

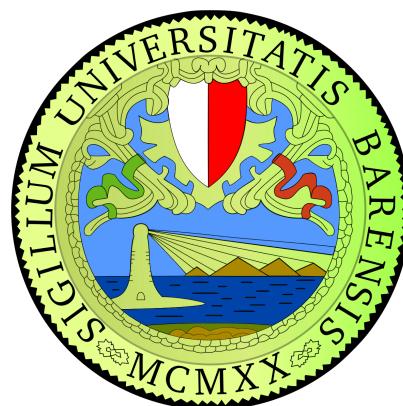
Università degli Studi di Bari Aldo Moro

Dipartimento di informatica

Corso di Laurea Triennale in Informatica

Generazione procedurale di contenuti in un videogioco rompicapo

Tesi di laurea in Sviluppo di Videogiochi



Laureando:
Gianluca Colella 719762

Relatore:
prof. Pierpaolo Basile

Anno accademico 2022-2023

Ringraziamenti

Vorrei ringraziare di cuore tutti i miei amici, che si sono sempre mostrati disponibili a effettuare test, i questionari di valutazione e a darmi sempre preziosi suggerimenti per migliorare il gioco e la tesi.

Un ringraziamento speciale va alla mia famiglia, che mi ha sostenuto in ogni fase del mio percorso di formazione. Ho sempre potuto contare sul loro sostegno, e so che continueranno a farlo, incoraggiandomi a dare sempre il massimo di me stesso.

Inoltre, desidero ringraziare il Professore Pierpaolo Basile per la sua disponibilità e il prezioso supporto fornito durante lo sviluppo del gioco e la stesura della tesi.

Indice

1	Introduzione	7
2	Stato dell'Arte	10
2.1	La storia della PCG	10
2.2	Classificazione degli algoritmi	13
2.3	Algoritmi basati su ricerca	14
2.3.1	Rappresentazione dei contenuti	14
2.3.2	Funzione di valutazione	15
2.4	Answer Set Programming	16
2.5	Applicazione della PCG	17
2.6	Esempio di funzione di valutazione	19
2.7	Altri esempi di applicazione	20
2.7.1	Evolutionary Dungeon Designer	20
2.7.2	Angry Birds level generator	21
3	Metodologia	25
3.1	Scopo del gioco	25
3.2	Meccaniche e UI di gioco	25
3.3	Menù di gioco	29
3.4	Struttura del gioco	30
3.5	Algoritmo utilizzato	32
4	Sviluppo del Lavoro	39
4.1	Tools utilizzati	39
4.1.1	Componenti utilizzate in Unity	41
4.2	Classificazione dell'algoritmo	45
4.3	Rappresentazione dei contenuti	46
4.4	Funzione di valutazione	46
5	Valutazione del Lavoro	47
5.1	Game Experience Questionnaire	47
5.1.1	Struttura del GEQ	48
5.2	Questionario	50

5.3	Risultati	51
6	Conclusioni	60
6.1	Sviluppi futuri	60

Elenco delle figure

2.1	Rogue	10
2.2	Elite	11
2.3	Dwarf Fortress	11
2.4	Minecraft	12
2.5	No Man's Sky	12
2.6	GeneratedMaze1	17
2.7	GeneratedMaze2	18
2.8	GeneratedMaze3	18
2.9	Binary tree maze	18
2.10	Binary tree	19
2.11	I dodici diversi blocchi disponibili	21
2.12	Un esempio di struttura completamente generata	24
3.1	Giocatore	25
3.2	Uscita	26
3.3	Muro	26
3.4	Tempo	26
3.5	Percorsi	26
3.6	Freccie direzionali	27
3.7	Immagini del movimento in game	27
3.8	Respawn button	27
3.9	Respawn avaible	27
3.10	Respawn not avaible	27
3.11	Pulsante di zoom	28
3.12	Tempo rimanente	28
3.13	Menù di gioco	29
3.14	Evento rotazione	30
3.15	Mosse disponibili	31
3.16	Stanza generata	33
3.17	Stanza generata con player e uscita	33
3.18	Coda	34
3.19	Albero	35
3.20	Algoritmo BFS	35

3.21 Stanza generata con indicati i tile esaminati	37
3.22 Stanza generata e controllata con segnati i muri aggiunti	37
3.23 Stanza finale	38
4.1 Aseprite	39
4.2 Unity	40
4.3 SFXR	40
4.4 Visual Studio Code	41
4.5 Tiles e tilemap	42
4.6 Esempio di collider2D[11]	44
5.1 Risultati del modulo centrale parte 1	52
5.2 Risultati del modulo centrale parte 2	53
5.3 Risultati del modulo post-gioco	54
5.4 Risposte della domanda 10 del modulo centrale	55
5.5 Risposte della domanda 13 del modulo centrale	56
5.6 Risposte della domanda 14 del modulo centrale	56

Capitolo 1

Introduzione

La generazione procedurale di contenuti (PCG) ha rivoluzionato l'industria dei videogiochi, offrendo nuove possibilità e sfide creative. La capacità di creare mappe, livelli, personaggi e oggetti in modo automatico e dinamico ha aperto le porte a esperienze di gioco uniche e sempre diverse. In questa tesi, esploreremo il campo della generazione procedurale di contenuti nei videogiochi, analizzando lo stato dell'arte e approfondendo alcuni esempi di giochi che ne fanno uso.

Nel secondo capitolo, esamineremo una selezione di giochi che sfruttano la generazione procedurale di contenuti. Studieremo come alcuni giochi abbiano implementato con successo la PCG per offrire mondi vasti, complessi e infinitamente rigiocabili. Successivamente approfondiremo il concetto di generazione procedurale di contenuti. Esploreremo gli algoritmi e le tecniche utilizzate per generare mappe e livelli in modo casuale e controllato.

Nei capitoli successivi, presenteremo Astro Maze, un puzzle-arcade game con elementi platform e roguelike, sviluppato come parte di questa tesi. Esploreremo le diverse caratteristiche del gioco, le sue meccaniche e la sua struttura generale.

Nel terzo capitolo, descriveremo dettagliatamente ogni elemento del gioco, dalle caratteristiche visive alle meccaniche di gioco uniche. Esploreremo gli algoritmi sviluppati per la generazione procedurale delle stanze nel gioco, analizzando come la PCG abbia contribuito a creare un'esperienza di gioco coinvolgente.

Il quarto capitolo, offre un'esaustiva panoramica sugli strumenti e le risorse impiegate nel corso del progetto e un'analisi dettagliata delle caratteristiche dell'algoritmo PCG impiegato nel progetto.

Il quinto, invece, si concentra sulla valutazione del gioco sviluppato. Attraverso l'utilizzo del questionario GEQ (Game Experience Questionnaire), abbiamo raccolto e analizzato le risposte degli utenti riguardo all'esperienza di gioco. Questa analisi ha consentito di comprendere come l'algoritmo PCG abbia influenzato l'esperienza complessiva del giocatore, identificando punti di forza e aree di miglioramento del gioco.

Nel capitolo conclusivo di questo lavoro di tesi, verrà fornita una profonda riflessione riguardante tutto il percorso di ricerca e analisi svolto, arricchita da considerazioni sulle possibili direzioni di sviluppo futuro.

Questa tesi tratterà numerosi algoritmi e altri contenuti tecnici, nonché discussioni sulla progettazione dei videogiochi. Prima di iniziare, è importante stabilire alcune definizioni chiave. Uno dei termini fondamentali è "contenuti", che comprendono tutti gli elementi presenti all'interno di un gioco: livelli, regole di gioco, personaggi, oggetti, missioni, musica, e altro ancora. Due concetti importanti da definire sono "procedurale" e "generazione", che implicano l'utilizzo di procedure informatiche o algoritmi per creare qualcosa di nuovo. Un metodo PCG (Procedural Content Generation) viene eseguito da un computer e genera un risultato specifico. Definire i giochi è un compito difficile (un approfondimento maggiore è riportato nel libro "Game definitions: A Wittgensteinian approach" [4]), ma possiamo affermare che con il termine gioco si comprende videogiochi, giochi da tavolo, giochi di carte, puzzle, e così via. È cruciale che il sistema di generazione dei contenuti tenga conto del design, delle possibilità e dei vincoli specifici del gioco per cui viene creato. Questa distinzione è ciò che differenzia il PCG da altre iniziative come l'arte generativa e vari tipi di grafica computerizzata, che non considerano necessariamente i vincoli e le possibilità del gioco stesso. In particolare, un requisito fondamentale per i contenuti generati è che siano giocabili, ossia che sia possibile completare un livello generato, attraversare una scala generata, utilizzare un'arma generata o vincere una partita generata. Di seguito viene presentata una lista di "metodi PCG".

- Un software che crea la mappa per un gioco d'azione e avventura come The Legend of Zelda senza l'intervento umano;
- Un sistema che crea nuove armi in un gioco di sparatutto spaziale in risposta al feedback dei giocatori, in modo che le armi presentate a un giocatore siano versioni evolute di armi che altri giocatori hanno trovato divertenti da usare;

- Un programma che generi autonomamente giochi da tavolo completi, giocabili ed equilibrati, magari prendendo come punto di partenza alcuni giochi da tavolo esistenti;
- Un motore di gioco che popoli rapidamente un mondo con fauna e flora;
- Un'intelligenza artificiale per un gioco da tavolo.[26];

Capitolo 2

Stato dell'Arte

2.1 La storia della PCG

La generazione procedurale di contenuti (PCG) consente la creazione automatica, all'interno di un videogioco, di mappe, personaggi, oggetti, e altro ancora. Il suo utilizzo è sempre aumentato nel corso degli anni perché permette di risparmiare tempo e preziose risorse. All'inizio, veniva utilizzata perché i computer e le console non disponevano della stessa quantità di memoria che i moderni calcolatori hanno oggi.

Vediamo come la generazione procedurale di contenuti abbia avuto un impatto nel mercato dei videogiochi, partendo da uno dei primi giochi che l'ha implementata, Rogue, fino ad arrivare a uno degli ultimi che ha suscitato maggiore interesse, No Man's Sky.



Figura 2.1: Rogue

Rogue è un videogioco per computer che si concentra sull'esplorazione del sottosuolo[24]. Si tratta di un gioco a turni, in cui l'azione si svolge su una griglia quadrata rappresentata in ASCII o in un altro set di caratteri fissi. Questo permette ai giocatori di avere il tempo necessario per determinare la mossa migliore al fine di sopravvivere. Rogue implementa il concetto di "permadeath" come scelta di design, rendendo ogni azione del giocatore significativa. Nessuna partita è uguale a quella precedente, poiché i livelli dei dungeon, gli incontri con i mostri e i tesori vengono generati proceduralmente per ogni partita[9].



Figura 2.2: Elite

Elite è un videogioco di commercio spaziale[14]. L'universo di Elite comprende otto galassie, ognuna con 256 pianeti da esplorare. A causa delle limitate capacità dei computer a 8 bit, questi mondi vengono generati proceduralmente. Vengono salvate solo una serie di numeri che identificano la galassia e i pianeti, e successivamente, una volta elaborati dall'algoritmo, vengono ottenute tutte le informazioni dettagliate[9].



Figura 2.3: Dwarf Fortress

Dwarf Fortress è un videogioco freeware ambientato in un universo fantasy che combina i generi roguelike e di costruzione[13]. Il mondo di gioco è generato proceduralmente, e il giocatore controlla indirettamente un gruppo di nani, cercando di costruire una fortezza solida e ricca. La critica ha elogiato il gameplay complesso ed emergente del gioco, ma ci sono state reazioni contrastanti riguardo alla sua difficoltà. Dwarf Fortress ha influenzato giochi come Minecraft, Rimworld e altri, ed è stato selezionato per essere esposto al Museum of Modern Art nel 2012, come parte della mostra sulla storia dei videogiochi[1].



Figura 2.4: Minecraft

Minecraft, invece, è un videogioco d'avventura sandbox ambientato in un mondo tridimensionale composto da blocchi[19]. Tutto l'ambiente di gioco, quindi foreste, deserti, grotte e altri biomi, vengono generati proceduralmente prima che il giocatore lo avvii per la prima volta.(in modo non dissimile da Dwarf Fortress). Dopo che gli intricati paesaggi del proprio mondo di Minecraft sono stati generati, il giocatore parte all'avventura da una posizione casuale nella mappa.[12].



Figura 2.5: No Man's Sky

No Man's Sky, invece, è un videogioco action-adventure[20]. Gli sviluppatori hanno cercato di soddisfare la grande richiesta da parte dei giocatori di nuovi paesaggi e caratteristiche complesse, automatizzando il processo di progettazione degli ambienti di gioco. Attraverso la generazione procedurale, il gioco offre un terreno di gioco incredibilmente vasto con pianeti, flora, fauna, atmosfere e altro ancora. I giocatori possono viaggiare verso qualsiasi dei 18 quintilioni di pianeti, a condizione di avere le risorse necessarie per la sopravvivenza. Il sistema di generazione assicura che se un giocatore visita un pianeta già scoperto, troverà lo stesso pianeta con le stesse caratteristiche, e tutto ciò non richiede ulteriori spazi di memorizzazione grazie alla generazione procedurale[29].

2.2 Classificazione degli algoritmi

Gli algoritmi di PCG assumono diverse forme in base all'utilizzo che bisogna farne. Non esiste una ricetta per ottenere il miglior algoritmo, ma bisogna conoscerne le differenze per capire quale sarà il migliore per il raggiungimento dell'obiettivo.

- **ONLINE vs OFFLINE:** I contenuti vengono generati online mentre il gioco è in runtime, oppure offline durante lo sviluppo di esso.
- **NECESSARI vs OPZIONALI:** I contenuti necessari per la progressione in gioco devono essere sempre corretti. Non è tollerabile che venga creato un contenuto che non permette la progressione in gioco. Invece quelli opzionali possono anche non funzionare come ci si aspetta, l'importante che è che non portino alla rottura del gameplay.
- **RANDOM SEEDS vs PARAMETER VECTORS:** Un algoritmo di PCG potrebbe prendere un semplice numero generato casualmente come input (seed) per la rappresentazione del contenuto generato, oppure potrebbe prendere un vettore multidimensionale di reali parametri che specificano diverse proprietà dei contenuti creati. Per esempio, un generatore di dungeon potrebbe prendere come parametro il numero di stanze.
- **GENERAZIONE STOCASTICA vs GENERAZIONE DETERMINISTICA:** Quanto la casualità è impattante nella generazione di contenuti? Quanto una sessione di gioco con parametri uguali deve essere diversa da un'altra? Queste sono scelte di design, è difficile dare una risposta giusta. Prendendo l'esempio di un generatore di dungeon in un gioco rogue-like la casualità è importantissima. (Da notare che il seed, il numero generato casualmente in input, non è considerato un parametro qui, quindi in pratica tutti gli algoritmi sono deterministici)

- CONSTRUCTIVE vs GENERATE-AND-TEST: Gli algoritmi "constructive" generano il contenuto, facendo attenzione che avvenga tutto correttamente, o che almeno, rispetti un minimo standard. Questo permette di evitare la creazione di contenuti che portano ad uno stato di blocco del gioco. Un algoritmo invece "generate-and-test", dopo che un contenuto è stato generato, esso diventa un candidato. Successivamente viene sottoposto ad un test (c'è un percorso fra l'entrata e l'uscita del dungeon?). Se il test fallisce il candidato viene scaricato o rigenerato e il processo continua finché il contenuto non sarà accettabile.

2.3 Algoritmi basati su ricerca

Prima di tutto, è importante precisare che con la definizione di "basati su ricerca" si intendono esplicitamente tutte le forme di algoritmi di ricerca/ottimizzazione euristiche e stocastici. Questi algoritmi rappresentano solo uno dei modi per affrontare la generazione procedurale di contenuti, seguendo l'approccio "generate-and-test". La funzione di test non si limita a accettare o rifiutare direttamente il candidato, ma lo valuta utilizzando un numero reale o un vettore di numeri reali, e questa funzione prende il nome di "Fitness". La generazione di nuovi contenuti dipenderà dalla valutazione ottenuta in precedenza.

In altre parole, nell'ambito della generazione procedurale di contenuti, i metodi basati su ricerca includono algoritmi di ricerca e di ottimizzazione, che cercano di generare nuovi contenuti sfruttando strategie euristiche e stocastiche. Questi algoritmi generano un candidato di contenuto, che viene quindi valutato tramite una funzione di fitness. La funzione di fitness assegna un valore numerico al candidato, riflettendo la sua compatibilità o qualità rispetto agli obiettivi del gioco.

È importante notare che la funzione di fitness non si limita a dare un semplice "sì" o "no" al candidato, ma produce una valutazione quantitativa. Questo valore di fitness influenzerà la generazione futura di nuovi contenuti, poiché il processo di generazione cercherà di ottimizzare le soluzioni, avvicinandosi a risultati più desiderabili sulla base della valutazione ottenuta.

2.3.1 Rappresentazione dei contenuti

Come vengono rappresentati i contenuti all'interno del processo di generazione? Innanzitutto, è importante chiarire la differenza tra genotipo e

fenotipo. Il genotipo è una struttura di dati presa dalla funzione di fitness ed è un vettore di parametri che contiene le informazioni di base del contenuto da creare. Il fenotipo, invece, è il prodotto creato finale, la struttura di dati che rappresenta il contenuto valutato dalla funzione. In altre parole, il genotipo è come uno stampino, mentre il fenotipo è il prodotto finito.

Un'altra distinzione importante è tra codifica diretta e indiretta. Nel caso della codifica diretta, c'è una relativa semplicità computazionale nella mappatura genotipo-fenotipo, ovvero la dimensione del genotipo è linearmente proporzionale a quella del fenotipo. Invece, nel caso della codifica indiretta, le dimensioni del genotipo e del fenotipo non sono proporzionali, e spesso è necessario un calcolo complesso per la creazione del fenotipo a partire dal genotipo.

In altre parole, il genotipo rappresenta la configurazione di base, una serie di parametri che definiscono il contenuto da generare, mentre il fenotipo è il risultato concreto di tali parametri, ovvero il contenuto generato effettivamente. La codifica diretta semplifica il processo di generazione, poiché il genotipo e il fenotipo hanno una corrispondenza diretta e immediata. D'altro canto, la codifica indiretta potrebbe richiedere un calcolo più elaborato o una procedura specifica per trasformare il genotipo nel fenotipo desiderato.

2.3.2 Funzione di valutazione

Una volta che il contenuto è stato generato, deve essere valutato da una funzione di fitness, assegnandogli uno scalare o un vettore di numeri reali che riflette la sua compatibilità con il gioco. Nella Procedural Content Generation (PCG) vengono utilizzati tre tipi di funzioni di valutazione: diretta, basata sulla simulazione e interattiva. La scelta dell'approccio più adeguato dipenderà dall'obiettivo specifico del gioco e dalle caratteristiche dei contenuti generati e ogni tipo può essere implementato in vari modi, a seconda delle esigenze del gioco e delle risorse disponibili.

Nel metodo diretto, alcuni dati sono estratti dal contenuto generato e utilizzati per calcolare il valore di fitness. Ad esempio, potrebbero essere considerati i numeri di percorsi per raggiungere l'uscita in un labirinto, la velocità di fuoco in un gioco di tipo sparatutto in prima persona (FPS), la quantità di risorse in un gioco di strategia in tempo reale (RTS), e così via. Questi dati possono essere utilizzati in modo lineare o non lineare per la valutazione, e solitamente non richiedono un'elevata potenza computazionale.

Tuttavia, non sempre è semplice trovare dati significativi da utilizzare nella valutazione. In questi casi, si potrebbe optare per il metodo basato sulla simulazione. In questo approccio, si utilizza un agente artificiale per giocare il contenuto da valutare, e si estraggono dati dal suo gameplay per calcolare il valore di fitness. Tali dati possono essere statici o dinamici: se l'agente rimane invariato durante tutto il gioco, si parla di una valutazione statica; se invece l'agente evolve nel tempo, magari imparando dagli errori, la funzione di valutazione dovrà tener conto di questa evoluzione.

Un terzo approccio è quello delle funzioni di valutazione interattive, in cui i dati vengono raccolti durante il gameplay di un giocatore. Questi dati possono essere ottenuti attraverso questionari o misurando implicitamente alcune interazioni del giocatore. Tuttavia, il problema di questo metodo è che spesso i dati risultano imprecisi e possono interrompere il flusso di gioco se non raccolti con attenzione[33].

2.4 Answer Set Programming

L'Answer Set Programming (ASP) è una forma di programmazione logica di tipo dichiarativo utilizzata per risolvere problemi di ricerca complessi, in cui il programma descrive i requisiti per la soluzione di un problema[3]. Il software scritto in AnsProlog è composto da un insieme di regole e predicati che vengono eseguiti per produrre un elenco di tutte le possibili soluzioni, chiamate "insiemi di risposte".

Rispettando le buone pratiche di ingegneria del software, un programma AnsProlog è codificato in quattro sezioni distinte. La sezione di definizione contiene tutti i fatti e le regole noti del sistema. La sezione di generazione viene utilizzata per generare i fatti derivati dal sistema. La sezione di test è dove vengono applicati i vincoli di integrità per eliminare le soluzioni indesiderate. Infine, la sezione di valutazione è dove avviene la valutazione del sistema.

Il linguaggio AnsProlog realizza questi costrutti con regole, fatti e vincoli di integrità. Una regola è definita come "head :- body". La testa rappresenta la regola da definire, mentre il corpo definisce i componenti logici che soddisfano la regola. Un fatto è definito come "head" e rappresenta qualcosa che è noto in modo incondizionato, è cioè una regola senza corpo. Un vincolo di integrità è definito come ":- body" e rappresenta tutte le cose che si sa che non sono vere, anch'esso è una regola senza testa.

Un utile costrutto di AnsProlog è la "regola di scelta", rappresentata come "lruleu", che fornisce un modo conveniente per selezionare in modo casuale un numero di regole dall'insieme delle regole valide, il cui numero è compreso tra l'intervallo l e u . Questa regola è particolarmente utile quando si desidera introdurre una componente di casualità nelle soluzioni generate dal programma[27].

2.5 Applicazione della PCG

La generazione procedurale di una mappa deve essere sviluppata in base al tipo di gioco che si vuole creare. Ad esempio, un dungeon viene creato in modo diverso rispetto a un puzzle. Supponiamo di voler creare un labirinto attraverso un algoritmo di generazione procedurale dei contenuti (PCG). Innanzitutto, avremo bisogno di una griglia che ci aiuti a definire i vari passaggi che la mappa avrà. Uno degli algoritmi più semplici in questo caso è il "Binary Tree". Partendo da una qualsiasi cella, dobbiamo visitarla tutte e, per ognuna, lanciare una moneta e decidere dove aprire la strada in base al risultato.

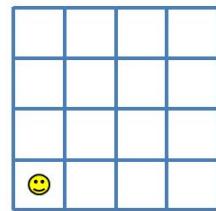


Figura 2.6: GeneratedMaze1

Ad esempio, partiamo dalla cella più in basso e ipotizziamo che se esce testa andremo sopra, mentre se esce croce andremo a destra. Supponiamo che esca testa, quindi apriamo la strada sopra.

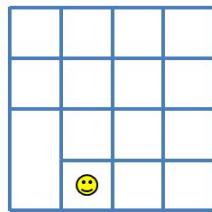


Figura 2.7: GeneratedMaze2

Ripetiamo questa operazione per ogni cella successiva, cancellando il muro in base al risultato del lancio della moneta. Vicino al bordo destro e superiore non sarà necessario lanciare la moneta perché non possiamo aprire il labirinto se l'uscita non è effettivamente presente. Quindi, semplicemente apriremo il bordo destro per la strada sopra e il bordo superiore per quella a destra.

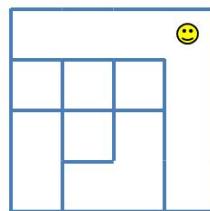


Figura 2.8: GeneratedMaze3

Una volta completata questa operazione di lancio della moneta per tutta la griglia, il risultato sarà simile a quello mostrato nella figura.

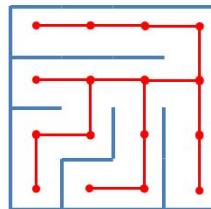


Figura 2.9: Binary tree maze

Come possiamo notare, il labirinto finale altro non è che un albero se lo ruotiamo di 45 gradi.

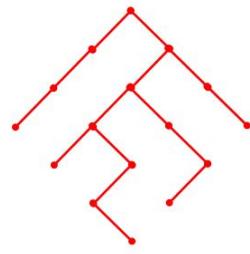


Figura 2.10: Binary tree

La generazione del labirinto è stata piuttosto semplice, ma è importante notare che se utilizzassimo questo algoritmo per un numero infinito di stanze, tutte le stanze avrebbero una particolarità in comune: i bordi destro e superiore sarebbero lunghi corridoi senza muri. È comune osservare pattern che si ripetono nello stesso algoritmo di generazione di un labirinto. Un altro algoritmo molto utilizzato è chiamato "Sidewinder". È molto simile al Binary Tree, ma quando esce croce, si sceglie casualmente una delle celle raggiunte fino a quel momento e si apre la strada sopra in quella cella. Questo algoritmo non presenta un lungo corridoio a destra, ma il corridoio superiore sarà ancora presente[8].

2.6 Esempio di funzione di valutazione

Per valutare la difficoltà di un gioco basato su un labirinto, può essere utilizzato l'algoritmo di Dijkstra. Questo algoritmo, ampiamente conosciuto e utilizzato, consente di determinare la distanza da un punto del labirinto a ogni altro punto interno.

Il processo inizia selezionando un punto di partenza, di solito l'ingresso del labirinto. Tutte le celle vengono impostate inizialmente a una distanza di zero, mentre la cella di partenza ha distanza zero. Successivamente, vengono esaminate le celle adiacenti non bloccate da muri e vengono impostate alla distanza precedente + 1, quindi a 1. Questo insieme di celle appena visitate viene chiamato "frontier". Si ripete il processo con le celle della "frontier", assegnando loro la distanza incrementata di uno rispetto alla distanza delle celle precedenti. Questo procedimento viene ripetuto fino a quando tutte le celle sono state visitate.

In questo modo, otterremo la distanza dal punto di partenza a ogni cella presente nel labirinto e potremo utilizzare queste informazioni per calcolare

la difficoltà del gioco. Ad esempio, possiamo considerare la distanza minima tra il punto di partenza e l'uscita come un indicatore della difficoltà del labirinto. Questo valore può essere utilizzato per impostare la difficoltà del gioco in modo che i giocatori si trovino di fronte a sfide più complesse o semplici in base alla lunghezza del percorso minimo.

L'utilizzo dell'algoritmo di Dijkstra ci consente quindi di valutare e regolare la difficoltà del labirinto in modo strutturato e basato sulla distanza dei punti nel gioco.

2.7 Altri esempi di applicazione

Cerchiamo di mettere a fuoco più da vicino un algoritmo che sfrutta la PCG per un possibile gioco roguelike e puzzle.

2.7.1 Evolutionary Dungeon Designer

L'EDD è un progetto che si propone di generare contenuti diversificati per ciascuna delle stanze presenti nel dungeon, ispirandosi alla struttura della camera nel famoso gioco "The Legend of Zelda". La camera è rappresentata come una griglia rettangolare di dimensioni $m \times n$, composta da 6 tipi di elementi: pavimento, muro, nemico, tesoro, entrata e porta. Tra questi, i muri sono gli unici elementi che bloccano il movimento del giocatore.

Per essere considerata una stanza valida, questa deve contenere obbligatoriamente un tesoro, un nemico e un passaggio che collega l'entrata a tutti i nemici, tesori e porte presenti all'interno della stanza.

La codifica di ogni stanza avviene attraverso un array bidimensionale $m \times n$ di numeri interi, dove ciascun numero intero (compreso tra 0 e 5) rappresenta uno specifico tipo di elemento.

L'algoritmo di evoluzione del progetto prevede l'uso di due popolazioni di individui che evolvono parallelamente: una popolazione per gli individui realizzabili (ossia, stanze che rispettano i requisiti di validità) e un'altra per gli individui non realizzabili. Ciascuna di queste popolazioni evolve attraverso un algoritmo distinto, con proprie operazioni di valutazione, selezione, crossover, mutazione e sostituzione.

Il crossover (incrocio) tra le due popolazioni mira a favorire la presenza di individui realizzabili e a incrementare la diversità all'interno della popolazione di stanze valide.

Per l’evoluzione, viene utilizzato un algoritmo FI-2Pop GA, in cui si applica un crossover a due punti e un alto tasso di mutazione (90). Ogni mutazione ha una probabilità del 20/100 di ruotare la stanza di 180°, mentre ha una probabilità dell’80/100 di mutare un singolo elemento della stanza in un tipo di elemento diverso e casuale. La mutazione che coinvolge la rotazione serve a favorire la comparsa di caratteristiche simmetriche nelle mappe generate.

Per stabilire il termine dell’algoritmo, si fissa un numero predeterminato di generazioni, evitando di basarsi su una soglia di fitness. Ciò garantisce un tempo di risposta consistente tra diversi set di parametri di input.

Per valutare la qualità delle stanze generate, vengono valutati inizialmente singolarmente i nemici, i tesori e i pattern delle stanze. Successivamente, viene combinato il valore di qualità di ciascuno di questi elementi per determinare la qualità finale della camera.[6]

2.7.2 Angry Birds level generator

I livelli di Angry Birds sono costituiti da diversi componenti: sul lato sinistro del livello c’è una fionda e una serie di uccelli che possono essere lanciati da essa. Sul lato destro ci sono vari blocchi, piattaforme e maiali, di solito disposti in un disegno interessante.

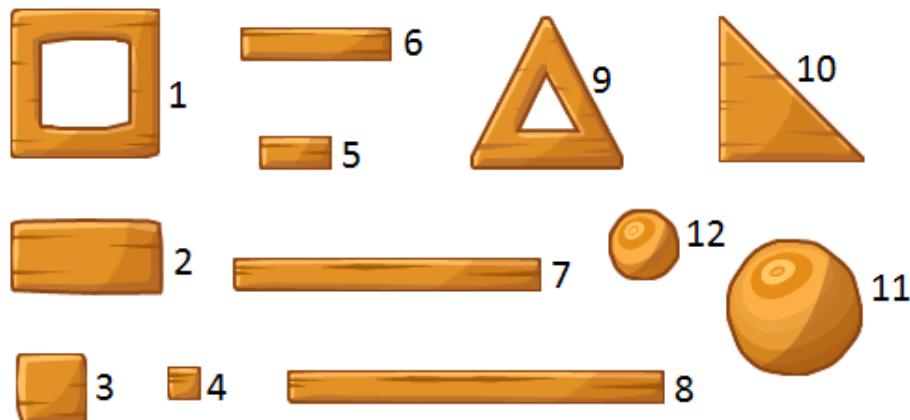


Figura 2.11: I dodici diversi blocchi disponibili

L’obiettivo principale di ogni livello consiste nell’eliminare tutti i maiali utilizzando gli uccelli a disposizione. Attualmente, il codice sorgente del gioco originale di Angry Birds non è disponibile. Pertanto, per questa versione, è stato utilizzato un clone basato su Unity creato da Lucas Ferreira (Ferreira e Toledo 2014b). Prima di descrivere l’approccio dell’algoritmo analizzato, è necessario definire alcuni termini che saranno utilizzati nel corso di questo articolo.

Un ”blocco” è un qualsiasi oggetto all’interno del livello che può essere spostato, ad eccezione degli uccelli e dei maiali. Attualmente, sono presenti dodici tipi di blocchi diversi nel clone di Unity. I primi otto blocchi sono considerati ”regolari”, mentre gli ultimi quattro sono definiti ”irregolari”. Una ”piattaforma” è una superficie fissa all’interno del livello, diversa dal terreno, che serve da supporto. Inoltre, introduciamo il concetto di ”spazio di livello”, che rappresenta un’area predefinita all’interno del livello in cui è possibile posizionare blocchi, piattaforme e maiali. Questo spazio di livello è stato progettato per evitare che gli oggetti vengano collocati troppo vicini alla fionda, al suolo o al di fuori della visuale della telecamera. Le posizioni della fionda e del terreno sono fisse all’interno di ogni livello, mentre gli altri oggetti vengono posizionati rispetto a queste due coordinate.

Il generatore di livelli proposto crea livelli di Angry Birds che consistono in un insieme di strutture indipendenti. Queste strutture vengono distribuite all’interno dello spazio disponibile nel livello, sia a terra che su piattaforme galleggianti. Il numero di strutture a terra e di piattaforme può essere determinato manualmente o selezionato casualmente. Prima di discutere la disposizione di queste strutture all’interno del livello, è necessario spiegare come vengono generate.

Nel nostro generatore di livelli, è presente un modulo autonomo che si occupa di creare le strutture complesse utilizzando gli otto blocchi regolari disponibili. Cinque di questi blocchi (2, 5, 6, 7 e 8) possono essere anche ruotati di 90 gradi per ottenere una forma diversa. Ciò porta a un totale di 13 tipi di blocchi regolari differenti. Le strutture generate dal nostro algoritmo sono composte da righe, dove ogni riga è formata da un singolo tipo di blocco. Ogni blocco viene anche assegnato casualmente uno dei tre possibili materiali: legno, ghiaccio o pietra. Ogni struttura viene generata in modo da rientrare in una specifica area all’interno del livello complessivo. Questa area viene utilizzata per definire i limiti massimi di larghezza e altezza che la struttura generata può occupare.

Per determinare la probabilità di selezione di un determinato tipo di

blocco, viene utilizzata una tabella di probabilità. Ogni tipo di blocco viene assegnato a una specifica probabilità di selezione, in modo che la somma delle probabilità sia uguale a uno. Questo tipo di blocco diventa il "picco" o i "picchi" della struttura, sotto i quali vengono posizionati gli altri blocchi. È possibile scegliere manualmente o automaticamente il numero di picchi e la distanza tra di essi. Tuttavia, se l'area occupata da tutti i blocchi dei picchi non rispetta i limiti massimi di larghezza o altezza della struttura, la combinazione di picchi viene considerata non valida e si procede a selezionare una nuova configurazione.

Dopo l'inizializzazione della prima riga, vengono create in modo ricorsivo altre righe di blocchi che verranno posizionate sotto la struttura già generata. I blocchi nella fila di base vengono suddivisi in sottoinsiemi in base alle distanze tra di essi. Vengono registrate tutte le possibili combinazioni di sottoinsiemi, come mostrato nella Figura 2. Successivamente, il tipo di blocco per la nuova riga viene selezionato utilizzando la tabella delle probabilità. Per ogni combinazione di sottoinsiemi, esistono tre possibili modi di posizionare i blocchi di supporto: sotto il centro di ogni sottoinsieme, sotto i bordi di ogni sottoinsieme o sotto i punti medi tra il centro e i bordi di ogni sottoinsieme. Queste tre scelte possono essere combinate per ottenere un totale di sette opzioni diverse. Ogni opzione viene creata per tutte le combinazioni di sottoinsiemi utilizzando il tipo di blocco selezionato, e successivamente viene verificata la validità della combinazione.

Qualsiasi situazione in cui i blocchi si sovrappongono viene considerata invalida e viene riposizionata come possibile alternativa. È fondamentale che i blocchi nella parte inferiore della struttura già generata siano sostenuti dalla nuova riga di blocchi. Il livello di supporto richiesto può essere impostato su una delle tre opzioni. La prima opzione richiede che ogni blocco sia supportato nella sua posizione centrale o in entrambe le posizioni ai bordi. La seconda opzione richiede che ogni blocco sia supportato in entrambe le posizioni ai bordi. La terza opzione richiede che ogni blocco sia supportato nella sua posizione centrale e in entrambe le posizioni ai bordi. Dopo aver effettuato i controlli di validità per tutte le combinazioni di sottoinsiemi e le posizioni di supporto dei blocchi, viene selezionata casualmente un'opzione valida tra tutte le possibilità. Se non ci sono opzioni valide disponibili, viene scelto un nuovo tipo di blocco e il processo viene ripetuto. L'opzione selezionata viene utilizzata come nuova fila inferiore della struttura. Questo processo continua finché la larghezza o l'altezza della struttura non superano i valori massimi specificati. A quel punto, viene rimossa l'ultima riga aggiunta e la struttura è completata. Questo processo garantisce che la struttura generata si adatti alle dimensioni specificate[28].

Un esempio di struttura generata completa è mostrato nella figura seguente.



Figura 2.12: Un esempio di struttura completamente generata

Capitolo 3

Metodologia

3.1 Scopo del gioco

Astro Maze è un puzzle-arcade game con elementi platform e roguelike e sfida il giocatore a risolvere più stanze possibili col minor tempo possibile. Il gioco presenta tematiche neon-spaziali, il giocatore è stato intrappolato in questo labirinto di stanze immenso, tutte generate proceduralmente, riuscirà ad uscire prima che il buco nero risucchi il labirinto?

3.2 Meccaniche e UI di gioco

Player

Questo cubo non è altro che il giocatore, dovrai muovere lui per trovare l'uscita dal labirinto.



Figura 3.1: Giocatore

Exit

L'uscita di una delle tante stanze nel labirinto.



Figura 3.2: Uscita

Muro, checkpoint e percorsi

Durante il tuo tragitto potresti incontrare questi tre tipi base di tiles.

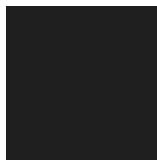


Figura 3.3:
Muro

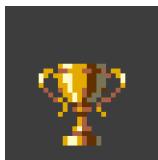


Figura 3.4:
Tempo



Figura 3.5:
Percorsi

La figura 3.3 raffigura il muro standard, ti fermerai davanti ad esso.

La figura 3.4 raffigura un piccolo trofeo, passandoci sopra otterrai del tempo extra, utile per averne di più per la risoluzione del puzzle. Inoltre puoi collezionarli, più ne otterrai più il tuo punteggio aumenterà.

La figura 3.5 mostra i percorsi attraverso il quale il tuo cubo potrà muoversi. Aiuta il giocatore ad avere meglio in mente tutti i possibili percorsi disponibili e scegliere il migliore per arrivare all'uscita.

Freccie direzionali

Come sappiamo nello spazio profondo l'attrito è assente, quindi la meccanica di movimento è stata sviluppata seguendo questo incipit. Attraverso i pulsanti presenti su schermo è possibile dire la direzione dove si è diretti, seguendo i 2 assi, quindi rispettivamente sopra, sotto, sinistra e destra.

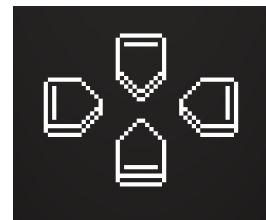


Figura 3.6: Freccie direzionali

Come detto precedentemente non esiste l'attrito quindi solo un muro fermerà l'avanzamento nella nuova posizione, una volta fermatosi il giocatore potrà procedere con il prossimo movimento.

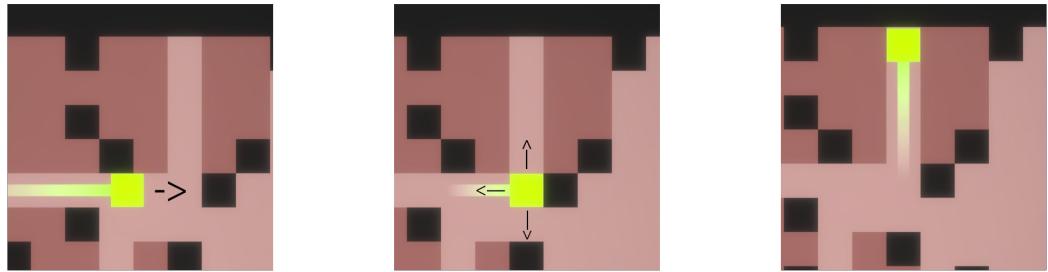


Figura 3.7: Immagini del movimento in game

Come possiamo notare nella figura 3.7 nella prima immagine il giocatore sta procedendo verso destra, nella seconda immagine si è fermato vicino al muro e ora ha 3 opzioni, tornare indietro o andare verso sopra o sotto. In questo caso, come notiamo nella terza immagine andrà sopra.

Reset

Un altro pulsante interessante per la risoluzione dei puzzle è il reset.

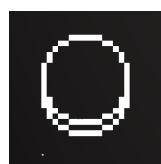


Figura 3.8:
Respawn
button

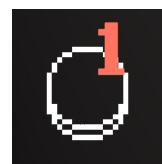


Figura 3.9:
Respawn
available

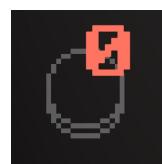


Figura 3.10:
Respawn
not available

La figura 3.8 raffigura il pulsante di respawn nella modalità classica del gioco.

La figura 3.9 raffigura il pulsante di respawn nella modalità hardcore, avrai un solo uso del pulsante a tua disposizione.

La figura 3.10 raffigura il pulsante di respawn usato e non più disponibile nella modalità hardcore.

Questo pulsante ci permetterà di tornare al punto di spawn del livello al patto di sacrificare una parte del tempo rimanente, un'arma a doppio taglio. Bisognerà quindi scegliere con cura la strada da percorrere altrimenti il tempo finirà prima di quanto si crede. Attenzione ad usarlo nella modalità hardcore, la disponibilità è limitata al singolo uso.

Zoom

Un pulsante che ti permetterà di cambiare lo zoom.

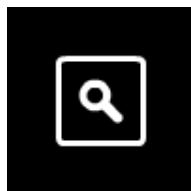


Figura 3.11: Pulsante di zoom

Ci sono due modalità di zoom, in quella di base avrai tutta la mappa a vista e la visuale sarà ferma sulla stanza, invece la secondaria sarà molto più ingrandita sul player e si muoverà seguendo i movimenti che il giocatore effettuerà.

Tempo

Questa barra ci indicherà il tempo rimanente che abbiamo a disposizione prima di perdere.

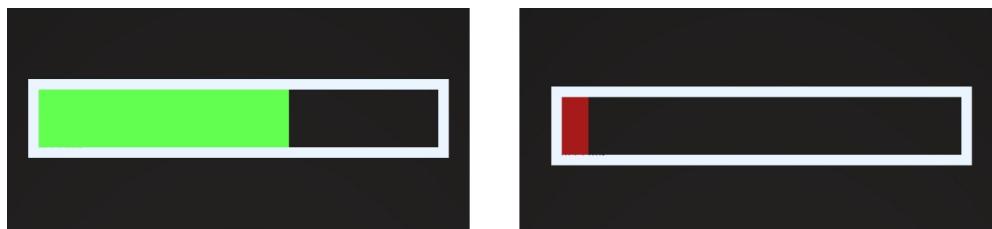


Figura 3.12: Tempo rimanente

Affrettati a risolvere quanti più puzzle possibili. Riuscire a risolvere una stanza ti ridarà una piccola percentuale di tempo. Insomma se sei abbastanza bravo potresti continuare all'infinito.

3.3 Menù di gioco



Figura 3.13: Menù di gioco

Una volta avviato il gioco, verrai accolto da una schermata iniziale semplice ma accattivante. Qui, troverai tre opzioni chiave:

Normal/Hardcore: Questo pulsante centrale è il cuore del menu. Toccare "Normal/Hardcore" ti porterà nell'universo del gioco. Ma c'è di più: noterai due frecce affiancate, una a destra e una a sinistra, che consentono di cambiare la modalità di gioco. Esse ti permettono di scegliere tra le due modalità di gioco: "Normale" e "Hardcore". Una volta visualizzato il nome della modalità desiderata sullo schermo, basta toccare il nome stesso (che funge anche da pulsante) per iniziare la tua avventura.

Settings: Se desideri personalizzare la tua esperienza di gioco, puoi accedere alle opzioni toccando questo pulsante. Qui, potrai regolare le impostazioni audio, controllare i tutorial o effettuare altre modifiche secondo le tue preferenze.

Exit: L'opzione "Exit" ti permette di uscire dal gioco, ma è progettata per evitare chiusure accidentali. Una volta toccato il pulsante "Exit", verrà visualizzata una conferma per assicurarti di voler davvero uscire dal gioco. Questa funzione impedisce chiusure involontarie, assicurandoti di goderti al massimo l'esperienza di Astro Maze.

3.4 Struttura del gioco

Il gioco è molto semplice che sia modalità normale o hardcore: vieni direttamente catapultato in un puzzle all'interno di una piccola stanza. Man mano che risolvi le varie stanze, queste diventano sempre più grandi, rendendo la risoluzione degli enigmi sempre più complessa. Tuttavia, devi fare attenzione al tempo rimanente, che è indicato da una barra nella parte superiore dello schermo. Se il tempo si esaurisce, è game over e dovrai ricominciare da capo. Durante la risoluzione dei puzzle, puoi trovare dei piccoli trofei che ti forniscono una piccola quantità di tempo extra, inoltre puoi collezionarne vari per aumentare il tuo punteggio. Tuttavia, la vera ricompensa è data dalla risoluzione dei puzzle stessi, che ti garantirà una quantità di tempo molto maggiore. L'obiettivo del gioco è cercare di risolvere il maggior numero possibile di stanze senza mai morire, sfruttando al meglio il tempo a tua disposizione.

Modalità normale: Eventi

Gli eventi sono situazioni inattese casuali che si verificano mentre il giocatore sta esplorando il labirinto di stanze. Questi eventi aggiungono un elemento di imprevedibilità al gioco, rendendo ogni partita unica. Attualmente, sono stati introdotti due eventi: la Rotazione e il Fog of War.



Figura 3.14: Evento rotazione

Evento Rotazione: Questo evento fa ruotare la stanza in cui il giocatore si trova di 90 gradi, in senso orario o antiorario, a intervalli regolari. Questa rotazione può rendere la soluzione del puzzle più difficile, poiché il layout della stanza cambierà continuamente. Il giocatore deve adattarsi rapidamente alla nuova disposizione degli elementi.

Evento Fog of War: In questo evento, la visibilità del giocatore è limitata alle immediate vicinanze del personaggio, rendendo difficile vedere l'intera stanza. Questo aggiunge una sfida tattica, poiché il giocatore deve

fare affidamento sulla sua memoria e sulla capacità di pianificazione per esplorare e risolvere il puzzle.

Probabilità di Evento: La probabilità di incontrare un evento casuale aumenta man mano che il giocatore risolve un numero crescente di stanze senza incontrarne uno. Questo significa che all'inizio del gioco, gli eventi potrebbero essere rari, ma diventano sempre più comuni man mano che il giocatore progredisce. Questo aggiunge un elemento di progressione e sfida graduale al gioco. In questo modo, il sistema di eventi nel gioco Astro Maze offre una varietà di esperienze di gioco e sfide impreviste, mantenendo l'interesse del giocatore durante le sessioni di gioco. La progressione delle stanze risolte senza eventi e l'introduzione di eventi originali aggiungono profondità e complessità al gameplay, invitando il giocatore a migliorare continuamente le proprie abilità e strategie.

Modalità hardcore

La modalità "Hardcore" di Astro Maze ti metterà alla prova in modo ancora più intenso rispetto alla modalità normale. Anche se la meccanica di gioco rimane simile, ci sono alcune differenze chiave che la rendono una sfida epica:

Ripristino del Tempo: A differenza della modalità normale, in modalità "Hardcore" il tempo si resetta ad ogni nuova stanza. Questo significa che non puoi accumulare il tempo extra da una stanza all'altra, rendendo ogni singola stanza una sfida autonoma.

Mosse Limitate: In questa modalità, il numero di mosse a tua disposizione è limitato.



Figura 3.15: Mosse disponibili

Quindi, oltre a gestire il tempo, dovrà anche fare attenzione a come utilizzi le mosse disponibili. Devi cercare di pianificare ogni passo in modo strategico per ottimizzare il tuo percorso verso il traguardo del puzzle.

Risorse: Come nella modalità normale, puoi trovare punti luminosi durante la risoluzione dei puzzle, ma in "Hardcore", il loro valore è ancora più prezioso. Ogni punto luminoso non solo aumenta leggermente il tuo tempo rimanente ma può anche fornirti un numero limitato di mosse aggiuntive.

Obiettivo Ulteriore: L’obiettivo principale in ”Hardcore” è lo stesso della modalità normale: risolvere il maggior numero possibile di stanze senza morire. Ma in questa modalità, la sfida è ancora più intensa poiché devi gestire il tempo e le mosse con attenzione per avere successo.

3.5 Algoritmo utilizzato

Tutto l’algoritmo è diviso in varie funzioni all’interno della classe GridManager che conterrà alcuni parametri base della mappa generata.

Fase di generazione

Inizialmente, si specifica la dimensione della stanza iniziale rispetto all’asse delle X e delle Y nel labirinto. È anche necessario stabilire una dimensione massima per entrambi gli assi. Con il superamento di ogni livello, la dimensione della stanza aumenterà di 1 tile lungo ciascun asse, fino a raggiungere la dimensione massima specificata. Questo approccio consente un aumento progressivo della complessità delle stanze man mano che il giocatore avanza nei livelli, offrendo un’esperienza di gioco sempre più difficile e coinvolgente.

Inoltre, viene calcolato un offset basato sulle dimensioni X e Y per posizionare la telecamera esattamente al centro di ogni stanza generata. Questo assicura che la visuale del giocatore sia centrata e bilanciata all’interno della stanza.

Successivamente, viene eseguito un doppio ciclo per coprire tutte le celle della griglia che ci interessano. Durante questo processo, vengono iniziati tutti i tile. Le posizioni di partenza e arrivo all’interno della stanza vengono riempite con dei muri, assegnandoli alla relativa mappa di tile (tile-map). Per quanto riguarda l’interno della stanza, viene assegnato l’87/100 di probabilità che siano pavimenti e il 12/100 di probabilità che siano muri.

Di seguito è riportato un esempio di una stanza, che utilizzeremo per comprendere il funzionamento dell’algoritmo di controllo.

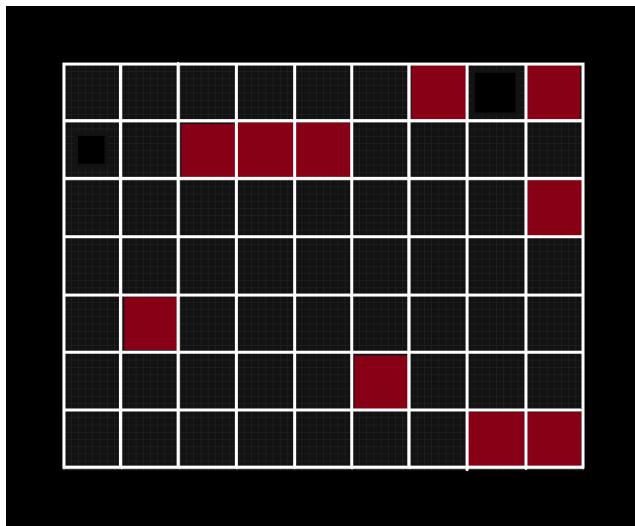


Figura 3.16: Stanza generata

La figura rappresenta la stanza del gioco, evidenziata in bianco sulla griglia. Le celle di colore rosso scuro rappresentano i muri, mentre il resto delle celle rappresenta il pavimento. È interessante notare che esattamente 10 celle su un totale di 63 sono occupate da muri, il che corrisponde approssimativamente al 12/100 come previsto.

Successivamente, viene selezionata una posizione casuale all'interno della stanza per lo spawn del giocatore e dell'uscita. Se queste posizioni sono occupate da un muro, vengono scelte altre posizioni.

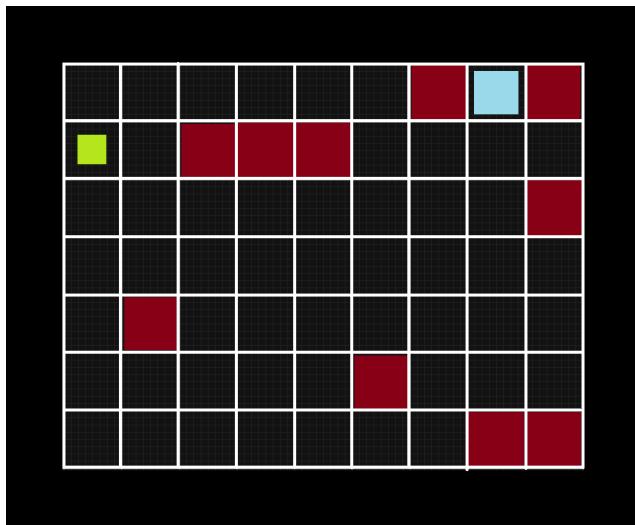


Figura 3.17: Stanza generata con player e uscita

Nella figura precedente(3.17), il giocatore è rappresentato dal quadrato verde, mentre l'uscita è indicata dalla cella di colore azzurro. Attraverso questo approccio, si garantisce che ogni stanza generata abbia una configurazione che permetta al giocatore di raggiungere l'uscita, fornendo una sfida stimolante e mantenendo un equilibrio tra giocabilità e complessità. A questo punto, viene avviata la funzione di valutazione per verificare se la stanza generata è effettivamente risolvibile o meno.

Fase di controllo

Prima di dare uno sguardo all'algoritmo della funzione di valutazione, bisogna spiegare le basi su cosa è una coda e su cosa è un albero.

La coda è una struttura dati costituita come una raccolta di entità tenute in una sequenza che può essere modificata aggiungendo entità a un estremo e rimuovendole dall'altro estremo della sequenza. Per questo motivo è una struttura di tipo FIFO(first in first out). Un esempio è la coda per la cassa al supermercato, ti metti in coda, aspetti che tutti quelli davanti a te finiscano e poi potrai pagare la tua spesa e uscire[10].

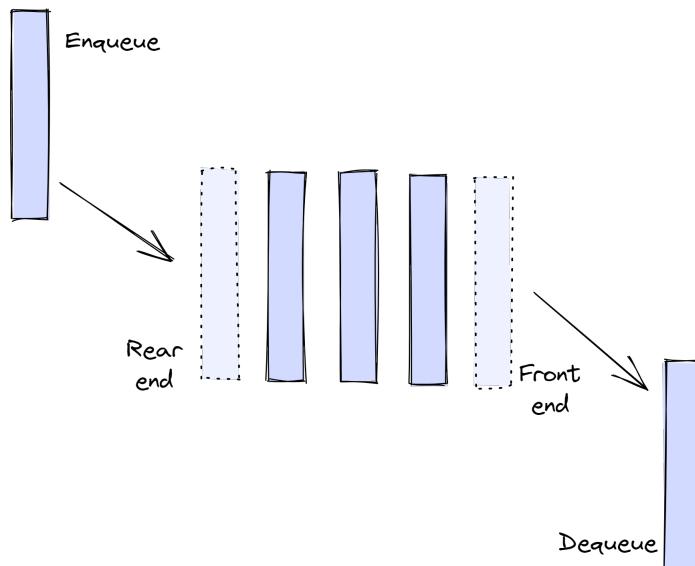


Figura 3.18: Coda

L'albero è la struttura dati che si riconduce al concetto di albero con radice presente nella teoria dei grafi, ovvero grafo non orientato nel quale due vertici qualsiasi sono connessi da uno e un solo cammino (grafo non orienta-

to, connesso e privo di cicli). Un albero si compone di due sottostrutture: il nodo, che contiene le informazioni, e l'arco, che stabilisce un collegamento gerarchico fra due nodi. In genere un nodo si dice nodo padre se da esso esce un arco orientato che lo collega ad un nodo, detto nodo figlio (un nodo può essere sia nodo padre che nodo figlio)[2].

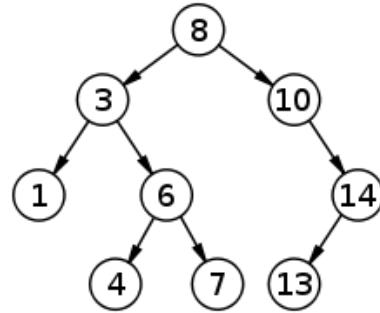


Figura 3.19: Albero

Tenendo a mente il concetto di albero discusso in precedenza andiamo a capire cosa è un algoritmo BFS

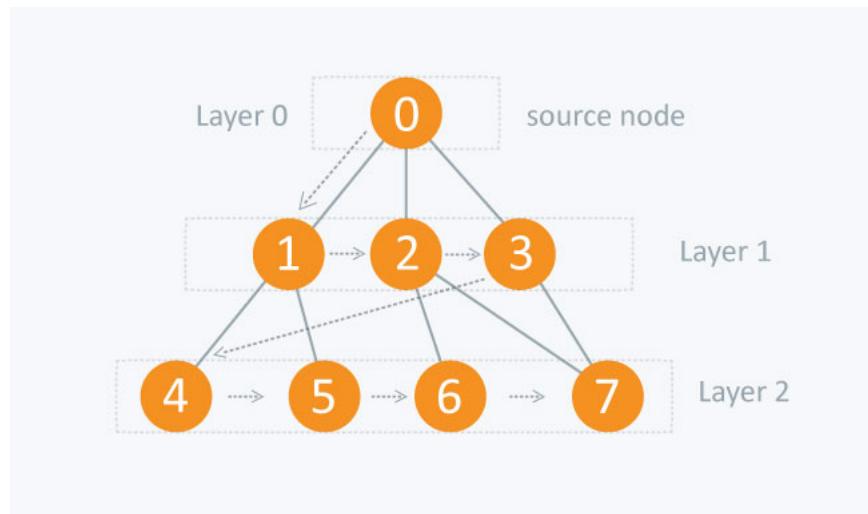


Figura 3.20: Algoritmo BFS

L'algoritmo Breadth First Search, o ricerca in ampiezza, viene usato per la ricerca nei grafi per trovare il cammino più breve ad un nodo scelto in precedenza[23].

Il seguente elenco puntato sintetizza il codice scritto ispirato all'algoritmo BFS usato per controllare che la mappa sia valida o meno.

- Salvataggio dei tile di start e di fine
- Creazione di una coda
- Creazione di una lista dedicata a contenere tutti i tiles già visitati
- Mettere in coda e nella lista il tile dello start(nodo sorgente)
- Inizio ricerca
- Funzione ExploreTilemap(verrà discussa più avanti, in breve toglie dalla coda un nodo, lo esamina e aggiunge nuovi nodi in coda se non sono già stati esaminati.)
- Se il tile di fine puzzle si trova in coda esso viene salvato.
- Se la coda non è vuota ritorna all'inizio ricerca, altrimenti esce e ritorna il numero di passi che ci ha messo per trovare il nodo di fine mappa.

Fase di controllo: ExploreTileMap

Inizialmente, viene rimosso dalla coda un nodo, che rappresenta la posizione di un tile all'interno del labirinto. Successivamente, vengono esaminate tutte e quattro le direzioni possibili: destra, sinistra, sopra e sotto. Per ogni direzione, viene eseguito un ciclo che sposta la posizione del nodo lungo quella direzione finché non viene raggiunto un muro.

Se la posizione precedente al muro non è presente nella lista dei tiles visitati e non è già presente nella coda, il tile viene aggiunto sia alla lista dei tiles visitati che alla coda. Una volta esaminata una direzione, l'algoritmo passa alla successiva.

Di seguito viene mostrato un esempio del primo ciclo dell'algoritmo, utilizzando come riferimento la figura 3.17, che rappresenta una stanza nella fase di generazione del labirinto. Le stelle indicano i nuovi tiles trovati e aggiunti alla coda.

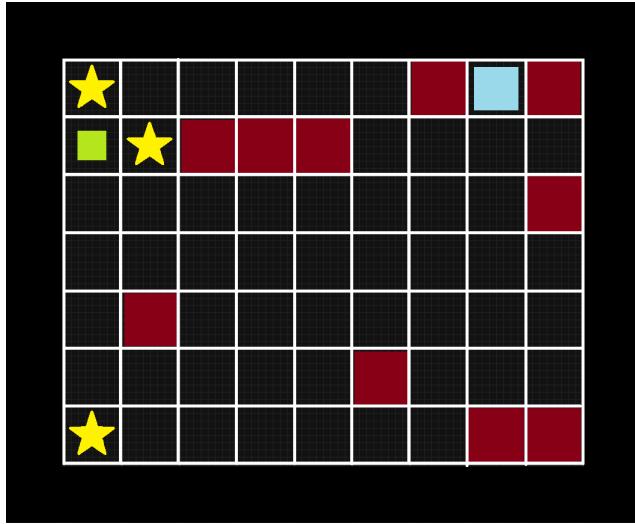


Figura 3.21: Stanza generata con indicati i tile esaminati

L’algoritmo parte dalla posizione del giocatore (start) e trova, seguendo le direzioni, tre nuovi tiles. È importante notare che, anche se lo start viene trovato come posizione durante l’esplorazione, non verrà considerato perché è già stato visitato. L’operazione di esplorazione continua finché non vengono esaurite tutte le possibili strade. Di seguito è mostrato un esempio.

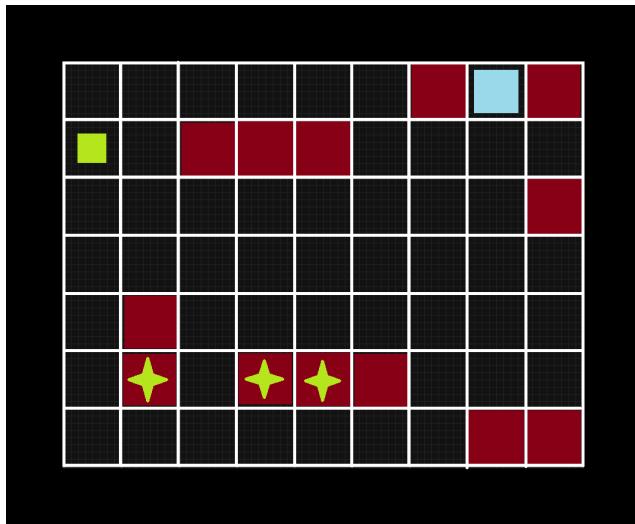


Figura 3.22: Stanza generata e controllata con segnati i muri aggiunti

La generazione procedurale del labirinto per il gioco di rompicapo continua fino a quando la coda non è vuota. In questo modo, tutte le strade percorribili vengono mappate e il resto viene riempito con muri. È interessante osservare che i muri (contrassegnati con la stella verde nella figura

3.22) vengono posizionati in quei tiles in cui il giocatore non sarebbe in grado di passare. Ciò permette di avere una mappa pulita e chiara per il giocatore, concentrandosi esclusivamente sul puzzle di fronte a lui.

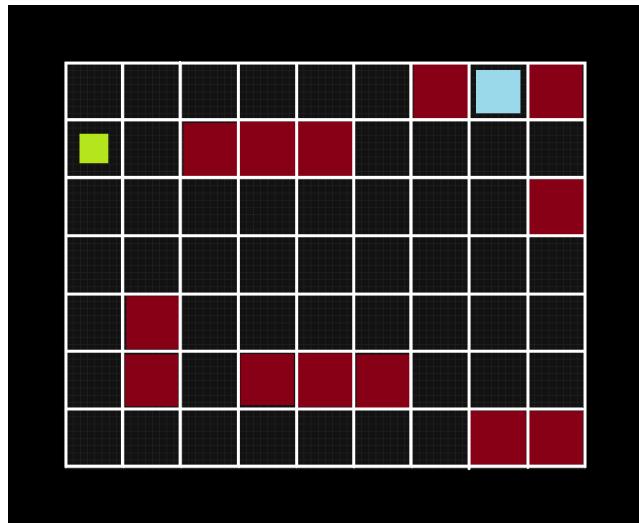


Figura 3.23: Stanza finale

Questo approccio di generazione del labirinto garantisce una struttura ben definita e un design che supporta l'obiettivo principale del gioco: la risoluzione di puzzle.

Capitolo 4

Sviluppo del Lavoro

4.1 Tools utilizzati

Aseprite

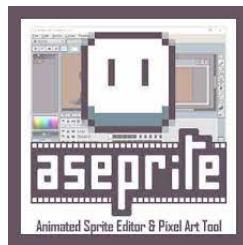


Figura 4.1: Aseprite

Aseprite è uno strumento software specializzato per la creazione e l'animazione di pixel art. È ampiamente utilizzato dagli artisti digitali e dagli sviluppatori di giochi per creare grafiche pixelate di alta qualità, che sono spesso utilizzate in giochi retrò, giochi indie e altre applicazioni digitali[5].

Unity

Unity è una popolare piattaforma di sviluppo di videogiochi utilizzata da sviluppatori di tutto il mondo. È un potente motore di gioco che offre una vasta gamma di strumenti e funzionalità per creare giochi coinvolgenti e interattivi su diverse piattaforme, come PC, console, dispositivi mobili e molto altro ancora. Il programma fornisce una vasta gamma di strumenti e risorse per implementare algoritmi di generazione procedurale nel gioco.

Gli sviluppatori possono utilizzare il linguaggio di scripting C sharp per creare algoritmi personalizzati e integrarli nell'ambiente di sviluppo di



Figura 4.2: Unity

Unity. Inoltre, offre una vasta libreria di asset grafici e audio, effetti visivi, animazioni e molto altro ancora per arricchire l'esperienza di gioco complessiva[34].

SFXR

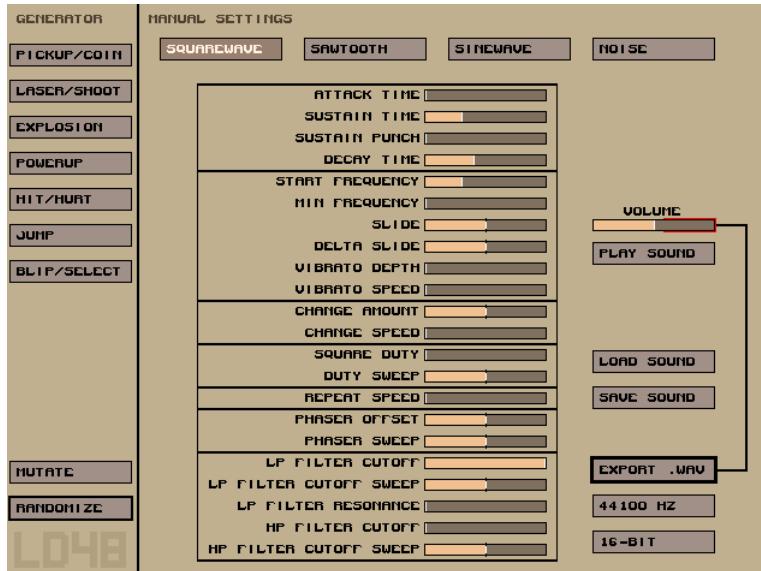


Figura 4.3: SFXR

SFXR è un'applicazione desktop creata da Tomas Pettersson che consente agli sviluppatori di generare facilmente suoni chiptune per videogiochi, animazioni e altre applicazioni interattive.

L'applicazione offre una semplice interfaccia utente che consente di regolare vari parametri sonori per creare effetti sonori unici. I parametri includono forme d'onda, frequenza, volume, attacco, decadimento e molte altre opzioni. Modificando questi parametri, gli utenti possono creare una vasta gamma di suoni, tra cui colpi, esplosioni, rumori di sfondo e altro ancora.

SFXR fornisce anche una funzione di anteprima in tempo reale, che consente agli utenti di ascoltare i suoni che stanno creando man mano che modificano i parametri. Una volta soddisfatti del suono generato, gli utenti possono esportare il suono in diversi formati, come file WAV, il formato di tutti i suoni creati e usati in Astro Maze[25].

Visual Studio Code

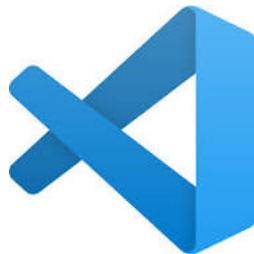


Figura 4.4: Visual Studio Code

Visual Studio Code (VS Code) è un editor di codice sorgente sviluppato da Microsoft. Si tratta di un'applicazione gratuita e open-source, disponibile per Windows, macOS e Linux, che offre una vasta gamma di funzionalità per gli sviluppatori di software.

VS Code offre un potente ambiente di editing con evidenziazione della sintassi, completamento intelligente, rientro automatico, formattazione del codice e altro ancora. Supporta un'ampia varietà di linguaggi di programmazione e offre estensioni per aggiungere funzionalità personalizzate.

L'editor inoltre offre strumenti di debug integrati per diverse piattaforme e linguaggi, fra cui Unity e C sharp, ovvero la piattaforma e il linguaggio usati nel progetto[35].

4.1.1 Componenti utilizzate in Unity

GameObjects

I GameObjects costituiscono gli elementi chiave nella composizione delle scene di Unity e servono a rappresentare una vasta gamma di elementi, tra cui personaggi, oggetti all'interno della scena, ambientazioni, telecamere, punti di riferimento e altro ancora.[15].

Grid

La componente "Grid" in Unity rappresenta un sistema di griglia che può essere utilizzato per organizzare e allineare gli oggetti del tuo gioco, come i "Tiles". Questa griglia ha la caratteristica di avere dimensioni potenzialmente infinite, il che significa che puoi utilizzarla per posizionare oggetti in un vasto spazio di gioco senza limitazioni. La principale funzione della "Grid" è quella di garantire che gli oggetti vengano disposti in modo regolare e preciso sulla mappa di gioco.[16].

Tiles

I "Tiles" in Unity sono elementi grafici 2D che vengono utilizzati per popolare una "Tilemap". Ciascun "Tile" rappresenta un'immagine o una texture e può essere posizionato su una "Tilemap" per creare il layout visivo di un livello 2D. Quelli utilizzati sono tre, il tile 'muro', 'terreno' e 'uscita'(Figura 3.7). Sono stati creati attraverso il programma "Aseprite" nominato precedentemente[30].

Tilemaps

Le tilemaps in Unity sono un modo efficace per creare e gestire livelli 2D all'interno del tuo gioco. Possono essere utilizzate per definire la disposizione e l'aspetto di vari elementi di gioco, come pavimenti, muri o uscite

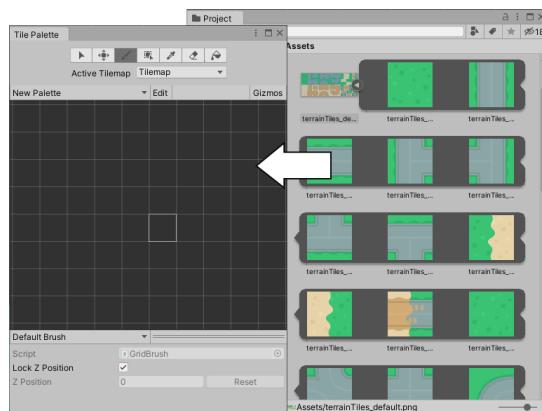


Figura 4.5: Tiles e tilemap

Tilemap del Muro: Questa tilemap contiene tiles che rappresentano i muri nel tuo livello. Inoltre ha anche una componente "Tilemap Collider 2D" associata. Questo significa che i tiles che rappresentano i muri avranno un collider 2D associato a loro. Questo collider 2D impedirà al giocatore di attraversare i muri, quindi il giocatore sarà bloccato quando cercherà di muoversi attraverso di essi.

Tilemap del Pavimento: Questa contiene tiles che rappresentano il pavimento della stanza. Può essere utilizzata per definire l'area accessibile del livello.

Tilemap dell'Uscita: Questa tilemap contiene tiles che rappresentano l'uscita del tuo livello. Questa tilemap ha una componente "Tilemap Collider 2D" associata, il che significa che i tiles che rappresentano l'uscita hanno dei colliders. Questi colliders saranno in grado di rilevare le collisioni con il giocatore quando questo cerca di raggiungere l'uscita. Quando il giocatore collide con l'uscita, verrà avviata la generazione del livello successivo.[31].

Rigidbody 2D

Nella progettazione di videogiochi in 2D, un componente chiave per gestire la fisica dei GameObject è il Rigidbody 2D. Esso permette di controllare il movimento e la rotazione di un oggetto utilizzando il sistema di fisica di Unity. Mentre il Rigidbody 2D condivide alcune proprietà con il suo equivalente in 3D, il Rigidbody standard, è stato appositamente progettato per adattarsi allo sviluppo in due dimensioni.

Il funzionamento di un Rigidbody 2D può essere compreso considerando il componente Transform dell'Editor di Unity. Quest'ultimo definisce la posizione, la rotazione e la scala di un GameObject e dei suoi oggetti figlio all'interno della Scena. Modificando il componente Transform, si possono influenzare anche altri componenti, come i collider che gestiscono le collisioni tra oggetti. Tuttavia, il sistema di fisica 2D di Unity richiede una modalità per comunicare questi spostamenti dei collider al componente Transform. Ed è proprio qui che entra in gioco il componente Rigidbody 2D, che sovrascrive il Transform e ne aggiorna la posizione e/o la rotazione in base alle forze fisiche applicate.

Nel contesto delle interazioni con i collider, un aspetto importante da considerare è la connessione tra Collider 2D e Rigidbody 2D. Quando si aggiunge un componente Collider 2D allo stesso GameObject o al suo figlio, esso è automaticamente collegato al Rigidbody 2D presente sullo stesso GameObject. Questo collegamento fa sì che il Collider 2D si muova insieme al Rigidbody 2D, evitando problemi di collisioni erronee. Pertanto, è fondamentale spostare il Rigidbody 2D invece di muovere direttamente il Collider 2D tramite il componente Transform o gli offset del collider. Questo approccio garantisce il miglior rendimento e una corretta rilevazione delle collisioni. Inoltre, quando più Collider 2D sono collegati allo stesso Rigidbody 2D, essi non entreranno mai in collisione tra di loro, permettendo di creare insiemi di collider che agiscono come un unico collider composto, mantenendo il movimento e la rotazione sincronizzati con il Rigidbody 2D[22].

Collider 2D

In Unity, un Collider 2D è un componente utilizzato nel sistema di fisica 2D per definire l'area di rilevamento delle collisioni di un oggetto in un ambiente bidimensionale. Un collider è essenzialmente una forma geometrica (come un rettangolo, un cerchio, una capsula, ecc.) che circonda l'oggetto e viene utilizzata per interagire con altri collider nel mondo del gioco.

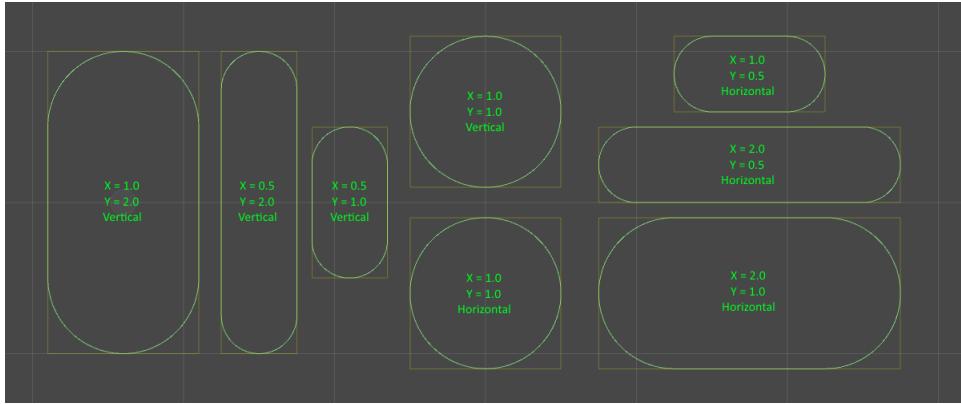


Figura 4.6: Esempio di collider2D[11]

Quando si vuole consentire agli oggetti di reagire alle collisioni o di interagire tra loro in un gioco 2D, è necessario aggiungere un componente Collider 2D all'oggetto. Questo componente definisce la "scatola invisibile" attorno all'oggetto e, quando un altro collider si sovrappone o entra in contatto con esso, vengono attivate le funzioni di gestione delle collisioni. Scegliere il tipo di collider adeguato dipende dalle esigenze specifiche del gioco e dalla forma dell'oggetto che deve essere presa in considerazione per le collisioni.

Nel contesto di Astro Maze, è fondamentale evidenziare l'importanza dei due principali collider utilizzati. Innanzitutto, è presente il Tilemap Collider 2D, il quale consente di definire un'intera area di collisione per i tile disegnati all'interno della tilemap. Questo collider permette di creare delle zone di collisione definite dai tile, che determinano come il giocatore interagisce con l'ambiente circostante[32].

Inoltre, il secondo collider di rilievo è il Box Collider 2D del personaggio giocante. Questo collider controlla se il personaggio si scontra con i muri o raggiunge un'uscita. Attraverso questo componente, è possibile gestire in modo preciso le interazioni del personaggio con gli elementi del labi-

rinto, determinando quando il personaggio entra in contatto con i confini dell'ambiente di gioco o con le uscite presenti[7].

4.2 Classificazione dell'algoritmo

Per una maggiore chiarezza sugli algoritmi utilizzati, è utile confrontare quanto discusso nella teoria generale della generazione procedurale di contenuti con l'implementazione effettiva nel videogioco. L'algoritmo di generazione procedurale utilizzato è di tipo online, il che significa che la mappa del puzzle deve essere generata in modo diverso ad ogni avvio del gioco e ad ogni nuovo livello. Inoltre, la configurazione della stanza cambierà man mano che il giocatore avanza verso punteggi sempre più alti, rendendo i puzzle successivi sempre più complicati.

I contenuti generati includono sia elementi essenziali che opzionali. Gli elementi essenziali includono la stessa stanza del puzzle, poiché è fondamentale che il puzzle sia sempre risolvibile. Al contrario, i punti per guadagnare tempo extra sono considerati contenuti opzionali, e la loro presenza all'interno della stanza non è strettamente necessaria.

Attualmente, viene generato un seed per ogni puzzle, anche se al momento non viene utilizzato. In futuro, potrebbe essere possibile utilizzare questa meccanica per creare puzzle personalizzati o salvare livelli che non sono stati risolti a causa del poco tempo rimasto.

La casualità è di grande importanza, poiché un giocatore non sarebbe interessato a giocare a un gioco in cui deve risolvere gli stessi puzzle in modo identico ogni volta. Pertanto, è fondamentale che l'algoritmo di generazione sia stocastico e non deterministico.

L'algoritmo utilizzato è di tipo "generate-and-test", ma l'obiettivo è trasformarlo in un algoritmo "constructive". Attualmente, la mappa viene generata e successivamente vengono esplorate tutte le possibili strade per verificare se esiste un percorso verso l'uscita. Se non viene trovato alcun percorso, la mappa viene scartata e rigenerata da capo. Un'ottimizzazione futura potrebbe prevedere di riposizionare l'uscita in un punto accessibile anziché scartare completamente il puzzle, al fine di evitare caricamenti lunghi su dispositivi più vecchi.

4.3 Rappresentazione dei contenuti

Il genotipo è rappresentato unicamente dalle dimensioni X e Y della stanza da generare. Il fenotipo, invece, è rappresentato da una lista di coordinate che definiscono la posizione degli elementi di gioco, come l'inizio e la fine del livello e i vari tiles nel livello stesso. La relazione tra genotipo e fenotipo è indiretta: il calcolo non è particolarmente complesso, ma la dimensione aumenta esponenzialmente in base alle dimensioni della stanza. L'aggiunta di ulteriori parametri consentirebbe la creazione di livelli ancora più interessanti, mantenendo comunque un tempo di esecuzione accettabile per l'algoritmo.

4.4 Funzione di valutazione

La funzione di fitness prende il puzzle creato e cerca di trovare un percorso dalla posizione di inizio fino all'uscita. Per una spiegazione più dettagliata, si può fare riferimento al capitolo precedente sulla fase di controllo. In base al risultato ottenuto (trovare o non trovare l'uscita), il programma rigenera il puzzle o accetta il prototipo.

Capitolo 5

Valutazione del Lavoro

5.1 Game Experience Questionnaire

Il Game Experience Questionnaire, noto anche come GEQ, è uno strumento di valutazione ampiamente utilizzato nel campo dei videogiochi per misurare l'esperienza di gioco degli utenti. È stato sviluppato per fornire una valutazione completa delle dimensioni soggettive dell'esperienza di gioco, come l'immersione, l'eccitazione, la competenza e la tensione.

Il GEQ viene utilizzato per ottenere feedback accurati e quantitativi sugli aspetti emotivi e cognitivi di un'esperienza di gioco. Questo strumento è ampiamente applicato sia in studi accademici che nell'industria dei videogiochi per valutare l'impatto di diversi elementi di design, meccaniche di gioco e contesti di interazione sugli utenti. Rispetto ad altri strumenti di valutazione dell'esperienza di gioco, il GEQ si distingue per la sua natura multidimensionale. Esso permette di catturare diversi aspetti dell'esperienza di gioco, consentendo una valutazione approfondita e dettagliata. Le sue dimensioni coprono una vasta gamma di elementi, inclusi gli aspetti affettivi, cognitivi e motivazionali dell'esperienza di gioco.

Inoltre, il GEQ è stato ampiamente validato attraverso studi empirici, dimostrando una buona affidabilità e validità come strumento di valutazione dell'esperienza di gioco. La sua struttura chiara e le domande ben definite rendono il GEQ un metodo affidabile e robusto per raccogliere dati sulla percezione dei giocatori.

Scegliere il GEQ per valutare il tuo gioco permette di ottenere un'analisi dettagliata delle esperienze dei giocatori, consentendo di identificare punti di forza e di miglioramento. La sua ampia applicabilità e validità scientifica ne fanno uno strumento di valutazione affidabile e riconosciuto nel campo

dei videogiochi. Utilizzando il GEQ per valutare il tuo gioco, sarai in grado di raccogliere dati preziosi per comprendere meglio l'impatto del tuo gioco sugli utenti e guidare le decisioni di progettazione future.[18]

5.1.1 Struttura del GEQ

Il Questionario sull'Esperienza di Gioco (Game Experience Questionnaire, GEQ) ha una struttura modulare ed è composto da tre parti: il questionario centrale, il modulo di presenza sociale (non presente nel questionario utilizzato per la valutazione di Astro Maze) e il modulo post-gioco. Inoltre, è stata sviluppata un modulo per la valutazione dell'esperienza durante il gioco. Tutti e tre i moduli sono progettati per essere somministrati immediatamente dopo la sessione di gioco, nell'ordine indicato sopra. La parte 1 e la parte 2 indagano i sentimenti e i pensieri dei giocatori durante il gioco, mentre la parte 3, il modulo post-gioco, valuta come i giocatori si sono sentiti dopo aver smesso di giocare.

La parte 1 costituisce la parte centrale del GEQ e valuta l'esperienza di gioco attraverso sette componenti: Immersione, Flow, Competenza, Affetto Positivo e Negativo, Tensione e Sfida. Per una misurazione accurata, sono necessari cinque elementi per ciascuna componente. Poiché la traduzione degli elementi del questionario, nonostante la cura apportata, a volte può portare a modelli di punteggio non ottimali, è stato aggiunto un elemento di riserva per tutte le componenti.

La parte 2, il modulo di presenza sociale, indaga il coinvolgimento psicologico e comportamentale del giocatore con altre entità sociali, che siano esse virtuali (ad esempio, personaggi all'interno del gioco), mediate (ad esempio, altri giocatori online) o co-localizzate. Questo modulo dovrebbe essere somministrato solo quando almeno uno di questi tipi di co-giocatori è coinvolto nel gioco, infatti esso è stato rimosso nel questionario per Astro Maze.

La parte 3, il modulo post-gioco, valuta come i giocatori si sono sentiti dopo aver smesso di giocare. Questo modulo è rilevante per valutare il gioco naturalistico (cioè quando i giocatori hanno deciso volontariamente di giocare), ma può essere pertinente anche nella ricerca sperimentale.

La versione In-game del GEQ è una versione concisa del questionario centrale. Ha la stessa struttura delle componenti e consiste in elementi selezionati da questo modulo. Il questionario In-game è stato sviluppato per valutare l'esperienza di gioco in diversi intervalli durante una sessione

di gioco o una sessione di riproduzione. In quanto il gioco è relativamente semplice e non cambia tra una sessione e l'altra è stato deciso di non inserire neanche questo modulo.

Queste linee guida di punteggio aiutano a calcolare i punteggi delle diverse componenti del GEQ basandosi sulle risposte degli utenti al questionario. I punteggi delle componenti vengono calcolati come la media dei valori degli elementi corrispondenti. Utilizzando questi punteggi, si potrà analizzare e interpretare l'esperienza di gioco degli utenti in relazione a ciascuna dimensione misurata dal GEQ.

Punteggi del modulo centrale GEQ:

Il modulo centrale GEQ è composto da sei componenti, e i relativi elementi sono elencati di seguito. I punteggi delle componenti vengono calcolati come la media dei valori dei loro elementi(Domande)[17].

- Competenza: Domanda 7, 9 e 15.
- Immersione sensoriale e immaginativa: Domanda 16.
- Flow: Domanda 4.
- Tensione/fastidio: Domanda 10, 13 e 14.
- Sfida: Domanda 7, 11 e 15.
- Affetto negativo: Domanda 2, 6.
- Affetto positivo: Domanda 1, 3, 5 e 8.

Punteggi del modulo post-gioco GEQ:

Il modulo post-gioco GEQ è composto da quattro componenti, e i relativi elementi sono elencati di seguito. I punteggi delle componenti vengono calcolati come la media dei valori dei loro elementi(Domande).

- Esperienza positiva: Domanda 1, 3, 4, 6.
- Esperienza negativa: Domanda 2 e 7.
- Stanchezza: Domanda 5.
- Ritorno alla realtà: Domanda 8.

5.2 Questionario

Il questionario è stato distribuito a una vasta gamma di giocatori, dando priorità alla diversità anziché alla quantità. Per la sua creazione e gestione online, è stato utilizzato il sito "Polfish", che consente di creare questionari personalizzati, mantenerli online e visualizzare i risultati in tempo reale.[21].

I giocatori dovevano indicare come si sono sentiti durante il gioco per ciascuno degli elementi, utilizzando la seguente scala (corrisponde alla quantità di stelle selezionate durante le rispettive domande):

per niente - un po' - moderatamente - abbastanza - estremamente
1 - 2 - 3 - 4 - 5

Modulo centrale

1. Mi sono sentito/a soddisfatto/a
2. Mi sono sentito/a confuso/a
3. Ho trovato che fosse divertente
4. Ero completamente concentrato/a sul gioco
5. Ho trovato soddisfacenti le dinamiche di gioco proposte
6. L'ho trovato noioso
7. Ho pensato che fosse difficile
8. Ho provato una grande soddisfazione nel completare i puzzle
9. Ero veloce nel raggiungere gli obiettivi del gioco
10. Mi sono sentito/a sotto pressione
11. Mi sono sentito/a sfidato/a
12. Mi sono sentito/a coinvolto/a nel processo di risoluzione dei puzzle
13. Mi sono sentito/a frustrato/a quando non riuscivo a trovare una soluzione
14. Ho sentito la pressione del tempo
15. Ho dovuto fare molto sforzo per completare i puzzle
16. Quanto ti sei sentito/a coinvolto/a nel gioco?

Modulo post-gioco

1. Mi sono divertito
2. Mi sono sentito/a frustrato/a per la lentezza nel completare i puzzle
3. Ho trovato stimolante la sfida dei puzzle proposti
4. Mi sono sentito/a motivato/a a migliorare le mie abilità di risoluzione dei puzzle
5. Mi sono sentito/a esausto/a
6. Ho provato una sensazione di gratificazione quando ho superato i puzzle difficili
7. Mi sono sentito/a confuso/a
8. Quanto hai sentito il desiderio di continuare a giocare anche dopo aver smesso?

Le domande sono state selezionate attentamente in base a diversi criteri. Innanzitutto, è stato considerato il contesto del documento GEQ e del gioco Astro Maze, cercando di includere domande pertinenti e rilevanti. Questo può includere domande sulla meccanica di gioco, sulle regole o sugli obiettivi del gioco.

Inoltre, le domande sono state progettate e selezionate per coprire un'ampia gamma di argomenti pertinenti. Ciò può includere domande sulle sfide o sulle strategie di gioco e per incoraggiare una riflessione critica o una valutazione. Ciò può aiutare gli sviluppatori a migliorare nei punti carenti e soddisfare le aspettative dei giocatori.

5.3 Risultati

Per analizzare i risultati, è stato utilizzato un foglio di calcolo Excel. I dati relativi alle risposte di ciascuna domanda provenienti dai due questionari sono stati inseriti nel foglio di calcolo. Qui sotto sono stati riportati i risultati attraverso un'istogramma per ogni domanda sia del modulo centrale che del modulo post-gioco. (Le domande non sono ordinate)

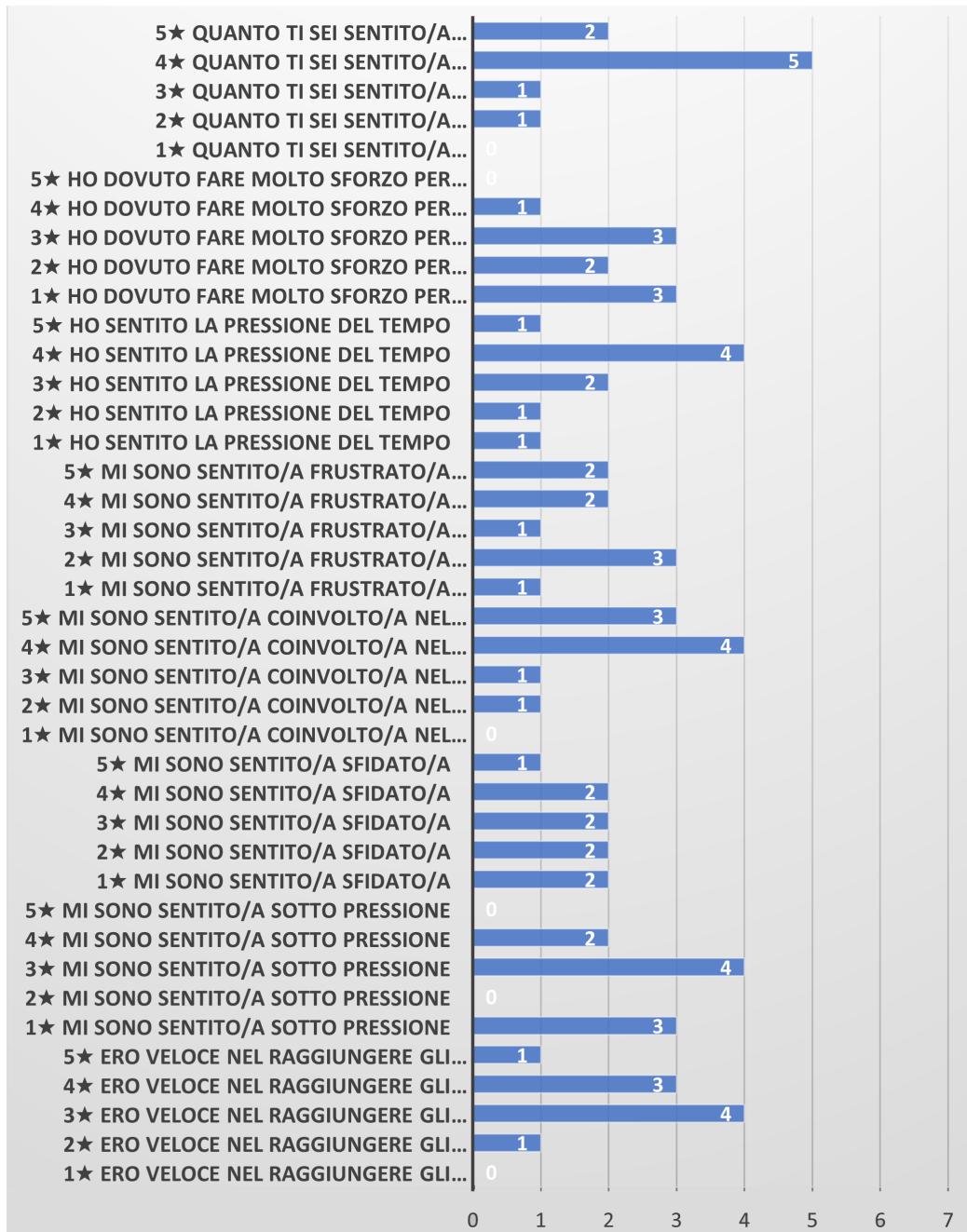


Figura 5.1: Risultati del modulo centrale parte 1

I partecipanti al questionario centrale hanno manifestato un alto grado di soddisfazione (domanda 1) e hanno trovato il gioco divertente (domanda 3). Inoltre, il gioco è riuscito a mantenere l'attenzione dei giocatori, poiché hanno segnalato di essere completamente concentrati durante il gioco (domanda 4). La dinamica di gioco proposta è stata valutata positivamente (domanda 5), e i giocatori hanno sperimentato una grande soddisfazione nel

completare i puzzle (domanda 8). Tuttavia, alcuni giocatori hanno segnalato che il gioco potrebbe essere migliorato per quanto riguarda la sfida e la competenza (domande 7 e 11). Inoltre, alcuni giocatori hanno riferito di sentirsi sotto pressione (domanda 10) e hanno notato la pressione del tempo (domanda 14).

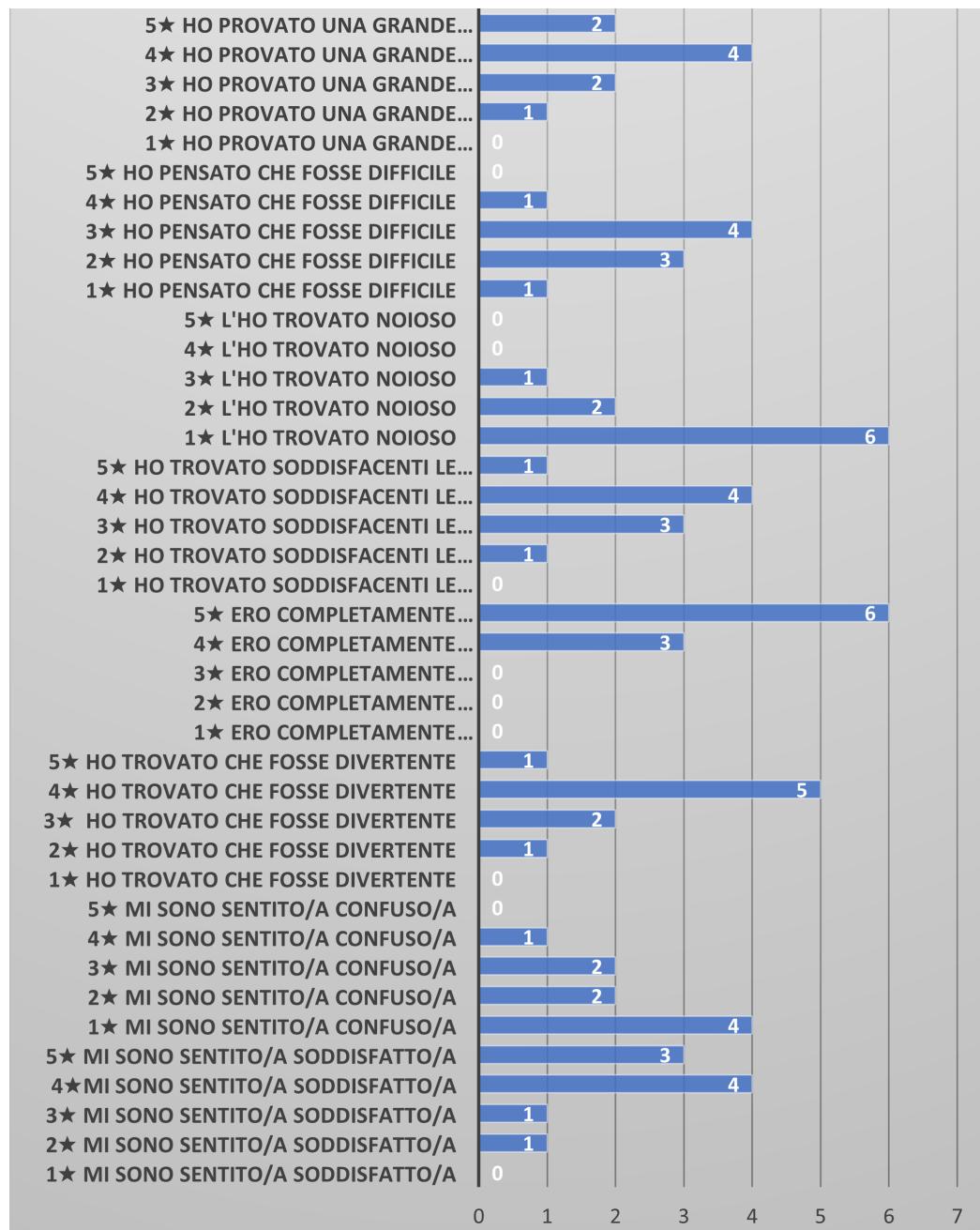


Figura 5.2: Risultati del modulo centrale parte 2

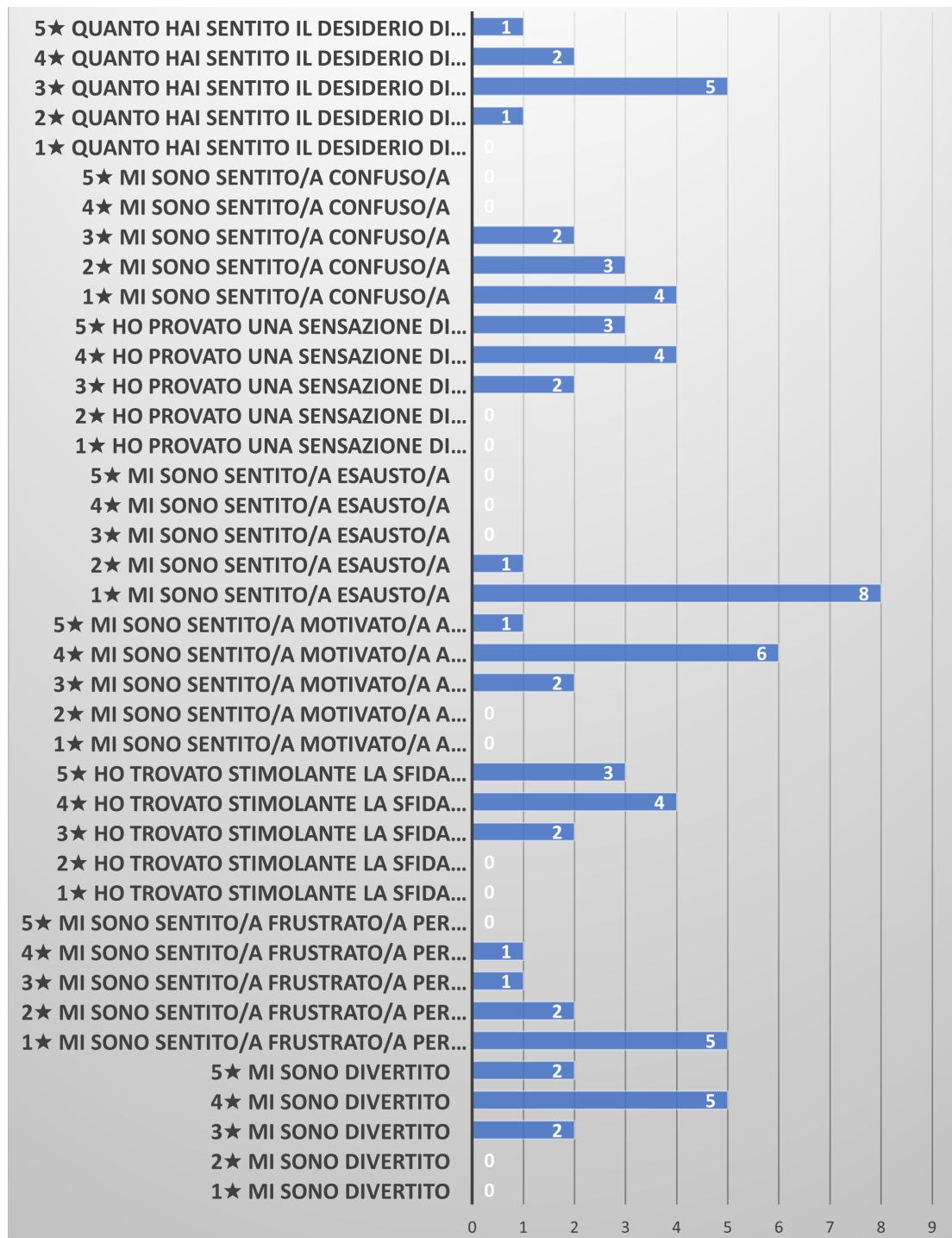


Figura 5.3: Risultati del modulo post-gioco

La maggior parte dei partecipanti al questionario post-gioco ha riportato di essersi divertito durante il gioco (domanda 1) e di aver trovato stimolante

la sfida dei puzzle proposti (domanda 3). Inoltre, i giocatori hanno manifestato una motivazione intrinseca a migliorare le proprie abilità di risoluzione dei puzzle (domanda 4) e hanno sperimentato una sensazione di gratificazione nel superare i puzzle difficili (domanda 6). Tuttavia, nonostante sono pochi, alcuni giocatori hanno segnalato di aver provato una leggera frustrazione per la lentezza nel completare i puzzle (domanda 2). Infine, il desiderio di continuare a giocare anche dopo aver smesso è risultato nella media (domanda 8). Questi ultimi aspetti potrebbero essere ulteriormente esaminati e affinati per garantire un'esperienza di gioco ancora più appagante.

Successivamente, è stato calcolato il valore medio per ogni singola domanda, tenendo conto delle risposte fornite dai partecipanti. Dopo aver ottenuto le medie per ciascuna domanda, sono state raggruppate le domande appartenenti alla stessa categoria. Per esempio, se c'erano domande riguardanti la competenza del giocatore, sono state messe insieme e calcolata la media complessiva che fornisce un'indicazione complessiva della categoria in questione.

Esempio del calcolo

Per spiegare come è stata calcolata la media per valutare il livello di tensione/fastidio riguardante il modulo centrale, analizzeremo i passaggi nel dettaglio.

Prima di tutto, abbiamo esaminato le risposte relative alle domande appartenenti alla categoria in questione, Q10, Q13 e Q14.



Figura 5.4: Risposte della domanda 10 del modulo centrale



Figura 5.5: Risposte della domanda 13 del modulo centrale

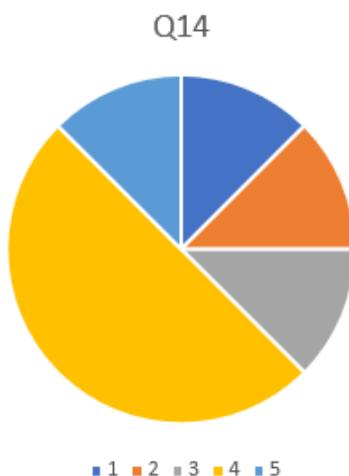


Figura 5.6: Risposte della domanda 14 del modulo centrale

Ora, procederemo con il calcolo della media per ciascuna domanda utilizzando una formula specifica:

$$((1 * x) + (2 * y) + (3 * k) + (4 * w) + (5 * z))/n$$

Nella formula, x rappresenta il numero di persone che hanno risposto con il valore 1 alla domanda corrispondente, y indica quelle che hanno risposto con 2, e così via fino a z, che rappresenta quelle con risposta 5. Inoltre, n rappresenta il numero totale di persone che hanno compilato il questionario. Questo calcolo fornisce la media ponderata della domanda, in cui il peso (1,

2, 3, 4 e 5) è determinato dalla natura specifica della domanda e dalla categoria di appartenenza.

Consideriamo un esempio in cui 8 persone hanno risposto al questionario. Applichiamo la formula alle domande Q10, Q13 e Q14:

$$Q10 = (1 * 3) + (2 * 0) + (3 * 3) + (4 * 2) + (5 * 0) / 8 = 2.5$$

$$Q13 = (1 * 1) + (2 * 3) + (3 * 1) + (4 * 1) + (5 * 2) / 8 = 3$$

$$Q14 = (1 * 1) + (2 * 1) + (3 * 1) + (4 * 4) + (5 * 1) / 8 = 3.375$$

Successivamente, per calcolare il valore complessivo della categoria, otteniamo la media dei valori delle domande calcolate precedentemente.

$$Tensione = (2.5 + 3 + 3.375) / 3 = 2.96 / 5$$

Il valore ottenuto, leggermente superiore alla metà, indica che durante il gioco c'è una tensione media presente. Tuttavia, è importante notare che tale tensione non è necessariamente negativa. In effetti, Astro Maze mira a mettere sotto pressione il giocatore durante la risoluzione dei vari enigmi, soddisfacendo le aspettative del gioco.

Modulo centrale

Di seguito riportiamo i risultati ottenuti dal calcolo dei valori medi di ciascuna categoria nel primo questionario:

- Competenza: 3.26/5

Il punteggio indica un livello moderato di competenza per i partecipanti. Suggerisce che i giocatori si sentono abbastanza competenti nel gestire le sfide proposte da Astro Maze, ma potrebbe essere possibile stimolare ulteriormente la loro competenza attraverso l'introduzione di nuovi elementi di gioco o livelli di difficoltà crescente.

- Immersione sensoriale e immaginativa: 3.89/5

Un punteggio positivo per l'immersione sensoriale e immaginativa indica che il gioco è in grado di coinvolgere i giocatori e stimolare la loro immaginazione. Ciò potrebbe essere attribuito a un design di livello che crea un'esperienza immersiva per i partecipanti.

- Flow: 4.67/5

Il punteggio di Flow suggerisce che Astro Maze riesce a offrire un'esperienza coinvolgente e gratificante. La sensazione di flow si manifesta quando le abilità dei giocatori si allineano con le sfide proposte dal gioco, creando un senso di coinvolgimento e soddisfazione.

- Tensione/fastidio: 3.00/5

Il punteggio indica che il gioco riesce a generare una certa dose di tensione e fastidio durante il gameplay, come inteso nell'obiettivo del gioco. Tuttavia, il punteggio suggerisce che la tensione potrebbe essere percepita solo a un livello moderato dai partecipanti, senza superare i limiti di frustrazione.

- Sfida: 3.04/5

Il punteggio di sfida indica che i partecipanti percepiscono il livello di sfida nel gioco come moderato. Ciò potrebbe significare che Astro Maze offre un bilanciamento adeguato tra la difficoltà dei puzzle proposti e le abilità dei giocatori. Tuttavia, potrebbe essere interessante esplorare l'opportunità di introdurre sfide più complesse o nuove meccaniche di gioco per mantenere l'interesse dei giocatori esperti.

- Affetto negativo: 1.72/5

Il basso punteggio di affetto negativo indica che il gioco genera sentimenti negativi minimi nei partecipanti. Questo è un risultato positivo, indicando che Astro Maze riesce a mantenere un equilibrio tra la tensione e il divertimento senza generare affetto negativo e frustrazione eccessiva.

- Affetto positivo: 3.75/5

Il punteggio suggerisce che i partecipanti percepiscono una buona dose di affetto positivo durante il gioco. Ciò indica che Astro Maze crea un'esperienza divertente e soddisfacente, che porta a una valutazione complessivamente positiva dell'esperienza di gioco.

Modulo post-gioco

Di seguito riportiamo i risultati ottenuti dal calcolo dei valori medi di ciascuna categoria nel secondo questionario:

- Esperienza positiva: 4.03/5

Il punteggio indica che i partecipanti percepiscono complessivamente un'esperienza positiva durante il gioco. Questo suggerisce che Astro Maze riesce a offrire momenti di divertimento, soddisfazione e coinvolgimento durante la risoluzione dei puzzle.

- Esperienza negativa: 1.67/5

Un basso punteggio di esperienza negativa suggerisce che il gioco genera sentimenti negativi minimi nei partecipanti. Questo è un risultato

positivo, indicando che Astro Maze riesce a mantenere un equilibrio tra la tensione e il divertimento senza generare frustrazione e stress eccessivi.

- Stanchezza: 1.11/5

Il punteggio indica che i partecipanti non percepiscono un alto livello di stanchezza durante il gioco. Ciò suggerisce che Astro Maze non richiede uno sforzo fisico o mentale eccessivo, consentendo ai giocatori di godersi l'esperienza senza affaticarsi in modo significativo.

- Ritorno alla realtà: 3.33/5

Il punteggio riflette la percezione dei partecipanti quando terminano la sessione di gioco e ritornano alla realtà. Un punteggio di 3.25 indica che i giocatori sperimentano una transizione moderata dal mondo virtuale del gioco al mondo reale. Questo può essere un segno positivo che Astro Maze riesce a offrire un'esperienza coinvolgente e che i giocatori si sentono coinvolti durante il gameplay.

Conclusione della valutazione

Complessivamente, i risultati dei questionari indicano che Astro Maze ha avuto successo nel raggiungere il suo obiettivo di mettere sotto tensione il giocatore senza esagerazioni, offrendo un'esperienza coinvolgente e stimolante. Il gioco genera un coinvolgimento emotivo positivo e limita la percezione di stanchezza, consentendo ai giocatori di divertirsi durante la risoluzione dei puzzle. Tuttavia, ci sono alcuni aspetti che potrebbero essere valutati e raffinati, come la sfida e la competenza, al fine di offrire una maggiore stimolazione ai giocatori esperti. Continuare a migliorare e affinare il gioco in base ai feedback dei giocatori può contribuire a garantire un'esperienza di gioco sempre più appagante e coinvolgente.

Capitolo 6

Conclusioni

Abbiamo esplorato il campo della generazione procedurale di contenuti nei videogiochi e abbiamo presentato Astro Maze, un puzzle-arcade game che sfrutta appieno le potenzialità di questa tecnologia. Attraverso lo studio dello stato dell'arte e l'analisi di giochi esistenti, abbiamo compreso l'importanza e l'impatto della PCG nell'industria dei videogiochi.

Abbiamo approfondito gli algoritmi e le tecniche utilizzate per la generazione procedurale di mappe e livelli, riconoscendo la necessità di bilanciare varietà, coerenza ed equilibrio nella creazione di contenuti. Abbiamo anche sperimentato l'applicazione pratica di tali algoritmi nello sviluppo di Astro Maze, creando un'esperienza di gioco coinvolgente e sempre sorprendente per i giocatori.

L'implementazione della PCG in Astro Maze ci ha permesso di offrire un labirinto immenso e unico ad ogni partita, con stanze generatrici di sfide sempre nuove. La combinazione di elementi puzzle, arcade, platform e roguelike ha creato una sinergia interessante e stimolante per i giocatori, che si sono trovati a dover bilanciare la risoluzione degli enigmi con la gestione del tempo rimanente.

6.1 Sviluppi futuri

Tuttavia, nonostante i risultati ottenuti, ci sono ancora molte opportunità per il futuro sviluppo di Astro Maze e per l'applicazione della generazione procedurale di contenuti nei videogiochi in generale. Ad esempio, potremmo esplorare l'introduzione di nuove meccaniche di gioco, personaggi o ostacoli generati proceduralmente, ampliando così ulteriormente la varietà di esperienze offerte ai giocatori.

Inoltre, potremmo esplorare l'implementazione di algoritmi di generazione più avanzati. Ciò potrebbe consentire di creare un'esperienza di gioco ancora più personalizzata e coinvolgente, adattata alle preferenze e alle abilità individuali dei giocatori.

Infine, potremmo considerare l'integrazione di elementi di intelligenza artificiale (AI) nella generazione procedurale di contenuti, consentendo al gioco di apprendere dai comportamenti e dalle scelte dei giocatori per creare esperienze sempre più avvincenti e adattive nel tempo.

In conclusione, Astro Maze rappresenta solo un piccolo passo nell'ampio panorama della generazione procedurale di contenuti nei videogiochi. Tuttavia, il suo sviluppo e i risultati ottenuti dimostrano il grande potenziale di questa tecnologia per offrire esperienze di gioco uniche, rigiocabili e coinvolgenti. Con ulteriori ricerche e sviluppi futuri, la generazione procedurale di contenuti continuerà a trasformare l'industria dei videogiochi e ad arricchire la vita dei giocatori con nuove avventure da scoprire e da affrontare.

Bibliografia

- [1] Tarn Adams. “Simulation principles from dwarf fortress”. In: *Game AI Pro 2* (2015), pp. 519–521.
- [2] *Albero (informatica)*. URL: [https://it.wikipedia.org/wiki/Albero_\(informatica\)](https://it.wikipedia.org/wiki/Albero_(informatica)).
- [3] *Answer set programming*. URL: https://it.wikipedia.org/wiki/Answer_set_programming.
- [4] Jonne Arjoranta. “Game definitions: A Wittgensteinian approach”. In: *Game Studies: the international journal of computer game research* 14.1 (2014).
- [5] *Aseprite*. URL: <https://www.aseprite.org>.
- [6] Alexander Baldwin et al. “Mixed-initiative procedural generation of dungeons using game design patterns”. In: (2017), pp. 25–32.
- [7] *b collider2d*. URL: <https://docs.unity3d.com/ScriptReference/BoxCollider2D.html>.
- [8] Jamis Buck. “Mazes for programmers: code your own twisty little passages”. In: *Mazes for Programmers* (2015), pp. 1–286.
- [9] ROBERTO CAPIOTTO. “Generazione procedurale di contenuti applicata ai livelli di un gioco”. In: (2015).
- [10] *Coda (informatica)*. URL: [https://it.wikipedia.org/wiki/Coda_\(informatica\)](https://it.wikipedia.org/wiki/Coda_(informatica)).
- [11] *collider2d unity*. URL: <https://docs.unity3d.com/560/Documentation/Manual/class-CapsuleCollider2D.html>.
- [12] Sean C Duncan. “Minecraft, beyond construction and survival”. In: (2011).
- [13] *Dwarf Fortress*. URL: https://it.wikipedia.org/wiki/Dwarf_Fortress.
- [14] *Elite*. URL: [https://it.wikipedia.org/wiki/Elite_\(videogioco\)](https://it.wikipedia.org/wiki/Elite_(videogioco)).
- [15] *GameObject component reference*. URL: <https://docs.unity3d.com/ScriptReference/GameObject.html>.

- [16] *Grid*. URL: <https://docs.unity3d.com/ScriptReference/Grid.html>.
- [17] Wijnand A IJsselsteijn, Yvonne AW De Kort e Karolien Poels. “The game experience questionnaire”. In: (2013).
- [18] Daniel Johnson, M John Gardner e Ryan Perry. “Validation of two game experience scales: the player experience of need satisfaction (PENS) and game experience questionnaire (GEQ)”. In: *International Journal of Human-Computer Studies* 118 (2018), pp. 38–46.
- [19] *Minecraft*. URL: <https://it.wikipedia.org/wiki/Minecraft>.
- [20] *No Man's Sky*. URL: https://it.wikipedia.org/wiki/No_Man'_s_Sky.
- [21] *Questionario*. URL: <https://www.pollfish.com>.
- [22] *rb2d*. URL: <https://docs.unity3d.com/Manual/class-Rigidbody2D.html>.
- [23] *Ricerca in ampiezza*. URL: https://it.wikipedia.org/wiki/Ricerca_in_ampiezza.
- [24] *Rogue*. URL: [https://it.wikipedia.org/wiki/Rogue_\(videogioco\)](https://it.wikipedia.org/wiki/Rogue_(videogioco)).
- [25] *SFXR*. URL: https://www.drpetter.se/project_sfxr.html.
- [26] Noor Shaker, Julian Togelius e Mark J Nelson. “Procedural content generation in games”. In: (2016).
- [27] Anthony J Smith e Joanna J Bryson. “A logical approach to building dungeons: Answer set programming for hierarchical procedural content generation in roguelike games”. In: *Proceedings of the 50th Anniversary Convention of the AISB*. 2014.
- [28] Matthew Stephenson e Jochen Renz. “Procedural generation of levels for angry birds style physics games”. In: *Proceedings of the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment*. Vol. 12. 1. 2016, pp. 225–231.
- [29] Emma R Tait e Ingrid L Nelson. “Nonscalability and generating digital outer space natures in No Man's Sky”. In: *Environment and Planning E: Nature and Space* 5.2 (2022), pp. 694–718.
- [30] *Tile component reference*. URL: <https://docs.unity3d.com/Manual/Tilemap-ScriptableTiles-Tile.html>.
- [31] *Tilemap component reference*. URL: <https://docs.unity3d.com/Manual/class-Tilemap.html>.
- [32] *tmcollider2d*. URL: <https://docs.unity3d.com/Manual/class-TilemapCollider2D.html>.

- [33] Julian Togelius et al. “Search-based procedural content generation: A taxonomy and survey”. In: *IEEE Transactions on Computational Intelligence and AI in Games* 3.3 (2011), pp. 172–186.
- [34] *Unity*. URL: <https://unity.com>.
- [35] *VSC*. URL: <https://code.visualstudio.com>.