**PROGETTO DI INFORMATICA**

FOREST RUNNER: ENDLESS ADVENTURE

Progetto svolto da: Paolo Bianchessi, Mattia Belloni, Tedi Shiku

Alunni della 4IC del IIS Galileo Galilei Crema

**STUDIO DI FATTIBILITA’**

* Siamo partiti dall’idea di creare un gioco senza un’idea molto precisa. Ci siamo imbattuti nel web e abbiamo ritrovato Subway Surfer un gioco che usavamo quando eravamo più piccoli. Da lì abbiamo deciso di creare la nostra versione con le nostre preferenze di gioco.
* REQUISITI: il personaggio deve poter eseguire tre movimenti 🡪 spostamento sx/dx, saltare e abbassarsi. Deve essere in grado di superare determinati ostacoli e poter sparare a dei nemici.

**PIANIFICAZIONE ATTIVITÀ**

* Prima settimana di lavoro: stesura della documentazione, abbiamo definito l’idea, scelto il linguaggio di programmazione (C#) e ci siamo divisi i compiti.
* Seconda settimana: continuiamo a lavorare in team al progetto, aggiungendo elementi alla nostra idea. Miglioria della logica del gioco e implementazione del miglioramento delle armi e della vita del personaggio.
* Terza settimana: verifica del funzionamento lato client e consegna del progetto.

**ANALISI DEI REQUISITI**

* La nostra applicazione sarà un videogioco utilizzabile su pc dal browser attraverso SIMMER.io.
* Il videogioco sarà endless runner, con possibilità di sparare a dei nemici. Lo scopo del gioco è quello di raccogliere più monete possibili, in moda da poter migliorare la propria arma e percorrere più strada possibile senza urtare gli oggetti presenti nel gioco.
* All’avvio sarà presente un menù con le opzioni PLAY – SETTING. Al click del bottone play si entrerà nella modalità di gioco, schiacciando setting si avrà la possibilità di modificare i livelli audio del gioco (volume generale / volume musica / volume effetti gioco).

**PROGETTO E REALIZZAZIONE**

* Il videogioco lo sviluppiamo attraverso Unity con il linguaggio C#
* Gli elementi grafici sono stati creata da noi attraverso l’Asset Store di Unity

(nemici: <https://assetstore.unity.com/packages/3d/characters/meshtint-free-boximon-fiery-mega-toon-series-153958>

Menù interazioni: <https://assetstore.unity.com/packages/2d/gui/icons/2d-casual-ui-hd-82080>

Bottoni: <https://assetstore.unity.com/packages/2d/gui/icons/simple-button-set-01-153979>

Monete: <https://assetstore.unity.com/packages/3d/props/simple-collectibles-pack-123092>

Armi: <https://assetstore.unity.com/packages/3d/props/guns/low-poly-fps-weapons-lite-245929>

Ambiente: <https://assetstore.unity.com/packages/3d/environments/landscapes/low-poly-simple-nature-pack-162153> )

**COMANDI PER POTER GIOCARE**

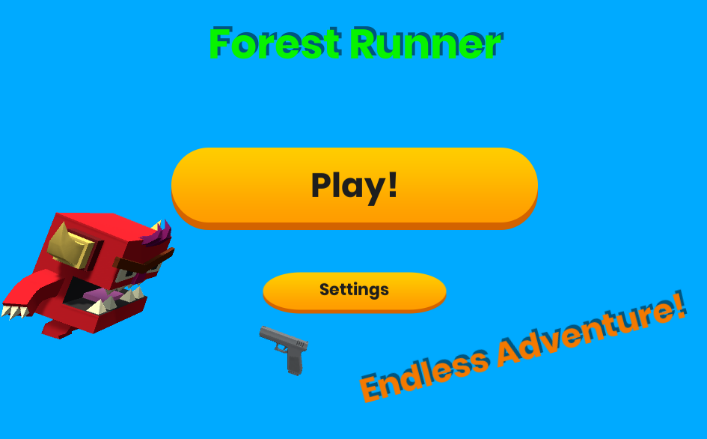
* P 🡪 Iniziare la partita
* W / freccia su 🡪 Saltare
* S / freccia giù 🡪 Abbassarsi
* A / freccia sinistra 🡪 Spostarsi verso SX
* D / freccia destra 🡪 Spostarsi verso DX

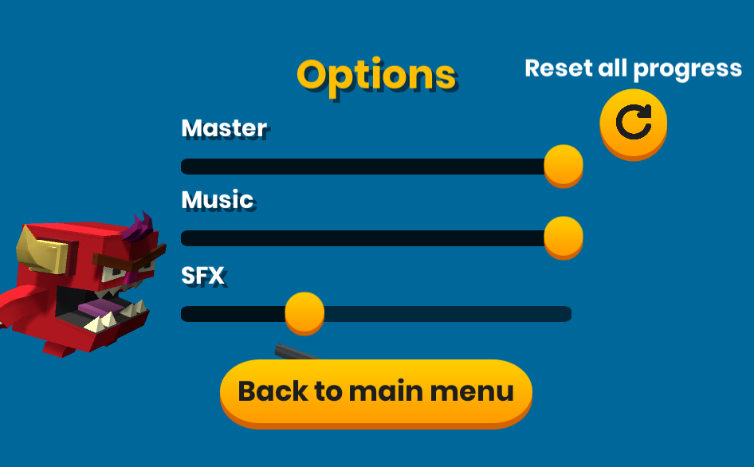
**TEST E MANUTENZIONE**

* Per testare il nostro gioco abbiamo deciso di proporlo ai nostri compagni di classe così che potessero restituirci un feedback e capire cosa si potesse migliorare.
* Durante il corso dell’anno usciranno delle versioni aggiornate del gioco con migliore delle armi e dei personaggi.

**SCREENSHOT DI GIOCO**

* Schermata del menù inziale



* ****Schermata menù audio
* ****Schermata prima della partita
* Schermata durante una partita



* Schermata GAME OVER



* Screen menù pausa



**DOCUMENTAZIONE CODICE**

**Nel nostro progetto ci sono 17 classi, ognuna di esse è “attaccata” al rispettivo GameObject nella scena Unity:**

* ***Gestione UI***
  1. CameraFollow.cs: questo ci script consente di seguire costantemente il giocatore, con un offset di dieci unità.
  2. GameUI.cs: gestisce tutte le interazioni con gli elementi grafici nella scena di gioco, avendo presuddiviso la UI in categorie (quella che si presenta prima della partita, quella durante la partita, la schermata del game over, quella di quando il gioco è in pausa) risulta molto semplice attivare/disattivare questi elementi.
  3. MainMenuUI.cs: stessa logica della classe precedente, usata nella scena iniziale.
  4. ModalDialog.cs: Quando vengono premuti i pulsanti di ritorno al menu e di reset dei progressi, viene presentato un menu di scelta (yes/no) per evitare possibili interazioni indesiderate.
  5. SettingsMenu.cs: sono presenti due funzioni: una per il reset dei progressi e l’altra per aprire il menu modale.
* ***Gestione Audio***
  1. AudioManager.cs: sono presenti due array di `Sound`, uno per le musiche e uno per i suoni di gioco, all’avvio del gioco, verranno “attaccate” al GameObject tante AudioSource quante sono le clip audio negli array. Sarà poi possibile riprodurre i suoni chiamando la funzione *play* da altri scripts.
  2. Sound.cs: è la modellizzazione di una clip audio, presenta varie variabili come *name(string), volume(float), loop(bool)*
* *Gestione Oggetti di gioco*
  1. GameManager.cs: è responsabile di tutto ciò che riguarda gli eventi di gioco: game over, gioco in pausa, gioco non ancora avviato; gestisce il punteggio e le monete. All’avvio del gioco recupera da una sorta di database chiamato *PlayerPrefs* il numero di monete e il punteggio record.
  2. CoinPickUp.cs: gestisce la collisione con il giocatore e aumenta il numero di monete.
  3. GroundSpawner.cs: genera i primi otto pezzi di terreno all’inizio del gioco.
  4. GroundTile.cs: rappresenta un tile di gioco, generato all’infinito.
  5. Obstacle.cs: gestisce le collisioni del giocatore con gli ostacoli (rocce, tronchi).
  6. EnemyController.cs: gestisce le animazioni, la vita e le collisioni dei nemici.
  7. PlayerController.cs: gestisce tutte le meccaniche del giocatore (corsa, salto, …), tutte le animazioni, e le collisioni con i nemici e con gli ostacoli.
* *Gestione arma*
  1. GunController.cs: gestisce il comportamento dell’arma del giocatore, è presente la funzione *shoot*, che genera un proiettile e lo accelera.
  2. BulletController.cs: muove il proiettile nel mondo di gioco, se arriva oltre il range e non colpisce nulla, viene distrutto.
  3. Purchase.cs: