DOCUMENTAZIONE PROGETTO DI LAM

Nome: Riccardo

Cognome: Lambertini

Email: riccardo.lambertini5@studio.unibo.it

Matricola: 0000934793

IDEA

Per il progetto assegnatoci avevo pensato di sviluppare un videogioco che rispecchiasse i miei gusti personali nell'ambito dei giochi mobile. Essendo un gande appassionato di giochi retro ho deciso di svilupparne uno su quella linea. Inoltre mi sarebbe piaciuto dare la possibilità all'utente di poter muovere a proprio piacimento il personaggio, di conseguenza ho escluso fin da subito un gioco di carte. Dopo varie idee ho scelto di sviluppare un gioco dove il player avrebbe dovuto competere con l'avversario in una gara di raccolta di monete cercando di avvantaggiarsi anche tramite alcuni power-up.

L'idea del gioco è molto semplice, come detto sopra ogni giocatore dovrà cercare di raccogliere il maggior numero di monete possibile entro un tempo limitato e dovrà stare anche attento a evitare alcuni oggetti che gli ostacoleranno il compito e cercare di prenderne altri che invece lo aiuteranno. Alla fine della gara verrà mostrato un punteggio dove verrà riassunto il numero di monete raccolte dal player1 (l'host) e dal player2 (il client) e il vincitore della sfida.

STRUMENTI

Per sviluppare un gioco simile ho deciso di sperimentare una tecnologia mai provata prima, ovvero, Unity2D. Unity è un motore grafico

sviluppato da Unity Technologies specializzato nello sviluppo di videogiochi mobile, computer e console. Ho scelto prorprio questa tecnologia anzichè AndroidStudio semplicemente per curiosità, per poter osservare e toccare con mano le potenzialità di Unity anche per ulteriori progetti futuri. Dato che le specifiche chiedevano un videogioco multigiocatore ho installato e importato all'interno di Unity, tramite l'asset store, una libreria di rete di alto livello di nome Mirror per poter lavorare con le meccanoche Client-Server. Per ultimo ho importato anche un'altra libreria di nome ParallelSync per poter clonare il progetto e far si che potesse simulare uno dei due giocatori.

FUNZIONALITA’

Come ho utilizzato Mirror

Ho utilizzato Mirror per la connessione Client-Server tramite delle soket specificando l’indirizzo IP dell’Host e il numero di porta (7777 di default). Mirror usa il protocollo Kcp in grado di ridurre la latenza media in modo migliore rispetto al protocollo precedente (LAN) dato che è stato progettato per la velocità di flusso cioè la quantità di tempo necessaria per inviare un pacchetto da una parte all’altra con uno spreco della larghezza di banda del 10%-20% in cambio di una velocità di trasmissione più veloce del 30%-40% rispetto a TCP. Tutto ciò si trova all’interno del componenete Kcp Tranport associato al gameObject NetworkManager.

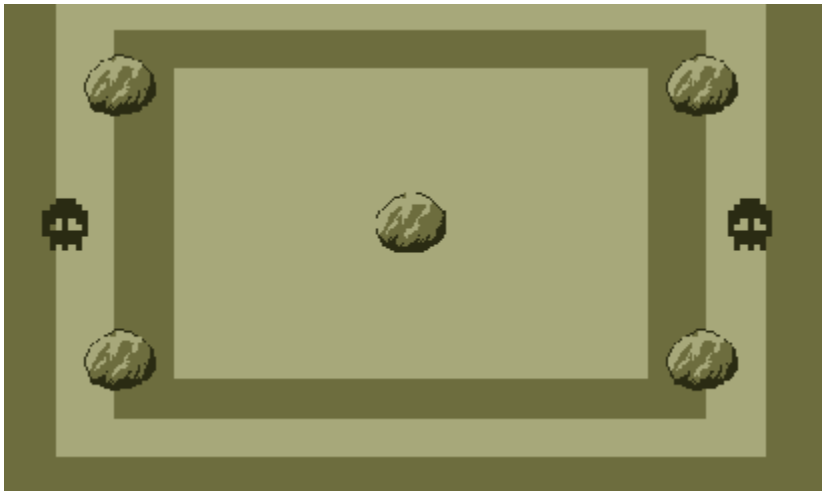
Prototipo1

Per prima cosa ho implementato la classe “NetworkManager” per gestire l’inizializzazione e lo spown dei giocatori e degli altri elementi creati al run-time come gli ostacoli. Il tutto viene eseguito all’interno della funzione “OnServerAddPlayer” che istanzia il nuovo giocatore che desidera entrare in partita facendolo spownare in una precisa posizione determinata da un gameObject già presente all’nterno della scena, mentre per quanto riguarda la posizione degli ostacoli essa viene inserita direttamente da codice. Questo metodo viene chiamato ogni volta che l’utente preme il tasto “Host” o il tasto “Join”. Per far si che un prefab possa spownare c’è bisogno che esso abbia almeo un componente chiamato “Network Identity” che permette di controllare l’identità univoca di un oggetto di gioco sulla rete e viene utilizzato per rendere il sistema consapevole dell’esistenza di quell’oggetto.

Dopo esser riuscito a far spownare i players e gli ostacoli il compito successivo era quello di riuscire a far muovere il personaggio. Per fare ciò ho creato lo script “Player” dove ho inserito dentro il metodo “FixedUpdate” (un metodo che funge da ciclo while(true)) un controllo sulla posizione del touch/mouse che facesse muovere il prorprio player verso quella posizione con una data velocità. Qui però si era creato un problema, ovvero, il player si muoveva ma l’altro giocatore non vedeva la risposta, di conseguenza vedeva l’avversario sempre fermo nel punto in cui era spownato. Risolvere questo è stato semplice, infatti è bastato aggiungere ai players un componente di nome “Network Transform” che permette di condividere la propria posizione e i propri movimenti. Inoltre essendo comandati dai giocatori ho dovuto segnare a true la casella “Client Autority” permettendo di mandare la posizione dal Client al Server.

Per ultimo ho aggiunto un box collider2D agli ostacoli e ai players più un rigid body2D (solo ai players) per evitare di poter passare attraverso le pietre e la zona di gioco.

Una volta implementate tutte queste funzionalità ho teminato il prototipo1.



Prototipo2

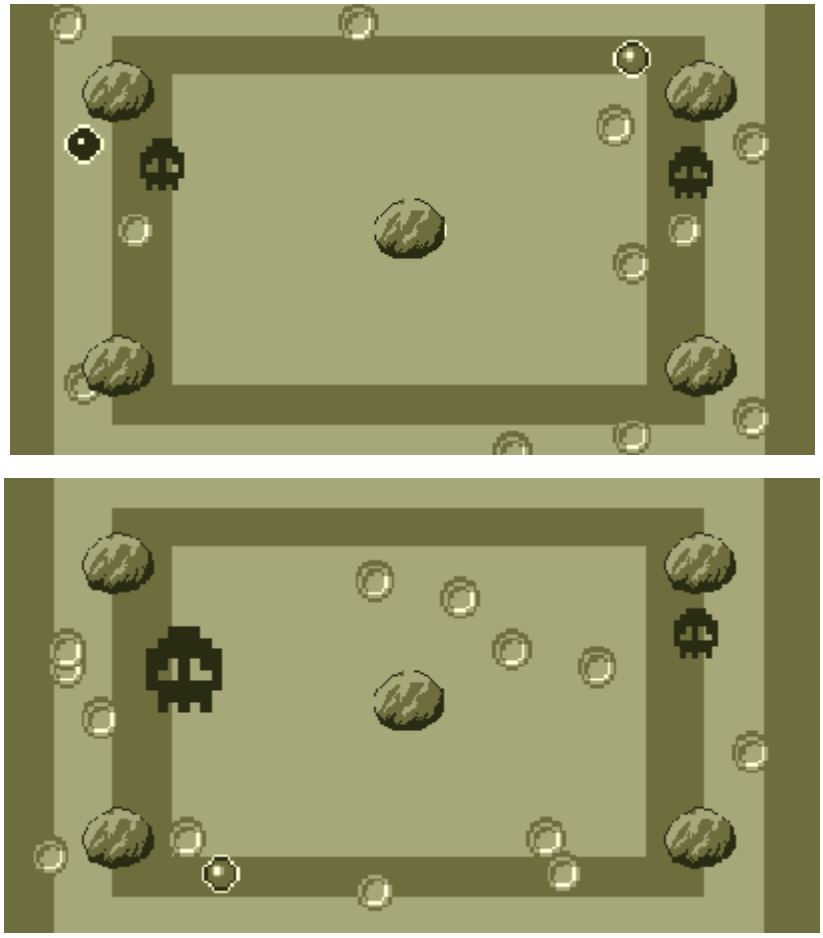
Per il prototipo2 l'obbiettivo era quello di implementare le meccaniche core del progetto, di conseguenza lo spown e la raccolta delle monete, dei power-up e l'aggiornamento del punteggio.

In primo luogo mi sono occupato delle monete, esse vengono fatte spownare di tre in tre in una zona random all'interno dell'area di gioco ogni tre secondi. Questo comportamento viene fatto tramite una Coroutine che dopo ogni tre secondi istanzia tre monete con posizioni random e setta un boolean a true per poter ricominciare. Questo comportamento comtinua finchè il tempo a disposizione è superiore a zero. Dopodiché ho creato lo script "GrabCoin" che ho associato alle monete per farle scomparire una volta entrate in contatto con un giocatore. Per poter cancellare una moneta sia dal Server che dal Client ho sfruttato dei metodi di Mirror per comunicare tra i due giocatori, ovvero, [Comand] e [ClientRpc]. Il comando [Comand] permette di far

eseguire il codice sul Server mentre il comando [ClientRpc] su tutti i Clients. Solitamente quindi per far si che un evento scatenato dal Clinet accada sia su esso che sul Server bisogna far eseguire il comportamento associato sia sul Clinet che (con un[Comand]) sul Server. Per concludere i comportamenti relativi alle monete ho implementato il sistema per aggiungere un punto al giocatore che ha toccato la moneta, tutto viene eseguito all'interno del metodo "ClaimPrize" dove prendo il giocatore in questione e gli aumento la sua variabile "score" di uno. La variabile "score" in questione è una variabile [SyncVar], ovvero, quando viene modificata anche gli altri giocatori vedono questa modifica. Questo comportamento va dal Server ai Clients. Inoltre ho aggiunto un ulteriore script al player di nome "PunteggioPlayer" che semplicemente disegna un rettangolo dove viene visualizzato il punteggio aggiornato del proprio player.

L'ultima cosa che ho implementato per concludere il prototipo2 sono stati i power-up che ho deciso di classificare in tre tipi diversi. Il primo power-up rende chi lo prende più piccolo, il secondo più grande e il terzo più veloce. Anche questi vengono fatti spownare dal Server ogni tot secondi in posizioni random come le moente ma singolarmente. Ho associato lo script "GrabPower" che consiste nel eliminare il potere quandi viene preso e di attivare l'effetto su chi lo ha toccato, anche in questo caso per poter ottenere lo stesso risultato visivo da entrambi i giocatori ho utilizzato i metodi [Comand] e [ClientRpc]. Il potere della velocità viene, diversaente dagli altri due, attivato dentro lo script "Player" in modo da rendere più semplice poi la terminazione dell' effetto. Per distinguere se un player ha ottenuto la super velocità modifico la sua trasparenza in modo impercettibile dal punto di vista del giocatore così da sapere se utilizzare la velocità normale oppure no. Dopo cinque secondi il player viene risettato alla grandezza naturale e al colore iniziale in modo tale da far terminare il potere. Dato che la

super velocità è il power-up più forte ho deciso di associargli una probabilità più bassa di poter essere spownato rispetto agli altri due.

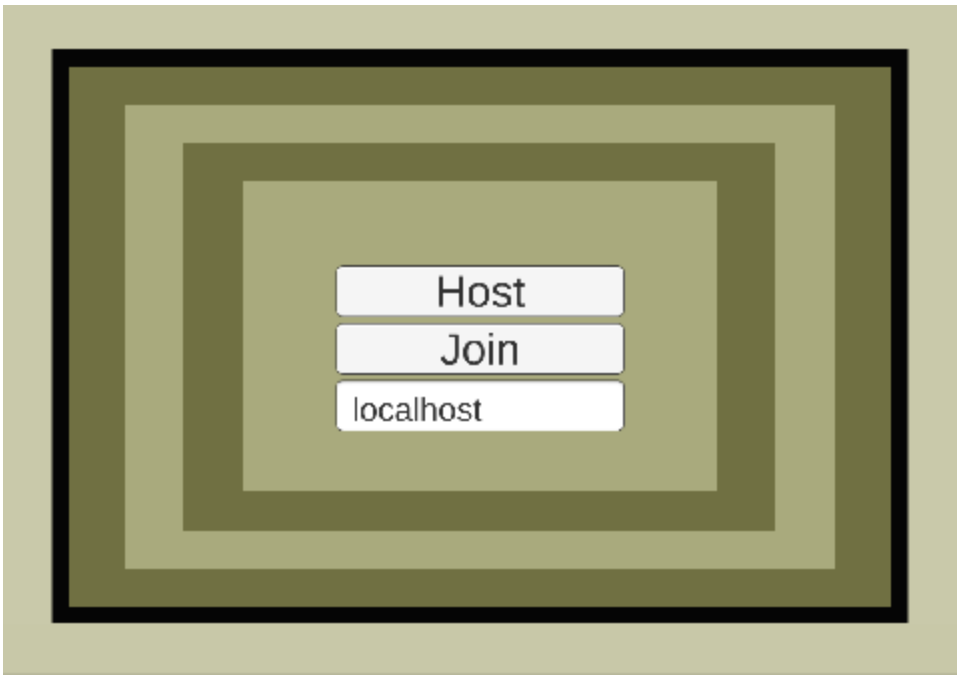


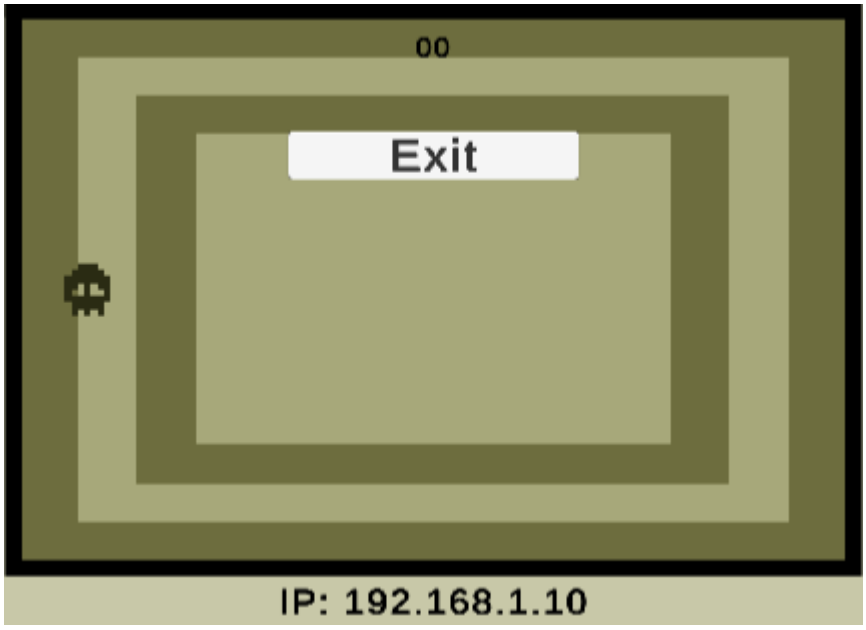
Prototipo3

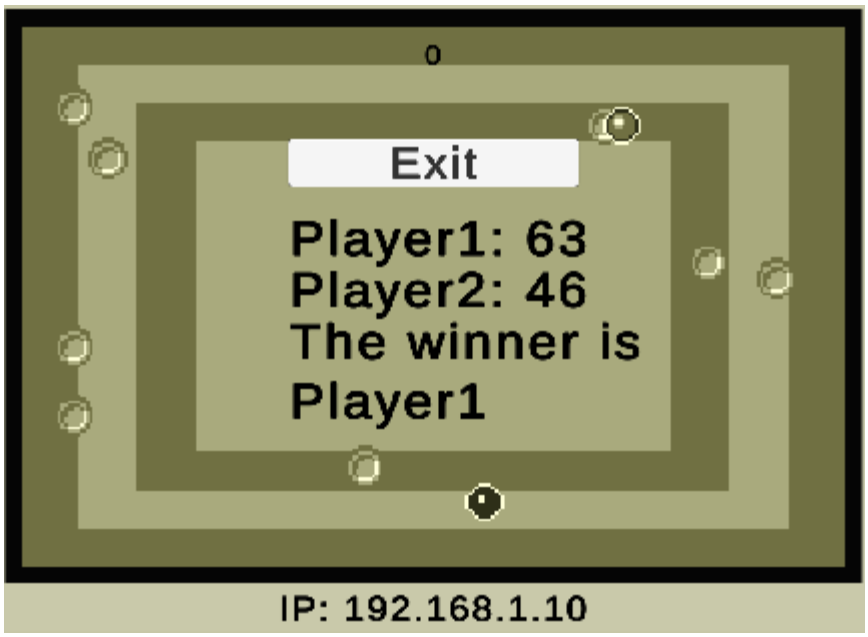
Quello che mancava ora al videogioco erano una schermata di inizio, un timer e una schermata di fine che mostrasse il risultato.

Per la schermata iniziale ho inserito nella scena dei bottoni per selezionare “Host”, “Join”, e un textbox per inserire l’IP dell’host. Tutti questi comportamenti sono inseriti all’interno dello script “Menu” associato al gameObject “NetworkManager”. Una volta che l’host entra

in partita viene mostrato in basso il suo IP da comunicare all'altro giocatore e viene messo in attesa. Durante questa attesa gli viene comunque data la possibilità di interagire con il proprio player. Quando anche il Client si unisce il gioco fa spownare tutti gli ostacoli, le monete e i power-up. Se nel mentre il giocatore 1 si trovasse in una posizione predestinata per lo spown di un ostacolo questo non causerebbe nessun errore in quanto verrebbe fatto spostare una volta spownato l'ostacolo a causa del suo rigid body2D e il suo box collider2D. Nelle impostazioni del componente "Kcp Transport" ho assegnato il valore 3000 al Timeout così da disconnetere il Client e far tornare l'host nel menù iniziale dopo tre secondi che il gioco viene messo in pausa da uno dei due giocatori. Questo comportamento è stato reso possibile grazie ai metodi "OnServerDisconnect" che mi permette di ricaricare la scena quando il Client si disconnette e "OnClientDisconnect" che permette di ricaricare la scena quando il Server si disconnette. Per quanto riguarda il tempo invece ho implementato un timer che parte da 120 e una volta arrivato a 0 il gioco termina cancellando i player e gli ostacoli. Tutto ciò accade nello script "Timer" dove viene anche mostrata la schermata di fine gioco con i punteggi, il nome del vincitore e un tasto "Exit" per poter tornare al menù iniziale per iniziare una nuova sfida. Ho implementato questo comportamento nel metodo "stop" dentro lo script "Menu" dove viene specificato di ricaricare la scena con il nome di quella corrente, quindi il risultato è una scena identica a quella precedente.


Host
Join
localhost


00
Exit
IP: 192.168.1.10



Prototipo4

Infine nel prototipo4 ho inserito le musiche e gli effetti sonori. I suoni delle monete e dei power-up li ho associati ai players dato che se li avessi associati alle monete non avrebbero effettuato nessun suono perchè sarebbero state eliminate. Per prendere i suoni corretti ho creato un array dove dentro il metodo “Start” (un metodo che viene chiamato appena lo script parte) prendo i componenti AudioSource e li associo uno alla variabile “coinAudio” e l’altra alla variabile “powerAudio”. Inoltre ho anche aggiunto un pulsante per togliere o rimettere i suoni, per fare ciò ho implementato uno scritp di nome “AudioManager” che setta a 0 oppure 1 il volume dell’audio del gioco. Quando un giocatore preme il tasto per togliere i suoni viene anche reso invisibile il pulsante stesso “StopAudio” e viene reso visibile il pulsante “PlayAudio” che un coportamento speculare.

VERSIONI

Per fare questo gioco per mobile ho dovuto installare Android Build Support che supporta tutti I dsipositivi android dalla versione 4.4 KitKat fino alla più recente.