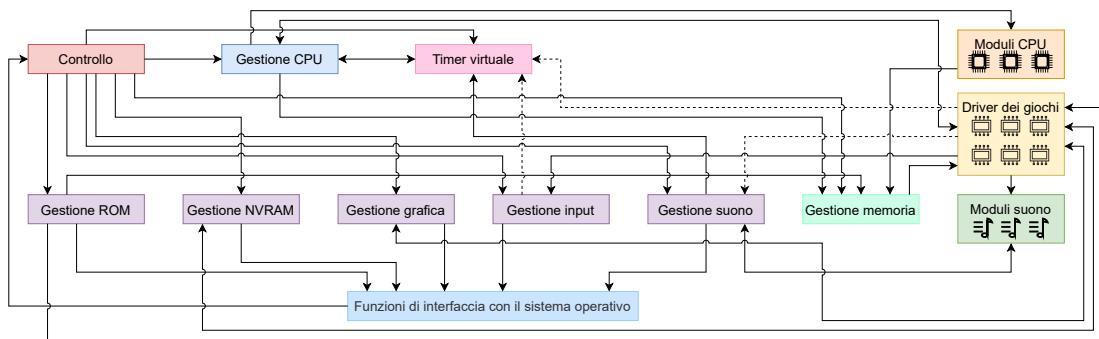


Figura 2.3: Schema della struttura del MAME

Per quanto riguarda la portabilità, di cui c'è uno schema in Fig. 2.4, ci sono solo tre compilazioni native differenti e sono quella per macOS, Windows ed UWP. In aggiunta c'è la compilazione SDLMAME⁴ in grado di funzionare su tutti i sistemi operativi supportati dalla libreria SDL. Quest'ultima è la compilazione obbligatoria per i sistemi Linux e per questo motivo è quella che si è scelta di utilizzare per questo server di cloud gaming.

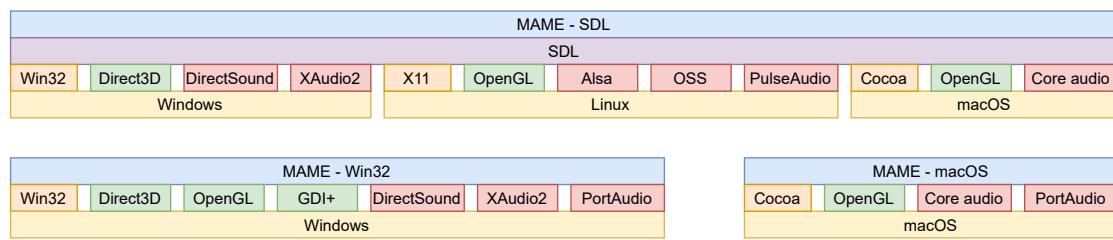


Figura 2.4: Librerie e interfacce utilizzate dal MAME sulle diverse piattaforme

SDL (Simple DirectMedia Layer) è una libreria multipiattaforma che fornisce accesso di basso livello ad audio, tastiera, mouse, gamepad, hardware 3D e framebuffer 2D. Come mostrato in

⁴SDLMAME era un port del MAME che utilizzava solamente la libreria SDL. Nel 2010 è stato incluso ufficialmente nel progetto MAME.

Fig. 2.4 nello schema in alto, SDL è costruito sopra le API di visualizzazione video del sistema operativo (in arancione), le librerie di rendering (in verde) e le librerie che si interfacciano alla scheda audio (in rosso) [SDL Community 2020].

Nel prossimi tre paragrafi verrano descritti i tre moduli su cui sono state apportate le modifiche per trasformare il MAME in una piattaforma di cloud gaming.

2.2.1 Rendering

Il rendering è il processo di generazione di un'immagine a partire dalla sua descrizione, che può essere in due o tre dimensioni, tramite un software. Gli elementi di base del rendering 2D sono le textures⁵ e le animazioni. Il motore grafico (il software che si occupa anche del rendering) prende gli elementi uno ad uno e li disegna nel framebuffer⁶ generando l'immagine finale [Mileff e Dudra 2012]. Come mostrato in Fig. 2.5, questo processo è formato da quattro fasi [Forsyth et al. 2015]:

1. trasformazione di modellazione: tramite una trasformazione trasporta le primitive geometriche in un sistema di coordinate universali (*world coordinate system*);
2. clipping: ritaglia porzioni delle primitive al di fuori della finestra di visualizzazione;
3. trasformazione di vista: tramite una trasformazione ridiga detta "trasformazione di vista" trasporta le primitive ritagliate dalle coordinate universali a quelle schermo (*screen coordinate*);
4. *scan conversion*: attraverso algoritmi di rasterizzazione⁷ genera l'immagine finale.

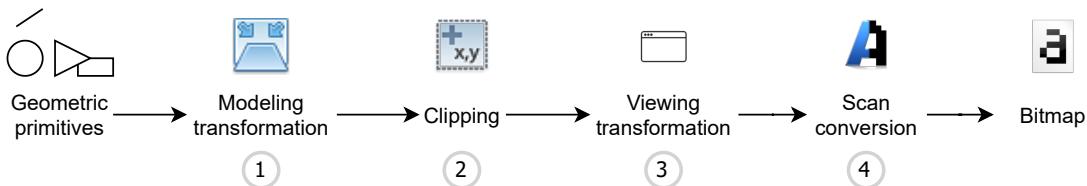


Figura 2.5: Pipeline di rendering 2D

Il MAME è in grado di emulare giochi sia 2D che 3D (ad es.: Tekken della Namco) ma sia per la volontà di emulare fedelmente l'hardware della macchina sia perché le varie schede ed API grafiche delle macchine emulate non lavorano esattamente come quelle moderne (ad es.: i poligoni della mesh utilizzano i rettangoli al posto dei triangoli) tutta la fase di rendering viene eseguita via software, per questo motivo ciò che viene inviato alla libreria grafica è un insieme di primitive e texture da disegnare sia nel caso di giochi 2D che 3D come spiegato nella FAQ ufficiale nel capitolo sulle performance: «*There are many things that are difficult to emulate without expending a large amount of CPU power. Some specific examples are: Games with 3D graphics. As of this writing, MAME does not pass polygons down to the video card of your*

⁵Una texture è un'immagine rappresentata come una matrice bidimensionale di pixel colorati (in inglese bitmap). Ogni pixel è rappresentato tramite una quaterna formata dai tre colori primari più un valore che ne indica la trasparenza (RGB + A); servono 4 bit per memorizzare ogni elemento della quaterna, quindi 32 in totale per un singolo pixel.

⁶Il framebuffer è uno spazio di memoria presente sulla scheda video in cui si memorizza l'immagine che verrà successivamente mostrata a video.

⁷La rasterizzazione è il processo di approssimazione delle primitive geometriche in immagini bitmap.

system; instead, it renders all 3D graphics by hand in software. Although this code is generally optimized to take advantage of multiple CPUs, it is still quite taxing to do this. Some 3D games give you control of the output resolution; reducing it will reduce the CPU requirements.» [MAME Team 2018].

La UI del MAME viene renderizzata con le stesse procedure utilizzate per l'emulazione per cui è possibile effettuarne lo streaming ed utilizzarla al posto del front-end HTML.

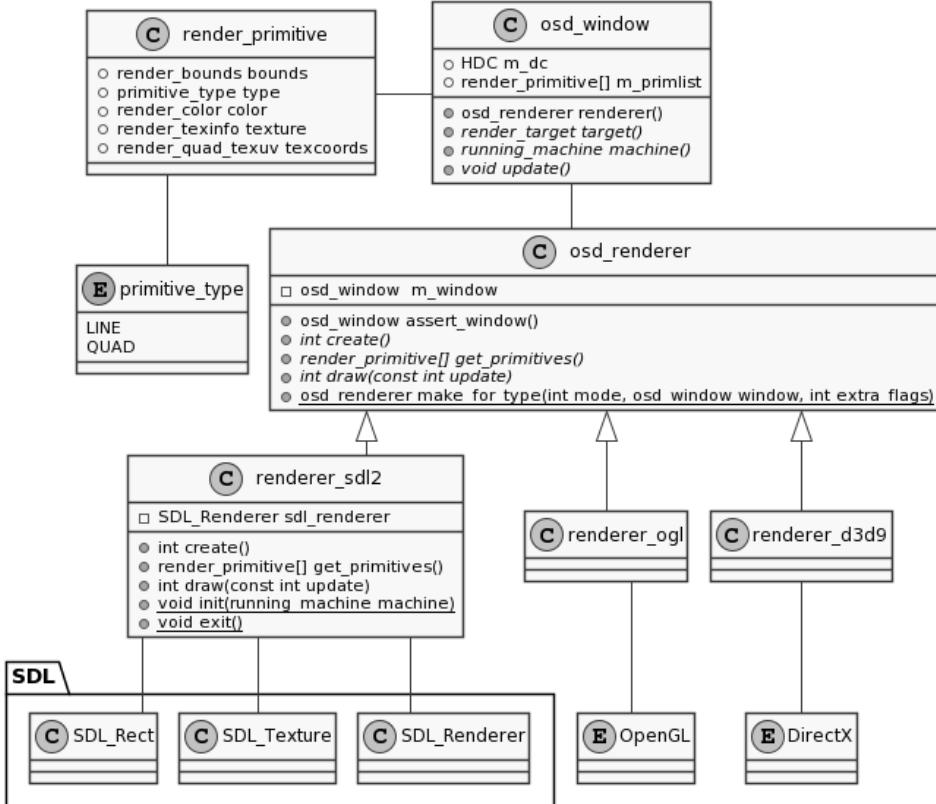


Figura 2.6: Diagramma delle classi relative al rendering

Come detto precedentemente il MAME supporta varie librerie multimediali per la fase di rendering e di missaggio audio. In Fig. 2.6 è visibile un diagramma delle classi relativo alla funzionalità di rendering. Quest'ultimo viene eseguito innanzitutto creando la finestra in cui verrà visualizzato il rendering (classe `osd_window`), tramite la funzione `osd_renderer::make_for_type` viene istanziata una delle classi per il rendering (es.: `renderer_ogl`, `renderer_d3d9`, `renderer_sd12`, ecc...) che comunica direttamente con la libreria grafica. Alla creazione della classe per il rendering viene chiamato il metodo `osd_renderer::create` che si occupa di inizializzare il necessario per il processo. Per ogni frame della macchina che viene emulato c'è una fase di disegno tramite il metodo `osd_renderer::draw`.

La classe che si occupa del rendering utilizzando la libreria SDL è `renderer_sd12` che utilizza le seguenti funzioni SDL:

- `SDL_CreateRenderer`: crea un contesto di rendering 2D per una finestra;
- `SDL_SetRenderDrawColor`: imposta il colore utilizzato per le operazioni di disegno;

- **SDL_RenderFillRect**: riempie un rettangolo con il colore di disegno corrente; è usato per disegnare la primitiva QUAD;
- **SDL_RenderDrawLine**: disegna una linea con il colore di disegno corrente; è usato per disegnare la primitiva LINE;
- **SDL_RenderPresent**: aggiorna il contesto di rendering con il framebuffer corrente.

Di questi la prima viene utilizzata durante la fase di inizializzazione nella funzione `create`, mentre le altre vengono utilizzate durante la fase di disegno nella funzione `draw`.

2.2.2 Missaggio audio

Il missaggio audio è quel procedimento con cui due o più campioni audio sono fusi in un unico output sonoro. Attualmente il MAME emula una cinquantina di chip audio, ma per alcuni giochi degli anni '70 e '80 invece di emulare i circuiti si sono semplicemente utilizzati i suoni registrati dalla scheda originale poiché il sonoro era prodotto mediante circuiti analogici, la cui emulazione è più complessa. Sono supportate cinque librerie per la gestione della scheda audio: SDL (che è multi piattaforma), DirectAudio (dalla libreria DirectX) e XAudio2 per Windows, Core Audio per macOS, PortAudio per Windows e macOS [Nicola Salmoria 2002].

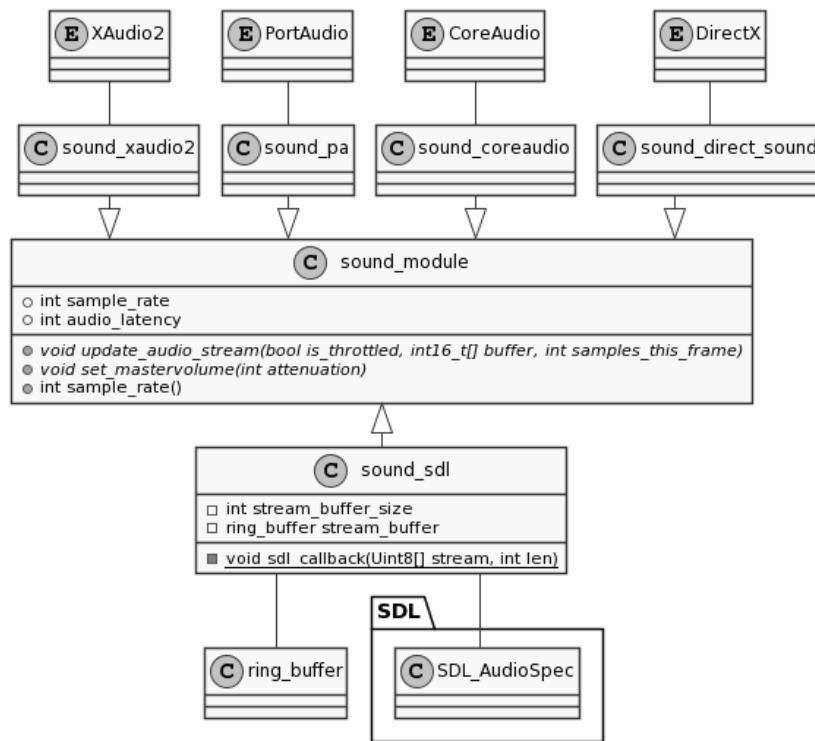


Figura 2.7: Diagramma delle classi relative al missaggio audio

La classe che si occupa del missaggio audio utilizzando la libreria SDL è `sound_sdl2` che utilizza le seguenti funzioni SDL:

- **SDL_OpenAudio**: apre il dispositivo audio;

- **SDL_PauseAudio**: mette in pausa o ripristina la riproduzione audio;
- **SDL_CloseAudio**: interrompere l'elaborazione audio e chiude il dispositivo audio.

La funzione `SDL_OpenAudio` utilizza la struttura `SDL_AudioSpec` che contiene informazioni come il formato del buffer audio, il numero di canali, i bit per canale, ecc..., ed un puntatore ad una funzione di callback asincrona; questa funzione viene utilizzata per riempire il buffer audio con il suono da riprodurre. Come è visibile in Fig. 2.7 la funzione `sdl_callback` ha due parametri `stream` e `len` che sono rispettivamente il buffer da riempire con il suono e la dimensione del buffer [Pazera 2003]. Il MAME utilizza un buffer ad anello (classe `ring_buffer`) in cui la macchina emulata aggiunge campioni audio e da cui la funzione `sdl_callback` li rimuove e li copia nel vettore `stream` che viene poi inviato da SDL alla scheda audio.

2.2.3 Gestione input

Il MAME nasce come emulatore di videogiochi arcade e quindi ha bisogno di gestire l'input della gettoniera, dei controlli per il personale di servizio⁸ e dei comandi del giocatore. Il dispositivo più usato è il joystick, tuttavia alcuni apparecchi montano volante e pedali, spinner, trackball, pistole e fucili ottici, ecc... [Nicola Salmoria 2002].

La libreria SDL gestisce la tastiera, il mouse e i joystick (pistola ottica, volante e pedali rientrano in questa categoria) tramite un sistema ad eventi:

- per la tastiera due tipi di eventi: pressione e rilascio dei tasti;
- per il mouse tre tipi di eventi: movimento del mouse, pressione e rilascio dei tasti;
- per il joystick tra i tre e i cinque tipi di eventi: pressione e rilascio dei tasti, movimento della leva di comando, movimento della levetta⁹ e movimento della trackball.

Come mostrato in Fig. 2.8 il sistema di gestione degli eventi di SDL offre tre metodi diversi per l'acquisizione [Pazera 2003]:

- attesa: questo è il metodo utilizzato per la gestione dell'input nei software desktop, in cui il programma è in attesa dell'input utente per poi eseguire un'azione;
- polling: è il metodo solitamente utilizzato nei videogiochi. Solitamente il ciclo di esecuzione di un videogioco è: processare l'input, aggiorare lo stato, eseguire il rendering e il missaggio audio;
- diretto: questo metodo dà la possibilità di leggere lo stato dei devices in qualsiasi momento in modo asincrono. Lo svantaggio di questo metodo è che bisogna comunque eseguire il polling dalla coda degli eventi.

⁸I controlli per il personale di servizio sono quegli interruttori presenti su apparecchi non recenti che servono per testare il funzionamento dell'apparecchio, regolare il livello di difficoltà, ecc...; sono stati poi sostituiti dai menu interattivi.

⁹In inglese chiamato "hat switch" oppure "POV switch".

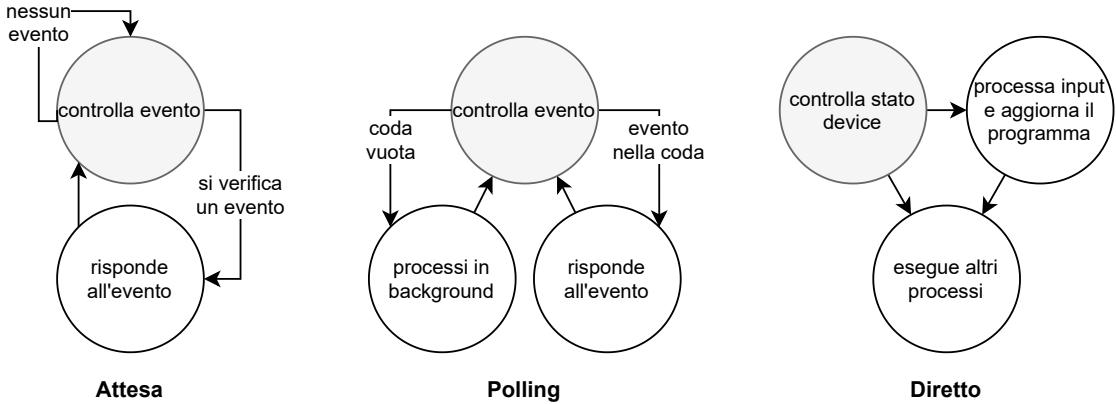


Figura 2.8: I tre metodi del gestore eventi della libreria SDL

Il sistema di gestione dell'input del MAME utilizza il metodo di polling attraverso le funzioni della libreria: `SDL_PollEvent` e `SDL_PumpEvents`; la prima esegue il controllo per gli eventi attualmente in sospeso e la seconda aggiorna la coda degli eventi e lo stato del dispositivo di input. Il sistema, schematizzato in Fig. 2.9, è formato dai device e dai moduli. La classe `input_module_base` che implementa un modulo di input dà la possibilità di sottoscrivere un device al sistema ad eventi e fornisce l'interfaccia per mapparlo con i comandi emulati dal MAME (ad es.: nel caso di SDL si mappa `SDLK_SPACE` con `P1_BUTTON1`). Il device, implementato tramite la classe `device_info`, offre le funzioni per gestire la coda dell'input e di impostarne lo stato (ad es.: tasto X premuto, tasto Y rilasciato).

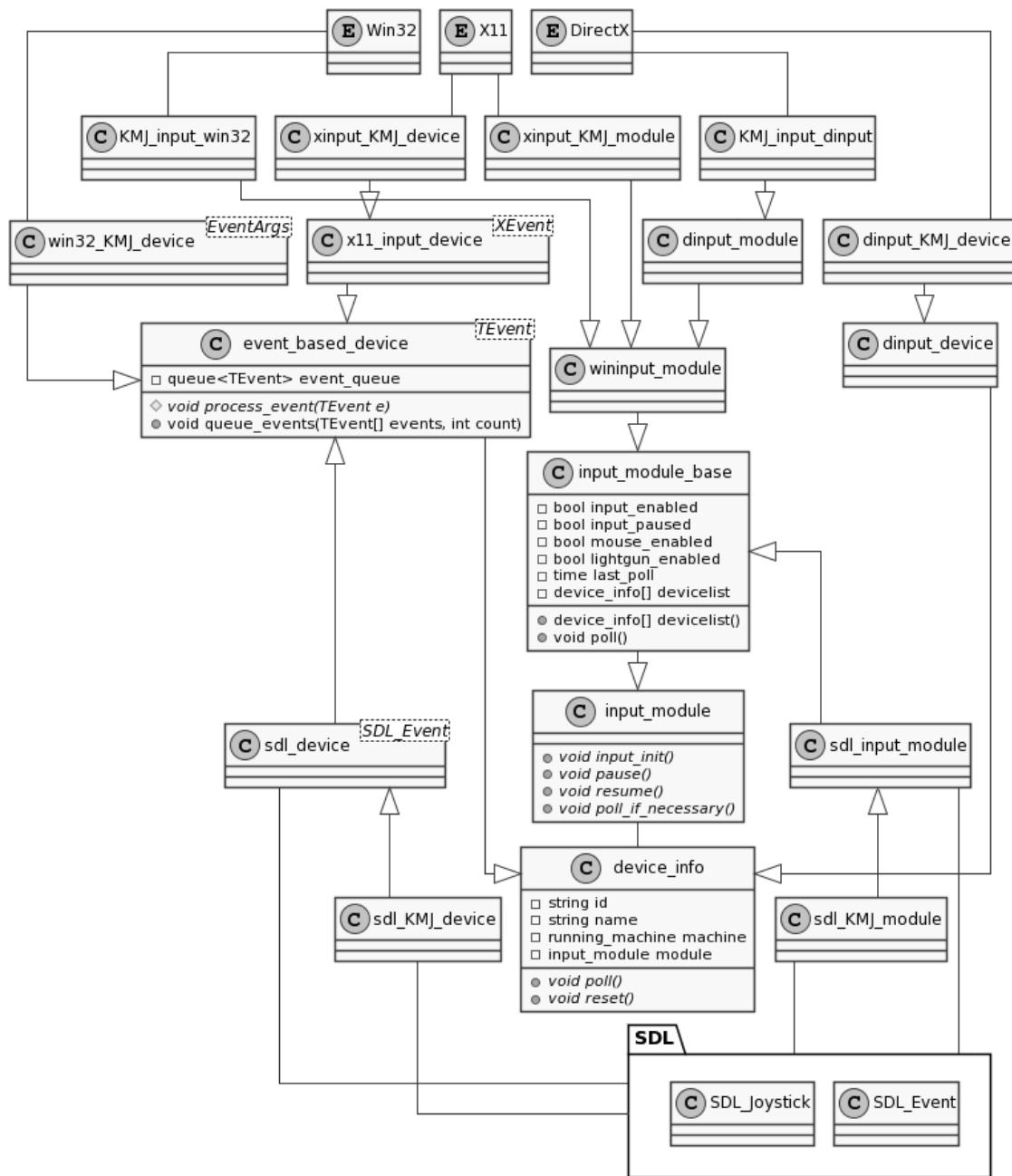


Figura 2.9: Diagramma delle classi relative ai moduli di input. Per ridurre la dimensione del diagramma sono state omesse le singole classi per gestire tastiera, mouse e joystick e sono state raggruppate con la sigla KMJ (Keyboard, Mouse, Joystick)

Capitolo 3

Implementazione

In questo capitolo verranno descritte le cinque fasi aggiuntive del cloud gaming e la loro implementazione in C++ come nuovi moduli del MAME: la cattura audio-video, la codifica, la trasmissione, la decodifica e la gestione dell'input utente.

3.1 Cattura

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.

3.1.1 Video

Le funzioni della libreria SDL utilizzate sono:

- CreateRGBSurfaceWithFormat: Crea una superficie RGB specificando il formato pixel da utilizzare;
- CreateRenderer: Crea un contesto di rendering 2D per una finestra;
- CreateSoftwareRenderer: Crea un contesto di rendering 2D per una superficie;
- RWFromMem: Prepara un buffer di memoria di lettura-scrittura da utilizzare con la struttura dati RWops (read-write opaque pointer structure);
- SetRenderDrawColor: Imposta il colore utilizzato per le operazioni di disegno;
- RenderFillRect: Riempe un rettangolo con il colore di disegno corrente;
- RenderDrawLine: Disegna una linea con il colore di disegno corrente;

- RenderPresent: Aggiorna il contesto di rendering con il frame generato dalle funzioni di disegno.

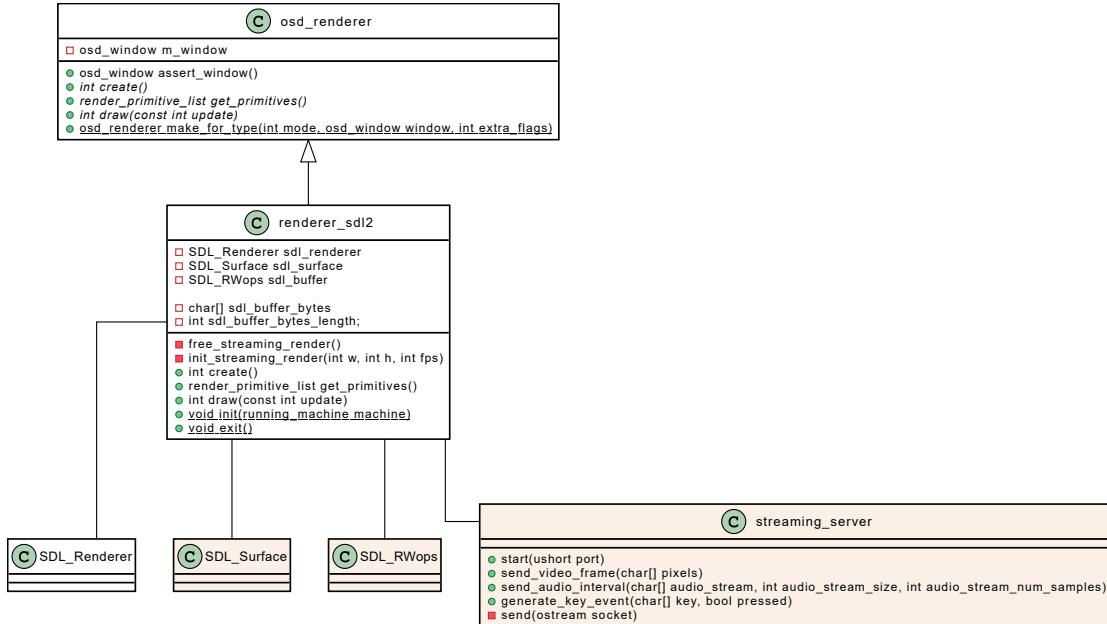


Figura 3.1: Diagramma delle classi (parziale) relative al rendering

3.1.2 Audio

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.

3.2 Codifica

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.

3.2.1 MPEG

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.

Compression

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.

Video

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.

Audio

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.

Trasmission

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.

3.2.2 FFmpeg

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum [FFmpeg team 2021].

Libs.

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.

3.3 Trasmissione

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.

3.3.1 Web APIs

Le API Web sono un insieme di API e interfacce che comprendono la potente capacità di creazione di script del Web. A seguire quelli utilizzati in questo progetto [Mozilla 2020].

WebSocket

WebSocket è un protocollo di comunicazione del computer che fornisce canali di comunicazione full-duplex su una singola connessione TCP. È compatibile con HTTP perché l'handshake WebSocket utilizza l'intestazione di aggiornamento HTTP per passare dal protocollo HTTP al protocollo WebSocket. È supportato nativamente da tutti i browser e il suo utilizzo è simile ai normali socket sia sul lato client che su quello server. Per questi motivi è il protocollo di comunicazione generico più utilizzato sul web [Mozilla 2021d].

3.4 Decodifica

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.

3.4.1 Web APIs

Le API Web sono un insieme di API e interfacce che comprendono la potente capacità di creazione di script del Web. A seguire quelli utilizzati in questo progetto [Mozilla 2020].

Canvas API

L'API Canvas fornisce un mezzo per disegnare grafica tramite JavaScript, si concentra principalmente sulla grafica 2D ma quando viene utilizzata dall'API WebGL può disegnare grafica 2D e 3D con accelerazione hardware. È completamente supportato da tutti i browser [Mozilla 2021b].

WebGL API

WebGL è un'API JavaScript, progettata e gestita dal gruppo no-profit Khronos, per il rendering di grafica 2D e 3D che consente l'utilizzo accelerato dalla GPU della fisica e dell'elaborazione e degli effetti delle immagini. WebGL 1.0 è supportato su tutti i browser, mentre WebGL 2.0 viene testato su Safari [Mozilla 2021c].

3.4.2 Librerie JavaScript

Per il front-end, sono state utilizzate una libreria JavaScript open source per la decodifica del filmato.

JSMpeg

JSMpeg è una libreria composta da un demuxer MPEG-TS, un decoder video MPEG1 e audio MP2, con un sistema di rendering basato sia su WebGL che su Canvas2D, ed un sistema di output audio basato su WebAudio. Consente lo streaming a bassa latenza (~50ms) tramite WebSocket, ed è rilasciata con licenza MIT [Dominic Szablewski 2021].

3.5 Gestione input

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.

3.5.1 Librerie JavaScript

Per il front-end, sono state utilizzate due librerie JavaScript open source per la gestione degli input.

Keypress

Keypress è una libreria per la cattura dell'input da tastiera specializzata per l'uso in contesti videoludici, rilasciata con licenza Apache 2.0. Viene utilizzata per gestire l'input da tastiera nel front-end [David Mauro 2021].

GameController.js

GameController.js è una libreria che estende le Web API per il gamepad, è rilasciata con licenza MIT. Nel front-end viene utilizzata per gestire i gamepads, per consentire il multiplayer sullo stesso dispositivo [Alvaro Montoro 2021].

3.5.2 SDL input

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.

Capitolo 4

Prestazioni

Questo capitolo analizza le prestazioni del progetto relativamente ai tre difetti intrinseci del cloud gaming: riduzione della qualità audio-video, bit-rate richiesto e il problema della latenza.

4.1 Qualità audio-video

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.



Figura 4.1: Comparazione tra rendering e streaming

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco

laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.

4.1.1 Peak signal-to-noise ratio

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.



Figura 4.2: Comparazione tra rendering e streaming

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.

4.1.2 Structural Similarity Index Method

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in

voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.

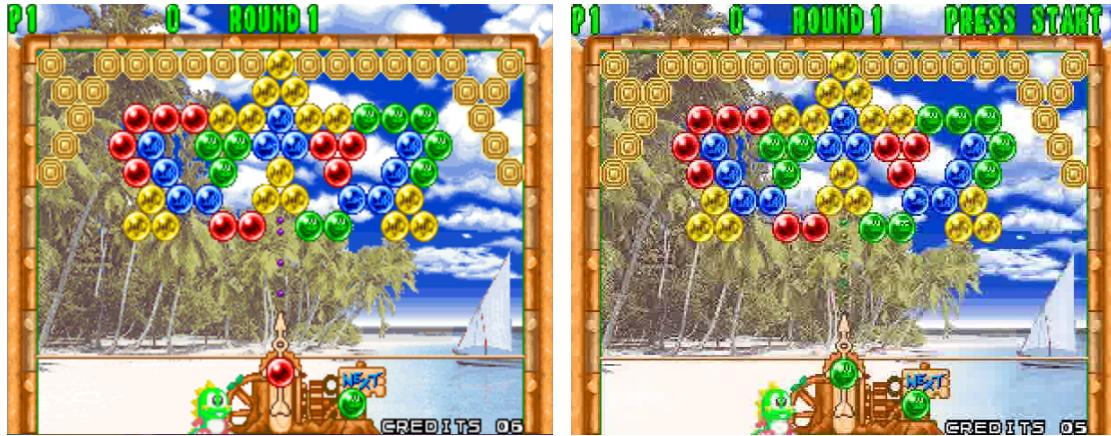


Figura 4.3: Comparazione tra rendering e streaming

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.

4.2 Bit-rate

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum [Maggiorini et al. 2016].

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.

4.3 Latenza

Per quanto riguarda la latenza è un problema che non può essere eliminato nel cloud gaming perché le fasi di un videogioco sono: ricezione input, esecuzione, rendering, display; mentre nel caso del cloud gaming si aggiungono: invio al server dell'input utente, codifica audio-video, invio all'utente dello stream video, decodifica audio-video. Un leggero ritardo in un filmato su internet o in una videochiamata molto probabilmente passa inosservato, ma durante una partita la latenza può rendere il gioco ingiocabile, una tempistica di esempio è visibile in Fig. 4.4, fortunatamente il rapido sviluppo delle reti a banda larga hanno reso questo problema meno evidente e il cloud gaming una realtà [Shea et al. 2013].



Figura 4.4: Latenza del videogioco: locale vs cloud gaming. Fonte: shadow.tech/blog/news/roadmap-cloud-gaming-without-latency

Come mostrato in tabella 4.1 lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum [Shea et al. 2013].

Tipo di gioco	Prospettiva	Soglia di ritardo (ms)
FPS	Prima persona	100
RPG	Terza persona	500
RTS	Onnipresente	1000

Tabella 4.1: Ritardo tollerato per tipo di gioco

Direzioni future di ricerca e conclusioni

Come detto precedentemente il progetto è stato pensato per essere utilizzato in uno stand per il retrogaming, ma se utilizzato come servizio web sarebbe necessario creare un sistema di account tramite cui il giocatore potrebbe salvare e caricare lo stato del gioco, pubblicare i punteggi nella leaderboard, invitare altri giocatori ad unirsi alla partita supportando così il multiplayer da devices differenti.

Per ridurre la dimensione dei pacchetti si potrebbero usare due codec open-source che in futuro saranno supportati nativamente dalla maggior parte dei browser: AOMedia Video 1 (AV1), progettato per trasmissioni video su Internet, e Opus, un codec audio lossy utilizzato per la comunicazione in tempo reale. Ambedue inseribili nel contenitore WebM.

Per diminuire l'overhead di comunicazione si potrebbe sostituire il protocollo WebSocket con RTP utilizzabile tramite la tecnologia WebRTC.

Appendice A

Listato

In questa appendice sono riportate alcune parti del codice C++ a cui si fa riferimento nel testo. Per ogni listato è riportato il file da cui è stato estratto.

A.1 Modifiche classi SDL

Di seguito le modifiche che sono state apportate alle classi originali del MAME nel main e per recuperare il rendering e il missaggio audio.

A.1.1 Main

File `osd/sdl/sdlmain.cpp`

```
1 int main(int argc, char** argv)
2 {
3     auto r = 0;
4     streaming_server::instance().activate(argc, argv);
5
6     if (streaming_server::instance().is_active())
7     {
8         streaming_server::instance().on_accept = [&](auto parameters)
9         {
10             streaming_server::run_new_process(argc, argv);
11
12             r = main_sdl(argc, argv, parameters["game"]);
13         };
14
15         streaming_server::instance().start(8888);
16     }
17     else
18         r = main_sdl(argc, argv, "");
19
20     return r;
21 }
```

A.1.2 Missaggio audio

File `osd/modules/sound/sdl_sound.cpp`

```

1 void sound_sdl::sdl_callback(
2     void* userdata, Uint8* stream, int len)
3 {
4     //... CODICE NON MODIFICATO ...
5
6     if (streaming_server::instance().is_active())
7     {
8         streaming_server::instance().send_audio_interval(
9             stream, len, this->sdl_xfer_samples);
10
11     memset(stream, 0, len); //silence local outputs
12 }
13
14 //... CODICE NON MODIFICATO ...
15 }
```

A.1.3 Rendering

Files `osd/modules/render/draw13.cpp` e `osd/modules/render/draw13.h`

```

1 void renderer_sdl2::free_streaming_render() const
2 {
3     if (m_sdl_buffer_bytes != nullptr)
4     {
5         SDL_RWclose(m_sdl_buffer);
6         SDL_FreeSurface(m_sdl_surface);
7         SDL_DestroyRenderer(m_sdl_renderer);
8
9         delete [] m_sdl_buffer_bytes;
10    }
11 }
12
13 void renderer_sdl2::init_streaming_render(
14     const int w, const int h, const int fps)
15 {
16     free_streaming_render();
17
18     m_sdl_buffer_length = w * h * 4;
19     m_sdl_buffer_bytes = new char[m_sdl_buffer_length];
20
21     m_sdl_surface = SDL_CreateRGBSurfaceWithFormat(
22         0, w, h, 32, SDL_PIXELFORMAT_RGBA32);
23
24     //Crea il rendering su superficie al posto di quello su finestra
25     m_sdl_renderer = SDL_CreateSoftwareRenderer(
26         m_sdl_surface);
```

```
27
28     m_sdl_buffer = SDL_RWFromMem(
29         m_sdl_buffer_bytes, m_sdl_buffer_bytes_length);
30
31     if (streaming_server::instance().is_active())
32         streaming_server::instance().init_encoding(w, h, fps);
33 }
34
35 int renderer_sdl2::create()
36 {
37     //... CODICE NON MODIFICATO ...
38
39     if (streaming_server::instance().is_active())
40     {
41         const auto nd = win->get_size();
42         init_streaming_render(
43             nd.width(), nd.height(), win->m_win_config.refresh);
44     }
45     else
46     {
47         if (video_config.waitvsync)
48             m_sdl_renderer = SDL_CreateRenderer(
49                 dynamic_pointer_cast<sdl_window_info>(
50                     win)->platform_window(),
51                     -1, SDL_RENDERER_PRESENTVSYNC |
52                     SDL_RENDERER_ACCELERATED);
53     else
54         m_sdl_renderer = SDL_CreateRenderer(
55             dynamic_pointer_cast<sdl_window_info>(
56                 win)->platform_window(),
57                     -1, SDL_RENDERER_ACCELERATED);
58 }
59
60     //... CODICE NON MODIFICATO ...
61 }
62
63 int renderer_sdl2::draw(int update)
64 {
65     //... CODICE NON MODIFICATO ...
66     SDL_RenderPresent(m_sdl_renderer);
67
68     if (streaming_server::instance().is_active())
69     {
70         streaming_server::instance().send_video_frame(
71             static_cast<uint8_t*>(m_sdl_surface->pixels));
72         SDL_RWseek(m_sdl_buffer, 0, RW_SEEK_SET);
73     }
74     return 0;
75 }
```

A.1.4 Gestione input

Files osd/modules/input/input_sdl.cpp

```

1  sdl_keyboard_device(
2      running_machine &machine, const char *name,
3      const char *id, input_module &module):
4      sdl_device(machine, name, id, DEVICE_CLASS_KEYBOARD, module),
5      keyboard({{0}})
6  {
7      streaming_server::instance().set_keyboard(this);
8 }
```

A.2 Moduli nuovi

Di seguito alcune funzioni nei due moduli che sono stati creati: codifica e server per la comunicazione WebSocket.

A.2.1 Codifica

File lib/util/encoding/encode_to_movie.hpp

```

1  static int write_packet(
2      void* opaque, uint8_t* buf, int buf_size)
3  {
4      auto* const this_ =
5          static_cast<encode_to_movie*>(opaque);
6
7      this_->socket->write(
8          reinterpret_cast<const char*>(buf), buf_size);
9
10     return 1; //1 element wrote
11 }
12
13 //Add video frame
14 void add_frame(const uint8_t* pixels)
15 {
16     if (!header)
17         write_header();
18
19     avpicture_fill(
20         reinterpret_cast<AVPicture*>(rgb_frame),
21         pixels, PIXEL_FORMAT_IN, width, height);
22
23 //RGB to YUV
24 sws_scale(
25     encoder_context->video_converter_context,
26     rgb_frame->data, rgb_frame->linesize, 0, height,
27     yuv_frame->data, yuv_frame->linesize
28 );
```

```

29
30     av_init_packet(&video_packet);
31     video_packet.data = nullptr;
32     video_packet.size = 0;
33
34     avcodec_send_frame(
35         encoder_context->video_codec_context, yuv_frame);
36
37     const auto got_packet_ptr =
38         avcodec_receive_packet(
39             encoder_context->video_codec_context,
40             &video_packet) == 0;
41
42     if (got_packet_ptr)
43     {
44         const auto ret =
45             av_interleaved_write_frame(
46                 encoder_context->muxer_context,
47                 &video_packet);
48
49         if (ret == 0)
50             send_it();
51         else
52             error("Error while writing video frame", ret);
53     }
54
55     av_packet_unref(&video_packet);
56 }
57
58 //Add audio instant
59 void add_instant(
60     const uint8_t* audio_stream,
61     const int audio_stream_size,
62     const int audio_stream_num_samples)
63 {
64     if (!header)
65         write_header();
66
67     wav_frame->nb_samples = audio_stream_num_samples;
68
69     auto ret = avcodec_fill_audio_frame(
70         wav_frame,
71         AUDIO_CHANNELS_IN,
72         AUDIO_SAMPLE_FORMAT_IN,
73         audio_stream,
74         audio_stream_size,
75         1); //no-alignment
76     if (ret < 0)
77         die("Cannot fill audio frame", ret);

```

```

78
79     if (convertedData == nullptr)
80     {
81         ret = av_samples_alloc(
82             &convertedData,
83             nullptr,
84             AUDIO_CHANNELS_OUT,
85             aac_frame->nb_samples,
86             AUDIO_SAMPLE_FORMAT_OUT,
87             0);
88         if (ret < 0)
89             die("Could not allocate samples", ret);
90     }
91
92     auto outSamples = swr_convert(
93         encoder_context->audio_converter_context,
94         //output
95         nullptr, 0,
96         //input
97         const_cast<const uint8_t**>(
98             wav_frame->data), wav_frame->nb_samples
99     );
100
101    if (outSamples < 0)
102        die("Could not convert", outSamples);
103
104    for (;;)
105    {
106        outSamples = swr_get_out_samples(
107            encoder_context->audio_converter_context, 0);
108
109        if (outSamples <
110            encoder_context->audio_codec_context->frame_size *
111            encoder_context->audio_codec_context->channels)
112            break;
113
114        swr_convert(
115            encoder_context->audio_converter_context,
116            //output
117            &convertedData, aac_frame->nb_samples,
118            //input
119            nullptr, 0);
120
121        const auto bufferSize = av_samples_get_buffer_size(
122            nullptr,
123            encoder_context->audio_codec_context->channels,
124            aac_frame->nb_samples,
125            encoder_context->audio_codec_context->sample_fmt,
126            0);

```

```

127
128     if (buffer_size < 0)
129         die("Invalid buffer size", buffer_size);
130
131     ret = avcodec_fill_audio_frame(
132         aac_frame,
133         encoder_context->audio_codec_context->channels,
134         encoder_context->audio_codec_context->sample_fmt,
135         convertedData,
136         buffer_size,
137         0);
138
139     if (ret < 0)
140         die("Could not fill frame", ret);
141
142     av_init_packet(&audio_packet);
143     audio_packet.data = nullptr;
144     audio_packet.size = 0;
145
146     int got_packet_ptr;
147
148     ret = avcodec_encode_audio2(
149         encoder_context->audio_codec_context,
150         &audio_packet, aac_frame,
151         &got_packet_ptr);
152
153     if (ret < 0)
154         die("Error encoding audio frame", ret);
155
156     if (got_packet_ptr)
157     {
158         audio_packet.stream_index = 1;
159
160         ret = av_interleaved_write_frame(
161             encoder_context->muxer_context, &audio_packet);
162
163         if (ret == 0)
164             send_it();
165         else
166             error("Error while writing audio frame", ret);
167     }
168
169     av_packet_unref(&audio_packet);
170 }
171 }
```

A.2.2 Server

File lib/util/streaming_server.hpp

```

1 //Invio input utente a SDL
2 void generate_key_event(
3     const char* key, const string& down) const
4 {
5     SDL_Event e;
6     e.type = (down == "D" ? SDL_KEYDOWN : SDL_KEYUP);
7
8     e.key.keysym.scancode =
9         SDL_GetScancodeFromName(key);
10
11    e.key.keysym.sym =
12        SDL_GetKeyFromScancode(e.key.keysym.scancode);
13
14    keyboard->queue_events(&e, 1);
15 }
16
17 //Singleton
18 static streaming_server& instance()
19 {
20     static streaming_server instance;
21     return instance;
22 }
23
24 void start(const unsigned short port)
25 {
26     server = make_unique<ws_server>();
27     server->config.client_mode = true;
28     server->config.port = port;
29     server->config.timeout_request = 0; //no timeout
30
31     auto& endpoint = server->m_endpoint["/?"];
32
33     endpoint.on_open = [this](auto connection)
34     {
35         cout
36             << "-Opened_connection_from_"
37             << connection->remote_endpoint_address << ":"
38             << connection->remote_endpoint_port << endl;
39
40         game_thread = make_unique<thread>(
41             on_accept, connection->parameters);
42
43         game_start_time = chrono::system_clock::now();
44
45         const auto game = connection->parameters["game"];
46

```

```

47     cout
48         << "Starting game:" << game
49         << endl;
50     };
51
52     endpoint.on_message = [this](auto connection, auto message)
53 {
54     const auto msg = message->string();
55     const auto values = ws_server::split(msg, ":");

56     if (values[0] == "ping")
57         process_pausing_mechanism();
58     else if (values[0] == "key")
59         process_key(values);
60     };
61
62     endpoint.on_error = [](auto connection, auto code)
63 {
64     cout
65         << "-Error on connection from"
66         << connection->remote_endpoint_address << ":"
67         << connection->remote_endpoint_port << endl
68         << ":" << code.message() << endl;
69     };
70
71     endpoint.on_close = [this](auto connection, auto status, auto reason)
72 {
73     const auto game_end_time = chrono::system_clock::now();

74     const auto game_total_minute_played =
75         chrono::duration_cast<chrono::minutes>(
76             game_end_time - game_start_time);

77     const auto game = connection->parameters["game"];

78     cout
79         << "-Closed connection from"
80         << connection->remote_endpoint_address << ":"
81         << connection->remote_endpoint_port << endl
82         << ":" << reason << endl;

83     cout
84         << game
85         << " played for :"
86         << game_total_minute_played.count()
87         << " min." << endl;

88     machine->schedule_exit();
89
90     cout
91         << game
92         << " played for :"
93         << game_total_minute_played.count()
94         << " min." << endl;
95 };

```

```
96
97     cout
98         << "Game.streaming.server.listening.on."
99         << port << endl;
100
101    server->start();
102
103    if (game_thread->joinable())
104        game_thread->join();
105 }
```

Appendice B

Manuale utente

In questo appendice viene descritta la procedura di configurazione del programma.

B.1 Configurazione

Nella cartella `./roms/` vanno messe tutte le ROM dei giochi che si vuole rendere disponibili.

Nella cartella `./Streaming/HTML/roms/` vanno messe le copertine dei giochi disponibili in formato PNG, della dimensione $222 \times 315px$.

Nel file `./Streaming/HTML/roms/list.txt` vanno messe le informazioni del gioco, nel seguente formato `rom_name;description;other_info` ad esempio:
`sfiii3nr1;Street Fighter III: 3rd Strike;Capcom - 1999.`

B.2 Esecuzione

Per avviare il MAME CGP bisogna eseguire il comando

`./mame64 -streamingserver -window -video accel -sound sdl -resolution 640x480@30`
Per Linux e Windows sono forniti nel progetto i due scripts `run.sh` e `run.bat`.

Bibliografia

- Bielievtssov, Stanislav, Igor Ruban, Kyrylo Smelyakov e Dmytro Sumtsov (dic. 2018). «Network technology for transmission of visual information». In: pp. 104–120.
- D'Angelo, Gabriele, Stefano Ferretti e Moreno Marzolla (mag. 2015). «Cloud for Gaming». In: pp. 1–6. DOI: 10.1007/978-3-319-08234-9_39-1.
- Domenico, Andrea, Gianluca Perna, Martino Trevisan, Luca Vassio e Danilo Giordano (dic. 2020). *A network analysis on cloud gaming: Stadia, GeForce Now and PSNow*.
- Forsyth, David A., Jean Ponce, Soumen Mukherjee e Arup Kumar. Bhattacharjee (2015). *Computer vision: a modern approach*. Pearson Education Limited.
- Huang, Chun-Ying, Kuan-Ta Chen, De-Yu Chen, Hwai-Jung Hsu e Cheng-Hsin Hsu (gen. 2014). «GamingAnywhere: The first open source cloud gaming system». In: *ACM Transactions on Multimedia Computing, Communications, and Applications (TOMCCAP)* 10. DOI: 10.1145/2537855.
- Ilya Grigorik (2013). *High Performance Browser Networking*. O'Reilly Media, Inc. ISBN: 1449344763.
- Karachristos, Theofilos, Dimitrios Apostolatos e D. Metafas (set. 2008). «A real-time streaming games-on-demand system». In: pp. 51–56. DOI: 10.1145/1413634.1413648.
- Maggiorini, D., Laura Ripamonti, Giacomo Quadrio, A. Bujari e Daniele Ronzani (giu. 2016). «Network analysis of the Sony Remote Play system». In: pp. 10–13. DOI: 10.1109/ISCC.2016.7543706.
- Manzano, Marc, Manuel Urueña, Mirko Suznjevic, Eusebi Calle, José Hernández e Maja Matijasevic (mar. 2014). «Dissecting the protocol and network traffic of the OnLive cloud gaming platform». In: *Multimedia Systems* 20, pp. 1–20. DOI: 10.1007/s00530-014-0370-4.
- Mileff, Peter e Judit Dudra (mag. 2012). «Efficient 2D software rendering». In: *Production Systems and Information Engineering* 6, pp. 99–110.
- Nicola Salmoria (2002). «Il progetto MAME: reverse engineering e macchine da gioco». Tesi di laurea mag. Università degli studi di Siena.
- Pazera, Ernest (2003). *Focus on SDL*. Premier Press.
- Popovic, Milica, Dejan Dražić, Philipp Svoboda, N. Nikaein, Srdjan Krco e Markus Laner (gen. 2016). «Latency analysis for M2M and online gaming traffic in an HSPA network». In: 31, pp. 259–277.
- Shea, Ryan, Jiangchuan Liu, Edith Ngai e Yong Cui (lug. 2013). «Cloud Gaming: Architecture and Performance». In: *Network, IEEE* 27, pp. 16–21. DOI: 10.1109/MNET.2013.6574660.
- Suznjevic, Mirko e Maja Homen (feb. 2020). «Use of Cloud Gaming in Education». In: ISBN: 978-1-83880-009-3. DOI: 10.5772/intechopen.91341.

Sitografia

- Adam Bankhurst (ott. 2018). *Microsoft Announces Global Game Streaming Service, Project xCloud, Beta Next Year.* ign.com/articles/2018/10/08/microsoft-announces-global-game-streaming-service-project-xcloud-beta-next-year. Accessed: 2021-05-12.
- Alvaro Montoro (apr. 2021). *gameController.js*. github.com/alvaromontoro/gamecontroller.js. Accessed: 2021-04-21.
- Amazon (apr. 2021). *Amazon Luna*. amazon.com/luna. Accessed: 2021-04-10.
- Andrea Pitzozzi (dic. 2019). *I 4 fallimenti più clamorosi del decennio*. wired.it/economia/business/2019/12/27/fallimenti-decennio. Accessed: 2021-05-16.
- Beijing Cloud Union (ago. 2018). *Cloud Union*. cloudunion.cn. Accessed: 2021-05-07.
- David Mauro (apr. 2021). *Keypress*. github.com/dmauro/Keypress. Accessed: 2021-04-21.
- Dominic Szablewski (apr. 2021). *JSMpeg - MPEG1 Video & MP2 Audio Decoder in JavaScript*. github.com/phoboslab/jsmpeg. Accessed: 2021-04-05.
- Electronic Arts (ott. 2018). *Project Atlas*. ea.com/news/announcing-project-atlas. Accessed: 2021-04-09.
- FFmpeg team (mar. 2021). *FFmpeg Documentation*. ffmpeg.org/doxygen/trunk. Accessed: 2021-03-30.
- Gaikai (giu. 2012). *Gaikai Open Platform*. gaikai.com/open-platform. Accessed: 2021-05-12.
- Google (mar. 2019a). *Google Partners with AMD for Custom Stadia GPU*. stadia.dev/intl/fr_ca/blog/google-partners-with-amd-for-custom-stadia-gpu. Accessed: 2021-05-12.
- (apr. 2021). *Google Stadia*. stadia.google.com. Accessed: 2021-04-10.
- (mar. 2019b). *Welcome to Stadia*. stadia.dev/intl/it_it/blog/welcome-to-stadia. Accessed: 2021-05-12.
- JP Mangalindan (ott. 2020). *Cloud gaming's history of false starts and promising reboots*. polygon.com/features/2020/10/15/21499273/cloud-gaming-history-onlive-stadia-google. Accessed: 2021-04-09.
- Kif Leswing (set. 2020). *Apple issues new rules for App Store that will impact streaming game services from Google and Microsoft*. cnbc.com/2020/09/11/apple-app-store-new-rules-will-affect-google-stadia-microsoft-xcloud.html. Accessed: 2021-04-20.
- Kyle Orland (set. 2020). *Amazon Luna servers will run Windows games directly on Nvidia T4 GPUs*. arstechnica.com/gaming/2020/09/amazon-luna-supports-existing-windows-games-on-turing-level-gpus. Accessed: 2021-05-13.
- MAME Team (ago. 2018). *FAQ:Performance*. wiki.mamedev.org/index.php/FAQ:Performance. Accessed: 2021-05-25.
- (apr. 2021). *The Official Site of the MAME Development Team*. mamedev.org. Accessed: 2021-05-03.
- Microsoft (apr. 2021). *Xbox Game Pass cloud gaming*. xbox.com/en-US/xbox-game-pass-cloud-gaming. Accessed: 2021-04-09.

- Mozilla (mar. 2021a). *Audio and Video Delivery*. developer.mozilla.org/en-US/docs/Web/Guide/Audio_and_video_delivery. Accessed: 2021-04-16.
- (feb. 2021b). *Canvas API*. developer.mozilla.org/en-US/docs/Web/API/Canvas_API. Accessed: 2021-04-21.
- (set. 2020). *Web APIs*. developer.mozilla.org/en-US/docs/Web/API. Accessed: 2021-04-21.
- (mar. 2021c). *WebGL*. developer.mozilla.org/en-US/docs/Web/API/WebGL_API. Accessed: 2021-04-21.
- (feb. 2021d). *WebSocket - Web APIs*. developer.mozilla.org/en-US/docs/Web/API/WebSocket. Accessed: 2021-04-16.
- Newzoo (apr. 2020). *Global Cloud Gaming Report*. newzoo.com/global-cloud-gaming-report. Accessed: 2021-04-09.
- Nvidia (apr. 2021). *GeForce Now*. nvidia.com/en-us/geforce-now. Accessed: 2021-04-11.
- Playkey (apr. 2021). *Playkey*. playkey.io. Accessed: 2021-04-19.
- RemoteMyApp (apr. 2021). *Vortex cloud gaming*. vortex.gg. Accessed: 2021-04-19.
- Ryan Whitwam (gen. 2014). *Sony's PlayStation Now uses custom-designed hardware with eight PS3s on a single motherboard*. extremetech.com/gaming/175005-sony-s-playstation-now-uses-custom-designed-hardware-with-eight-ps3s-on-a-single-motherboard. Accessed: 2021-05-12.
- SDL Community (dic. 2020). *SDL Wiki*. wiki.libsdl.org. Accessed: 2021-03-29.
- Sean Hollister (dic. 2010). *Gaikai enters closed beta, we get an exclusive first look*. engadget.com/2010-12-02-gaikai-enters-closed-beta-we-get-an-exclusive-first-look.html. Accessed: 2021-05-07.
- Sony (apr. 2021). *PlayStation Now*. playstation.com/en-us/ps-now. Accessed: 2021-04-10.
- Tom Warren (set. 2020a). *Amazon's Luna game streaming service is powered by Windows and Nvidia GPUs*. theverge.com/2020/9/25/21455610/amazon-luna-game-streaming-windows-nvidia-gpu-servers. Accessed: 2021-05-12.
- (giu. 2020b). *Microsoft to upgrade its xCloud servers to Xbox Series X hardware in 2021*. theverge.com/2020/6/18/21295326/microsoft-project-xcloud-xbox-series-x-servers-hardware-2021. Accessed: 2021-05-12.
- William Acke, Sam LaCroix, France Costrel e Melissa Wood (ago. 2020). *High Score - Ep. 1 - Boom & Bust*.

Elenco delle figure

1.1	Console iconiche, fino alla nona generazione	1
1.2	Ricavi globali dell'industria dei videogiochi dal 1971 al 2018 (non adeguati all'inflazione). Fonte: wikipedia.org	2
1.3	Instanze di GaaS	3
1.4	Streaming con perdita di pacchetti dell'8%	5
1.5	API, protocolli e servizi di rete del browser	5
1.6	Previsioni per il mercato globale del cloud gaming (in dollari americani). Fonte: newzoo.com/global-cloud-gaming-report	10
2.1	Panoramica del sistema	12
2.2	MAME, diagramma dei packages	12
2.3	Schema della struttura del MAME	13
2.4	Librerie e interfacce utilizzate dal MAME sulle diverse piattaforme	13
2.5	Pipeline di rendering 2D	14
2.6	Diagramma delle classi relative al rendering	15
2.7	Diagramma delle classi relative al missaggio audio	16
2.8	I tre metodi del gestore eventi della libreria SDL	18
2.9	Diagramma delle classi relative ai moduli di input. Per ridurre la dimensione del diagramma sono state omesse le singole classi per gestire tastiera, mouse e joystick e sono state raggruppate con la sigla KMJ (Keyboard, Mouse, Joystick)	19
3.1	Diagramma delle classi (parziale) relative al rendering	21
4.1	Comparazione tra rendering e streaming	25
4.2	Comparazione tra rendering e streaming	26
4.3	Comparazione tra rendering e streaming	27
4.4	Latenza del videogioco: locale vs cloud gaming. Fonte: shadow.tech/blog/news/roadmap-cloud-gaming-without-latency	28
B.1	Street Fighter Alpha artwork. © Capcom	

Elenco delle tabelle

1.1 Comparazione tra protocolli di trasporto	4
1.2 Piattaforme disponibili	9
4.1 Ritardo tollerato per tipo di gioco	28

Ringraziamenti 2.0

Ringrazio tutti coloro che hanno fatto parte del mio percorso di laurea magistrale:

- la mia famiglia che crede sempre che io possa fare tutto, senza capire che nella mia limitatezza è sempre una faticaccia;
- Laura Antonella (aka Dino) che con il suo amore ha chiuso il buco che avevo nel petto;
- Laura & Giulia, Simona, Eleonora, Martina, Greta e tutte le altre ragazze del dipartimento di farmacia. Grazie per aver reso divertenti le giornate di studio. C'era sempre il sole in biblioteca;
- i miei giocatori del BawiTeam. Tra lacrime, infortuni, risate e gioie. Che squadra meravigliosa;
- i miei compagni di corso: Carrarini, Dettori, Iervolino, Lombardi, Bonapace, Paduano, Vannucci, Zhab'yak per i fantastici progetti fatti insieme;
- i professori del dipartimento di informatica. Ho cercato di trarre il massimo dai vostri insegnamenti per poterli poi concretamente utilizzare;
- Mario, Fede, Nadia, Giorgio, Giovanni e Mariapina per esserci sempre stati (da oltre 20 anni!);
- Alessandro, Carmen, Claudio, Sba, Marika, Grazia, Emiliana, Kikka, anche se non ci siamo visti spesso siete stati vicini;
- i miei parenti di Treviglio che mi hanno aiutato e ospitato. Grazie per il supporto;
- tutti gli altri, anche se non menzionati, siete nel mio cuore.

Dedico questa tesi a Dino augurandole grandi successi accademici.

Milano, luglio 2021



Figura B.1: Street Fighter Alpha artwork. © Capcom