aMAZEing duo

Il progetto

aMAZEing duo è un multiplayer collaborativo in cui due giocatori devono uscire insieme da un labirinto nel quale sono presenti alcuni ostacoli. Uno dei due giocatori è sordo: non sente alcun suono ma può vedere il labirinto e gli ostacoli presenti; l'altro, invece, è cieco e, pertanto, non vede nulla se non i confini esterni del labirinto, sé stesso e il giocatore sordo.

Il gioco è composto da tre livelli: facile, medio e difficile. Gli ostacoli devono essere superati con l'ascolto di specifici codici/bip/suoni oppure evitandoli se visti. Al centro del gioco la collaborazione fra i due giocatori: alcuni livelli non sono superabili senza il contributo di entrambi.

Le fasi di lavoro

Fase 1 – Progettazione e creazione del materiale

Per prima cosa ho pensato agli ostacoli da superare. Dato che volevo che servisse l'impegno di entrambi i giocatori per superarli, quasi tutti gli ostacoli hanno una componente sonora e una visiva e sono:

- Position Door: porta gialla che, una volta collisa, attiva sulla mappa due "posizioni" gialle che devono essere raggiunte dai giocatori (uno in una e l'altro nell'altra) per poter far partire un audio casuale (fra quattro) con un *pitch* particolare. Se indovinato la porta si distrugge, se errato il tempo aumenta di dieci secondi e parte un nuovo audio;
- Code Door: porta azzurra che, una volta collisa, fa partire un codice (in numeri da 1 a 6) sotto forma di sequenze di bip. Se il codice viene inserito correttamente la porta si distrugge;
- Tile: mattonella che emette sei bip (udibili solo in sua prossimità) ogni quattro secondi e che può essere superata solo nel momento in cui non emette suono. Se attraversata mentre suona il giocatore viene riportato alla posizione di partenza;
- Spike: punte distribuite su alcuni muri del labirinto che, se toccate, riportano il giocatore alla posizione di partenza; le Spike non sono visibili al giocatore cieco e non emettono alcun suono.

Gli ostacoli sono distribuiti in modo diverso a seconda della difficoltà del labirinto, che diventa più grande e complesso da percorrere man mano che la difficoltà del livello aumenta.

Gli strumenti utilizzati per la creazione del materiale sono stati:

- Illustrator e Gimp per le sprites;
- Bfxr e AudioMass (https://audiomass.co/) per creare i suoni presenti nel gioco;
- OpenDyslexic (https://opendyslexic.org/) come font.

Fase 2 – Programmazione del networking e della comunicazione fra master e client

Si tratta della fase più complessa e che ha richiesto più tempo per la sua realizzazione. Per realizzare la comunicazione master-client ho sfruttato Photon Engine[1] in quanto più semplice da gestire e far funzionare. Per realizzare la lobby e le stanze è stato utilizzato il tutorial *Unity Multiplayer Tutorial - Custom Matchmaking (Part 4)* di Info Gamer[2], modificando in parte gli script e la grafica per sottometterli alle mie esigenze.

[1] Link al sito: https://www.photonengine.com/

[2] https://www.youtube.com/watch?v=onDorc3Qfn0&list=PLWeGoBm1YHVhH43SRzCo6Qr3Lm1W4Rw8z&index=6

Prima di scegliere Photon ho fatto qualche altro test anche con Mirror[3] (sempre seguendo alcuni tutorial) ma non sono stati soddisfacenti per i risultati ottenuti in quanto si verificavano errori di connessione che, dato anche il fatto che non avevo mai utilizzato nulla del genere prima, non sapevo come risolvere. Photon è stato molto più semplice e intuitivo da usare (e comprendere) e, dunque, ho optato per quello.

Fase 3 – Costruzione dei livelli e dei giocatori

Per terza cosa ho costruito i tre livelli previsti per il gioco, disposti ognuno in una scena diversa. Per ogni livello ho incluso un Game Object con la UI (pulsanti Pause e Leave, un testo di avviso per eventuali errori presenti nel gioco e le Spike, se previste), uno script che istanzia tutti gli oggetti (GameSetup), il labirinto e gli ostacoli previsti. Le porte hanno dei propri Panel che contengono le indicazioni su cosa fare e i tastierini per inserire i codici necessari per distruggerle.

I labirinti sono Tilemap su tre livelli: Base, Walls e Obstacles. Il terzo livello contiene gli ispessimenti dei muri ed è a parte in quanto, inizialmente, erano stati pensati come ostacoli "semplici" da dover evitare senza ripercussioni di sorta ed erano anche di colore diverso. Il gameplay ha rivelato che colori diversi avrebbero fatto pensare che sarebbe successo qualcosa a seguito di una collisione: per questo motivo sono stati ricolorati anche se mantenuti in un altro livello della Tilemap. Walls e Obstacles sono dotati di Tilemap Collider per far sì che i muri non vengano attraversati dai giocatori. Ho deciso di utilizzare le Tilemap per questo prototipo (anziché creare i livelli in modo procedurale) perché più semplici e veloci da realizzare; la varietà nei livelli è comunque data dallo *spawning* randomico degli ostacoli.

In seguito, ho realizzato i Prefab dei due giocatori e li ho posti su un Sorting Layer personalizzato in modo che non si creino *glitch* se si scontrano – ho riscontrato questo problema nelle prime fasi di testing – e ho scritto i loro script concentrandomi sul fatto che solo il Blind Player potesse sentire i suoni; per questo motivo a lui è associato anche un AudioListener come Component. La gestione del suono è stata un'altra cosa molto complicata per me e ha richiesto molti tentativi prima di riuscire nell'intento di rendere il Deaf Player completamente sordo. Il Blind Player, da parte sua, oltre a sentire i codici/suoni necessari per superare gli ostacoli sente anche alcuni suoni "ambientali" come, ad esempio, il *Bump* quando sbatte contro i muri o le esplosioni se si distrugge una porta.

Fase 4 – Testing

Per testare il gioco ho scaricato ParrelSync[4], una estensione di Unity che permette di utilizzare contemporaneamente due Editor e giocare senza *buildare* il gioco ogni volta. ParrelSync ha aiutato nella verifica delle principali funzionalità del gioco ma, riconoscendo di trovarsi sulla stessa macchina, duplicava alcune funzioni del gioco (come ad esempio i giocatori, che venivano istanziati due volte ciascuno) che sul momento sembravano funzionare; tuttavia, una volta *buildato* e testato il gioco con due PC (sia per Windows 10 che per Linux con Ubuntu 21.10), ho realizzato che, invece, si verificavano ancora molti problemi. Anche in questo caso la parte che ha richiesto molto tempo è stata modificare parti di codice e grafica per poi *buildare* il gioco e testarlo nuovamente. Il problema maggiore è stato sicuramente il riuscire a impostare un unico *seed* per entrambi i giocatori dato che con ParrelSync bastava utilizzare PlayerPrefs mentre con due macchine diverse il processo è più complesso. Un *seed* diverso faceva *spawnare* gli elementi in punti differenti per i due giocatori e non coordinava il suono della Tile, rendendo impossibile giocare.

Per aiutarmi col testing ho creato uno script (associato al Game Object ConsoleToGUI) che permette di avere una sorta di Developer Console anche durante il gioco e, per il giocatore cieco, ho impostato due tasti (Q e E) che permettessero di disattivare e riattivare il pannello nero che copre il labirinto.

---

[3] https://mirror-networking.com/

[4] https://github.com/VeriorPies/ParrelSync/releases/tag/1.5.0