PolygonSouls: Un joc d'acció amb vista zenital i esperit souls-like

Antoni Joan Passarell Dedeu

Resum— Aquest projecte consisteix en la creació d'un joc 2D amb vista aéria mitjançant el motor gràfic Godot. Els nivells es generaran procedimentalment de manera aleatòria, així com el seu contingut. Hi haurà enemics de diferents nivells; els més alts tindran una intel·ligència artificial que els hi permetrà contrarestar els atacs del jugador seguint el sistema de combat basat en Pedra-Paper-Tisora. En aquest treball s'ha creat un joc capaç de generar procedimental un nivell així com s'han implementat diverses intel·ligències artificials amb scripting dinàmic capaces de contrarestar els atacs del jugador donant-li així un repte a superar.

Paraules clau— Godot, generació procedimental, probabilitats amb pesos, "dynamic scripting", patrons d'atac, "souls-like", 2D, vista aérea, GDScript i intel·ligència artificial.

Abstract— This project consist in the creation of a 2D game with aerial view using the Godot Engine. The levels will be procedural generated randomly, as well as their content. There will be enemies of diferent levels, the highest ones will have an artificial intelligence that will allow them to counter the player's attacks following the combat system based on Rock-Paper-Scissors. In this project, a game has been created capable of procedurally generating a level as well as artificial intelligence with dynamic scripting capable of countering the player's attacks has been integrated, thus giving the player a challenge to overcome.

Index Terms— Godot, procedural generation, weighted proabibilities, dynamic scriptiong, attack paterns, souls-like, 2D, aerial view, GDScript and artificial intelligence.

1 Introducció

1.1 Concepte general

Els videojocs són un medi d'entreteniment audiovisual i digital amb el qual es pot interactuar mitjançant perifèrics. És van fer famosos als anys vuitanta amb títols com Tetris, Space Invaders o Pac-Man. Aquella mateixa dècada la popularitat va créixer enormement amb la sortida de les noves consoles que permetrien jugar des de la comoditat de casa teva.

Al millorar les capacitats tècniques de les consoles i els ordinadors, els jocs han anat evolucionant tant gràficament com conceptualment consolidant la superpotència que són ara mateix. A mesura que ha passat el temps les demandes dels jugadors han crescut i s'han anat creant nous gèneres per tal de satisfer-los. Un d'ells es el gènere "souls-like" el qual ha guanyat molta popularitat en el últims anys.

Com el seu nom indica els jocs de temàtica "souls-like" provenen de la saga "souls"[1], la qual és molt famosa per causar frustració als jugadors alhora que els motiva a que millorin les seves habilitats per tal d'aconseguir el que abans no podien. Aquest gènere requereix molta pràctica així com bons reflexes i precisió. Aquests jocs no són lineals i per tant es basen en l'exploració dels nivells. Al derrotar enemics aconseguiràs ànimes les quals et serviran per fer bescanvis. També es caracteritza per les lluites èpiques contra oponents colossals.

- *E-mail de contacte: antonijoan.passarell@e-campus.uab.cat*
- *Menció realitzada: Computació*
- *Treball tutoritzat per: Jorge Bernal del Nozal (Ciències de la computació)*
- *Curs 2020/21*

1.2 Objectiu general del projecte

L'objectiu general del treball és fer un videojoc creat amb Godot, amb temàtica souls i vista aèria 2D, que pugui ser jugat per qualsevol, sense importar les seves habilitats. Hi haurà l'opció d'anar lentament i aconseguir més ànimes per poder assolir tots els reptes minvant així la dificultat, però augmentant considerablement el temps de joc.

Al iniciar el joc, es generarà una masmorra aleatòria amb una estructura diferent cada cop. El contingut de cada sala es crearà a l'atzar, contenint el món com a mínim una entrada a la cripta i una foguera. El joc és centrarà en l'exploració de la masmorra, la lluita contra enemics per recol·lectar ànimes i la busqueda de l'entrada a la cripta, en la qual és podrà trencar el segell per tal d'accedir a la sala on es troba el guàrdia del nivell.

El sistema de combat es basa en el joc de pedra-paper-tisora, per tant, hi hauran tres tipus d'atacs: cos a cos (C), els distància (R) i màgic (M). Segons el mode d'atac que porti l'enemic i l'aliat el dany infligit se li aplicarà un multiplicador; si té avantatge (x1.5), si té desavantatge (x0.5) o si s'ha utilitzat el mateix tipus (x1).

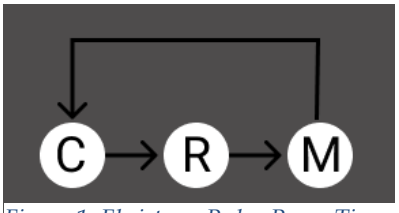


Figura 1: El sistema Pedra-Paper-Tisora de combat, C guanya a R, R guanya a M i M guanya a C.

El personatge principal podrà realitzar qualsevol del tres atacs, escollint segons convingui en cada situació. Es

disposarà d'una barra de salut i una de resistència: al perdre tots els punts de salut, el personatge mor. La resistència serveix per realitzar atacs, si no disposes de la quantitat necessària no podràs realitzar certs atacs; aquesta es regenera mentre estiguis en estat passiu.

Els enemics podran fer servir d'un a tres modes d'atac, depenent del seu nivell. El joc guardarà un històric dels atacs realitzats pel jugador, llavors els enemics faran estratègies segons el seu nivell per contrarestar la teva manera de jugar. Per exemple: si sempre ataques cos a cos, ells t'atacaran amb magia. Per tant, el jugador haurà d'intentar ser més llest que l'enemic i així contrarestar-lo. A més dificultat de l'enemic, més ànimes rebràs de recompensa al derrotar-lo.

Les ànimes tindran els següents usos: Trencar el segell per enfrontar-te al guàrdia del nivell, crear un punt per ressuscitar al morir i omplir tota la barra de vida.

2 ESTAT DE L'ART

Alhora de fer aquest projecte m'he inspirat en els següents videojocs: Hades, Dead Cells i Spelunky.

Hades [2] és una aventura d'acció de tipus "rogue-like" en el qual desafies el déu de la mort mentre intentes escapar del inframón de la mitologia grega. Combina els millors aspectes dels jocs anteriors de la companyia, el ritme frenètic de Bastion, la rica atmòsfera i la profunditat de Transistor i la narració de Pyre basada en els personatges.

Figura 2: Combat contra Megaera del joc Hades [2].

Dead Cells [3] és un "rogue-lite" de tipus acció-plataformes inspirat en Castelvania que et permet explorar un castell sempre canviant, assumint que puguis derrotar els enemics que vagin apareixent. El combat es semblant als "souls-like", ja que has d'evadir i atacar amb precisió, ja que els enemics són implacables.

Spelunky [4] és un joc de plataformes precursor en el tipus "rogue-like" on explores unes coves en les quals tens molta llibertat alhora de navegar els nivells ja que són totalment destructibles.

En tots els jocs presentats anteriorment es fa servir la generació procedimental per crear els seus nivells, això els hi dona molta rejugabilidad ja que cada partida és una experiència nova. Els jugadors han d'estar constantment adaptant-se als nous entorns.

En una pàgina anomenada tinysubersions [5] hi ha dos tutorials molt interesant que tracten sobre la generació procedimental de l'estructura del nivell i la col·locació dels obstacles en Spelunky.

En l'article de Chirs Hoekstra [6] podem veure els diferents mètodes utilitzats per implementar intel·ligència artificial (IA) als videojocs. Els mètodes descrits són:

- Scripting estàtic: Consisteix en fer un script que contingui totes les diferents accions que pot fer la IA i donades unes condicions decidir l'acció que farà l'agent. Com que no és adaptable, és molt fàcil de predir els seus moviments i per tant, de guanyar.
- Scripting dinàmic: Al ser adaptable, el script permet que l'agent pugui fer unes accions diferents donada la mateixa situació. Tot i no ser totalment aleatori, dona la sensació de ser-ho. Com que les condicions han estat creades prèviament per un programador la IA és limitada.
- Algorismes genètics: Aquest mètode es basa en la teoria de l'evolució, eliminant les solucions menys òptimes i passa els gens de les millors solucions a la següent generació. Després de moltes generacions, la IA feta amb algorismes genètics, és capaç d'arribar a solucions molt avançades.

3 REQUISITS

Inicialment, el projecte proposava la utilització dels motors gràfics Unity o Unreal Engine però finalment es va escollir Godot. Aquest entorn tenia molts avantatges sobre altres motors molt famosos: si comparem Godot amb altres motors gràfics, destaca per la seva accessibilitat ja que l'organització d'escenes i nodes és molt senzilla d'entendre i la corba d'aprenentatge és poc pronunciada, cosa que permet una ràpida introducció al motor, simplificant el prototipatge inicial.

La seva mida és molt reduïda amb comparació amb els altres i no és necessària la seva instal·lació, ja que és portable. Totes les eines necessàries per el desenvolupament del videojoc es troben dins de Godot, com podrien ser l'IDE, l'editor d'animacions o el creador de escenes, per tant no cal descarregar cap programa addicional.

L' IDE de Godot permet la utilització de dos llenguatges: C# i GDScript. A l'haver utilitzat principalment Python a la menció de Computació, utilitzaré GDScript, que és un llenguatge molt semblant i per tant ja tindré bastantes nocions del seu funcionament. Totes aquestes avantatges el fan la millor elecció possible pel projecte a realitzar.

3.1 Requisits del software

Els requisits del software [7] són:

- Windows 7 o més recent, macOS 10.10 o més recent, Linux (64-bit o 32-bit x86)
- Hardware que suporti OpenGL 2.1 com a mínim (a ser possible 3.3).

3.2 Requisits del hardware

Els requisits del hardware mínims especificats per Godot són:

- Hardware que suporti OpenGL 2.1.
- Qualsevol CPU remotament moderna de 2 nuclis és suficients per que funcioni sense problemes.
- Godot no necessita molta RAM així que amb 4 GB és suficient.
- Espai de emmagatzematge de 35 MB i si li sumem les plantilles de exportació 500 MB més.

Els requisits del hardware recomanats de Godot són:

- Hardware que suporti OpenGL 3.3:
  - AMD Radeon HD 7000 series o més recent.
  - NVIDIA GeForce 8 series o més recent.
  - Intel 3ra generació (Ivy Bridge) series o més recent.
- Comptar amb una CPU de 4 nuclis baixarà el temps de compilació considerablement.
- 8 GB de RAM.
- Espai de emmagatzematge de 35 MB al que hem de sumar 500 MB per afegir les plantilles de exportació.

Els requisits per executar el joc són:

- Una CPU moderna, mínim dos nuclis.
- Tenir 1 GB de RAM.
- Una GPU de gràfics integrats serà suficient per fer funcionar el joc, però es recomanables tenir una gràfica de les abans recomanades.

4 PLANIFICACIÓ DEL PROJECTE

S'han dividit les tasques a realitzar en aquest projecte en diferents parts. A cada feina se li ha assignat més temps del necessari per tal d'arreglar tots els imprevistos que puguin succeir.

- **Familitaritzar-se amb el motor:** En la fase inicial del projecte s'apendrà el funcionament bàsic de Godot familiaritzant-se així amb el motor gràfic. S'haurà de provar i entendre totes les eines necessàries per dur a terme el projecte. Aquesta fase s'extendra durant tot el mes de Octubre.
- **Disseny de nivells:** En la segona fase, es farà el disseny de nivells, el qual s'encarregarà de la generació procedimental del mapa i el contingut de cada sala. Aquesta fase s'extendra durant tot el mes de Novembre.

Riscs	Impacte	Conseqüència	Probabilitat	Pla de contingència
Generació procedimental repetitiva.	Fort	Afectarà l'experiència del jugador negativament.	Probable	Fer una recerca extensa d'implementacions procedimentals existents i avaluar l'eficàcia de l'algoritme mentre es desenvolupa.
IA dels enemics previsible.	Fort	Els enemics no suposaran un repte.	Probable	El nivell dels enemics s'ha de basar en les capacitats del jugador.
L'ordinador deixi de funcionar. Possible pèrdua de fitxers.	Crític	Retardaria el desenvolupament del joc o fins i tot la inhabilitat de continuar-lo.	Remota	Disposar d'un ordinador alternatiu amb el programari necessari per continuar el projecte. Per tal de no perdre fitxers, fer copies de seguretat.
Ser massa ambiciós amb el projecte.	Fort	Tenir parts del joc molt ben fetes i d'altres totalment inacabades.	Probable	Pensar en petit. Crear primer una base del que es vol implementar (prototip) i desenvolupar a partir d'allà.
Mala planificació inicial.	Fort	No acabar ni el videojoc ni el treball.	Probable	Fer una planificació realista de les tasques i el temps necessari per a fer-les. És important deixar un marge de temps addicional en cada tasca pel que pogués succeir.
Falta de familiarització amb el motor gràfic.	Crític	No saber com implementar els diferents requeriments del projecte	Remota	El primer mes servirà per familiaritzar-se amb el funcionament de Godot, així com totes les entitats necessàries per crear el videojoc.
Trobar errors crítics en la fase de proves del Gener.	Crític	Llocs concrets del joc deixessin de funcionar sense raó aparent.	Remota	A mesura que s'implementa el codi anar fent proves per comprovar i arreglar tots els errors que vagin sorgint.

Taula 1: Riscos i contingències del projecte.

- **Disseny d'enemics:** En la tercera fase es desenvoluparan els enemics, els seus patrons i la seva intel·ligència, que els permetrà reaccionar als moviments del personatge. Al final de Desembre s'acabarà la implementació del joc.
- **Fase de proves:** Aquesta serà la fase final, on es poliran els diferents aspectes del projecte. Es provarà el joc repetides vegades per tal comprovar que tot tingui coherència i funcioni correctament, s'hauran de corregir els errors que vagin apareixent. És podran fer petites modificacions per millorar l'experiència de l'usuari i fer el joc més entretingut. Aquesta fase s'extendra durant tot el mes de Gener.

El diagrama de Gantt amb totes les fases del projecte es pot trobar en l'apèndix 1.

5 RISCOS DEL PROJECTE

En la taula 1 es poden veure els riscos i contingències del projecte.

6 METODOLOGIA

La metodologia que s'ha utilitzat per dur a terme aquest projecte és Kanban [8]. Aquesta s'adaptava perfectament a les reunions setmanals que es feien amb el tutor del projecte. Kanban permet visualitzar totes les tasques que s'han de fer, prioritzar segons la importància de cada una i evitar fer feines innecessàries.

Al ser incremental i iterativa, a mesura que avança el treball s'actualitzen els requisits i les característiques del projecte. Per totes aquestes raons, és una metodologia molt flexible, que ha resultat amb informes precisos del que s'ha anat elaborant, hi ha permès una supervisió clara de les feines realitzades.

7 DESENVOLUPAMENT

A continuació s'explicaran totes les implementacions realitzades al videojoc així com tot allò que s'ha après en la seva elaboració.

7.1 Conceptes bàsics de Godot:

Per entendre el desenvolupament d'aquest projecte, primer caldrà explicar unes nocions bàsiques de Godot, el que més tard facilitarà la comprensió de conceptes més avançats.

Godot està basat en la programació orientada a objectes on cada objecte conformarà una escena [9]. Una escena és un arbre jeràrquic que contindrà múltiples nodes i cada un d'ells li donaran noves característiques i usos a l'objecte principal.

Dintre de cada escena hi haurà subordinació de nodes [10], conformant una jerarquia pare-fill, la qual s'utilitza per agrupar nodes en grups. Un possible ús d'aquesta funció és que si es mou el node pare, també ho faran tots els seus fills. Però, al contrari, si movem el node fill, el pare no es mourà. Cada escena tindrà un node principal, el qual serà l'arrel de l'arbre i englobarà tots els altres subnodes. Aquest node serà el que normalment contindrà el codi font ("script") de l'objecte, ja que simplificarà l'accés relatiu als seus fills en el codi i per tant el seu control.

Godot fa servir el sistema de coordenades cartesianes per determinar les posicions de cada node, però s'ha de tenir en compte que l'eix Y està invertit i la posició (0, 0) es troba a l'esquerra a dalt de la pantalla.

Tots els objectes creats podran ser instanciats en altres escenes, així que és bona pràctica dissenyar-los pensant en la seva reutilització posterior.

7.2 Disseny de nivell

La metodologia a seguir a l'hora de dissenyar els nivells serà totalment progressiva: es començarà implementant una escena bàsica, després es crearà una sala reutilitzable i finalment es farà la generació procedimental de la masmorra.

7.2.1 Escena bàsica:

Aquesta escena contindrà: un jugador capaç de moure's en les 8 direccions, una càmera estàtica i els blocs amb el que es faran les parets i portes. Aquesta escena serveix per fer proves de moviment i de col·lisió entre el jugador i els diferents blocs.

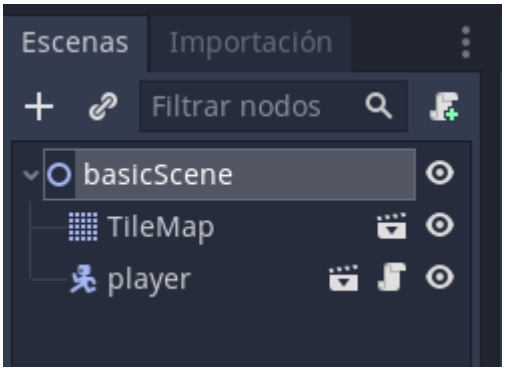


Figura 3: Estructura de la escena bàsica.

L'escena bàsica estarà formada per tres nodes:

- El node principal serà un "Node2D" [11] (anomenat "basicScene" en la figura 2), el qual és el més bàsic, aquest no té cap funcionalitat especial tot i que permet canvis en la seva posició i rotació. S'utilitza normalment per fer agrupacions de nodes.
- Un "TileMap" [12] és un node que ens permet agafar una imatge i dividir-la en subimatges les quals podrem usar com a caselles per construir l'estructura del nostre nivell.
  S'agafarà una imatge amb dos blocs de 32x32 píxels, un de color blanc simbolitzant una paret (el qual tindrà col·lisió) i un totalment transparent per crear les "portes" o blocs sense col·lisió. Usant el "TileMap", dissenyarem l'estructura de l'escena bàsica. Guardarem aquest node com a escena per poder-lo reutilitzar

posteriorment.

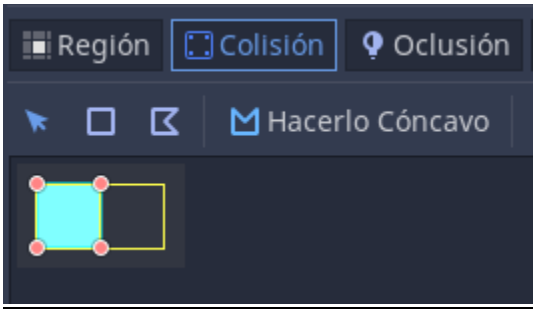


Figura 4: TileMap utilitzat per la creació de l'estructura de les sales.

- Finalment es farà servir un "KinematicBody2D" [13] (anomenat "player" en la figura 2), el qual permetrà crear el personatge que controlarà el jugador. Aquest node és un cos dinàmic per tant tindrà moviment i un "CollisionShape2D" [14] en forma de cercle que li permetrà col·lidir amb els blocs paret i travessar aquells que no tinguin col·lisió. Guardarem aquest node com a escena per poder-lo reutilitzar posteriorment.

7.2.2 Sala reutilitzable

Aquestes són les sales que utilitzarà la generació procedimental per dissenyar els nivells, per tant, cada sala serà un objecte que contindrà tota la informació rellevant sobre ella mateixa, així com funcions per la creació del seu contingut.

Aquesta escena començarà aparentment buida i s'anirà omplint a mesura que s'executi el codi. Contindrà:

- Un "Node2D" com a node principal, encarregat d'agrupar tots els subnodes i emmagatzemar el codi.
- Dos "Node2D" encarregats de marcar els ex-trems Nord-oest i Sud-est de cada sala, així com a referència a l'hora de crear l'estructura i d'actualitzar la posició de la càmera.
- Un node "Area2D" [15] (anomenat "roomArea"), el qual té una "CollisionShape2D" per detectar col·lisions i és el encarregat de detectar quan un cos entra i surt de la sala (només detectarà el "Kinematicbody2D" del jugador), per fer-ho s'utilitzaran els senyals [16] de _on_roomArea_body_entered(body) i _on_ roomArea _body_exited(body).
- S'utilitzarà l'escena guardada anteriorment amb el "TileMap" per crear l'estructura de la sala.
- Un node "Timer" [17], que servirà com a temporitzador, cada mig segon, cridarà el senyal _on_checkRoomClear_timeout() per comprovar si s'han assolit tots els requisits de la sala, si és així s'obriran totes les portes.
- En tots els passadissos, s'instanciarà [18] una porta, la qual s'obrirà en derrotar tots els enemics presents en la sala.

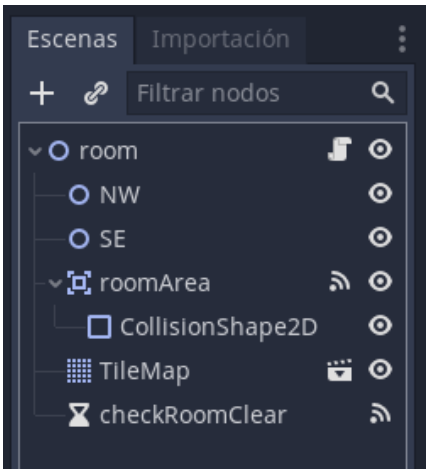


Figura 5: Estructura de la sala reutilitzable.

Per tal de crear totes les característiques d'una sala s'implementaran les següents funcions:

- drawBlockLine: Dibuixarà una línia amb el bloc que es passi per referència.
- drawRoom: Utilitzarà la funció drawBlockLine per dibuixar un rectangle que conformarà els límits de la sala.
- makeCorridors: Crearà els passadissos entre dues sales veïnes, canviant els blocs tipus paret per porta (sense col·lisió i transparents).
- makeDoors: Instanciarà una porta i la col·locarà a sobre de cada passadís.
- openDoors i closeDoors: Encarregades d'obrir i tancar les portes de les sales.
- setTypeOfRoom: Permet assignar el tipus de sala.
- _on_roomArea_body_entered(body): Quan el jugador entri en aquesta sala, segons el seu tipus es crearà un contingut o un altre.

7.2.3 Generació procedimental dels nivells:

En aquesta secció s'explicarà com funciona el disseny de nivells, concretament la generació procedimental. Cada nivell del joc conté un nombre de sales aleatòries igual al parametre maxRooms. Per simplificar, totes les sales tindran la mateixa mida.

El funcionament de la generació procedimental és:

1. Crear la sala inicial a la posició (0, 0).
2. Comprovar si hi ha sales a les quatre direccions i guardar les no construïdes.
3. Agafar aleatòriament una o dues i crear-les.
4. Anar a la següent sala creada en ordre cronològic i repetir el mateix procés fins que s'hagin creat el nombre de sales màximes preestablertes.
5. Un cop creades totes les sales es col·locaran els passadissos. Quan dues habitacions estiguin adjacents, es crearà un passadís entre elles, canviant així els blocs del tipus paret per porta, per tant passaran a no tenir col·lisió amb el personatge i permetran el pas.
6. Es col·locaran les portes a sobre dels passadissos.

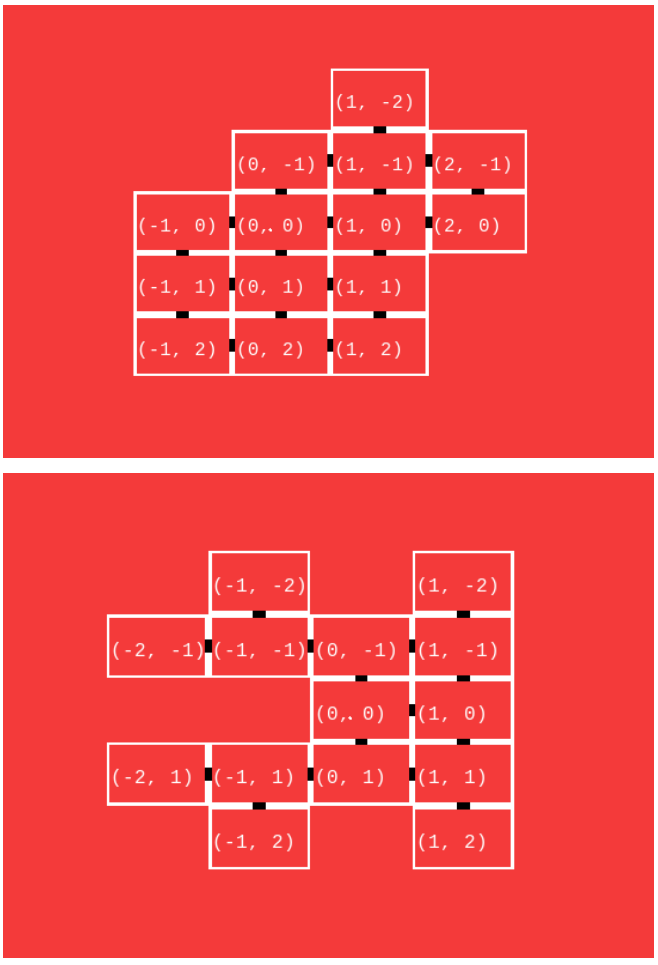


Figura 6a i 6b: En aquestes imatges es pot observar la diferencia entre dos nivells generats procedimentalment amb el mateix nombre de sales (14). A dintre de cada sala es pot veure la seva coordenada (x, y).

7.3 Càmeres

El projecte fa servir dues càmeres: una estàtica i una dinàmica. L'estàtica s'utilitza quan es posa en pausa el joc, aquesta càmera permet veure totes les sales prèviament visitades, semblant a les fotografies anteriors. La dinàmica és la que s'utilitza mentre es juga, aquesta només permet veure la sala en la qual està el personatge. En entrar en una sala diferent es canviarà la posició de la càmera a la nova sala i es farà una transició fluida entre elles. A més a més, tindrà dos "Labels" (nodes de text) els quals seran; un comptador d'ànimes aconseguides i un de temps.

7.4 Jugador

El jugador és un cos dinàmic que té moviment i pot col·lidir amb l'entorn. S'ha de tenir en compte que s'ha de normalitzar la velocitat, sinò el personatge aniria molt més ràpid movent-se diagonalment que en les quatre direccions bàsiques. Aquesta escena conté:

- Un node principal "KinematicBody2D", explicat en l'apartat 7.2.1.
- Un node "Sprite2D" amb la imatge del personatge, en aquest cas un cercle blanc.
- Una escena anomenada "playerUI", la qual contindrà:
  - Una barra de vida que servirà per emmagatzemar els punts de salut del personatge, en arribar a zero el jugador morirà i s'anirà a la pantalla de "Game Over".
  - Una barra de resistència que servirà per realitzar els diferents atacs, cada un d'ells consumirà un nombre determinat de punts de resistència: en cas de no disposar del nombre necessari no es podran realitzar certs atacs.
- Els tres patrons d'atac: Cos a cos (C), a distància (R) i màgic (M). S'utilitzarà un node "Label" (de text) per representar el mode d'atac actual al centre del personatge. Aquests atacs es poden realitzar en qualsevol de les 4 direccions. Els atacs a distància i màgic són escenes que s'instanciaran a l'hora de fer els atacs i s'eliminaran en xocar o sortir fora de la pantalla, alliberant així espai de memòria. S'utilitzarà un node "Timer" per controlar la cadència d'atacs a distància.
- Dues variables anomenades PA ("Player Attacks") i PAP ("Player Attack Pattern"), que seran crucials per fer la intel·ligència artificial dels "Spinning Enemies" i que s'actualitzaran cada vegada que el jugador realitzi un atac. Aquestes variables signifiquen:
  - PA: Guardarà els deu últims atacs que ha realitzat el jugador.
  - PAP: Serà un diccionari que guardarà les probabilitats de cadascun dels tres atacs segons els PA .
- Un node "Area2D", que l'utilitzarem com a "hitbox" per saber quan el personatge és atacat. Aquesta àrea permetrà detectar la col·lisió entre aquesta àrea i un altre o bé un cos. Tindrà un "CollisionShape2D" com a fill, el qual delimitarà la zona de detecció de col·lisió d'aquesta àrea. Una bona practica per la "hitbox" és ferla més petita que la col·lisió del "KinematicBody2D", d'aquesta manera no tindràs mai la sensació que has perdut salut d'un atac injustament. Segons el sistema de combat Pedra-Paper-Tisora, es rebran aventatges o desavantatges segons els enfrontament.
- Un node "AnimationPlayer" el qual permetrà crear animacions en l'escena jugador. Servirà per animar l'atac cos a cos. Per crear aquesta animació es necessitaran 3 pistes:
  - Sword: Servirà per girar 90 graus el node pare de la "Sprite2D" de l'espasa i fer-la visible només mentre ataca.
  - CollisionShape2D: Activarà la col·lisió de l'espasa mentre ataqui i la desactivarà a l'acabar.
  - Player: Canviarà el valor del booleà "canAttack", encarregat de permetre (true) o prohibir (false) que el jugador ataqui. Durant l'animació "canAttack" serà fals.

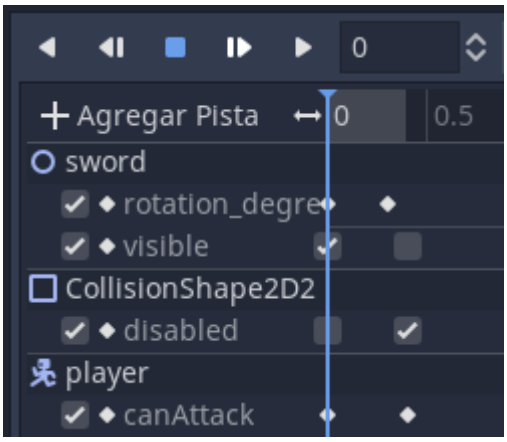


Figura 7: "AnimationPlayer" amb totes les pistes de l'animació d'atac cos a cos.

Per evitar crear una animació per cada una de les direccions d'atac és rotarà el node "sword" a la direcció i llavors es farà l'animació.

7.5 Enemics

En entrar a qualsevol sala, es crearà un nombre aleatori d'enemics entre dos paràmetres preestablerts (minNumEnemies i maxNumEnemies). Tots els enemics heretaran un "script" anomenat "enemies.gd", el qual contindrà propietats i funcions comunes entre tots ells. Aquest codi facilitarà la creació de nous enemics, ja que es podrà aprofitar molta informació. Cada enemic tindrà el seu propi "script", el qual interactuarà amb l'heretat per crear les seves individualitats.

Les escenes que continguin enemics entraran dins del grup [19] "Enemies", això servirà per detectar més fàcilment els nodes enemics.

El codi contindrà un "onready var" [20] que s'usarà per guardar un node, mitjançant una ruta relativa, en una variable del codi per poder cridar les seves funcions. En aquest cas servirà per obtenir el node jugador i cridar una funció per retornar la seva posició en coordenades cartesianes.

Tots els enemics descrits a continuació tindran una "hitbox" per detectar quan el jugador els ataca.

S'han creat quatre enemics:

- Kamikazes: La seva funció principal és detectar la posició del personatge i anar cap a ell: quan entren en contacte amb el jugador exploten [21] (atac cos a cos), causant-li així pèrdua de punts de salut. Aquesta funcionalitat s'ha implementat utilitzant una "Area2D" per detectar la col·lisió amb el jugador. Quan un atac del jugador entri a la "hitbox" del "Kamizake", aquest morirà.
- Turret: És un enemic estàtic, que rota sobre sí mateix apuntant en la direcció amb el jugador. Quan es carrega la barra, dispara una fletxa (atac a distància), que explota amb contacte a un altre cos. Quan un atac del jugador entri a la "hitbox" de la "Turret" aquesta morirà.
- Spinning Enemy: Enemic perseguidor amb tres punxes rotatòries que li envolten el cos, aquestes són unes "area2D" que serviran tant d'atac com de defensa contra atacs a distància i magics. Aquestes punxes estaran dintre d'un node "position2D" que serveixen com a centre de rotació. Aquest enemic tindrà una barra de vida com la del jugador, per tant, per derrotar-lo caldran diversos atacs. Aquest enemic té tres patrons d'atac:
  - Atac cos a cos: Les punxes giraran més ràpid dificultant els atacs a distància i màgics. En entrar en contacte amb el jugador li restarant punts de salut.
  - Atac a distància: Utilitzarà les fletxes explosives de les torretes.
  - Atac màgic: Llençarà unes esferes blanques, més lentes que l'atac a distància però amb una àrea de col·lisió més gran.

    L'enemic agafarà las PAP del jugador i crearà la variable EAP ("Enemy Attack Pattern"), que seran les mateixes probabilitats que el PAP, però utilitzant els atacs que contrarestin els del jugador. L'enemic decidirà aleatòriament quin atac usar segons els pesos de les probabilitats [22] ("dynamic scripting"[6]) de cada atac (utilitzant el EAP), per tant tindrà més probabilitats de fer un atac que contraresti la majoria dels PA del jugador que aquells que ha fet servir menys vegades.

    El seu moviment serà perseguir el jugador i quan tingui menys del 50% de la vida canviarà de color, augmentant la seva velocitat de moviment.
- Decagon Boss: Per tal d'arrivar a l'enemic final d'aquest joc, primer s'han de reunir mil S ("souls"/ànimes) per poder pagar el preu per accedir a la cripta. Alla espera l'enemic final, el qual:
  - Té deu vides.
  - Només pot ser ferit per atacs cos a cos i mentres estigui de color blanc o bé quan comença a canviar de color.
  - Dispara en deu direccions atacs màgics els quals contraresten els atac cos a cos del jugador.

Al derrotar-lo es pararà el cronometre i indicarà el temps total utilitzat per passar-se el joc.

8 RESULTATS

La bona planificació i metodologia seguida al projecte han permès assolir tots els objectius proposats al principi així com superar tots els reptes que han anat apareixent. Els resultats del projecte es poden veure en el següent enllaç: https://youtu.be/GeGw5vCwnQU

9 CONCLUSIONS

S'ha creat un joc tipus "souls-like" el qual disposa d'un sistema de generació procedimental de nivells altament

aleatori però amb sentit i perfectament escalable. S'ha implementat un enemic amb intel·ligència artificial, capaç d'adaptar-se als atacs del jugador i contrarestar-los de manera òptima gràcies al "dynamic scripting" [6] (decidir aleatòriament segons les probabilitats amb pesos de cada atac). A més a més, per evitar ser mecànic i predictible, de tant en tant utilitzarà atacs pocs probables per sorprendre el jugador.

Les dificultats sorgides en aquest treball han estat:

- La gran quantitat de feina que requereix fer un joc, ja sigui, ideant, dissenyant, programant o bé testejant.
- Tota la recerca necessària per familiaritzar-se i solucionar els problemes que sorgien amb el motor Godot i el seu llenguatge de programació GDScript.
- Triar el mètode a utilitzar per implementar la generació procedimental del nivell i el seu contingut. Com aplicar aquest mètode de manera efectiva, amb sentit i que fos totalment escalable és un dels punts claus a l'hora d'assolir aquest projecte.
- Com implementar la intel·ligència artificial del joc perquè resultes un repte lluitar contra ella i evitar que fos fàcilment predictible.

El treball es podria ampliar en els següents aspectes:

- En el disseny de nivell es podria crear sales amb diferents mides. En cada sala es podrien crear obstacles procedimentalment.
- Es podrien crear múltiples nivells on la dificultat anés incrementant a mesura que s'avança.
- Es podrien crear nous enemics amb una intel·ligència artificial més avançada que la presentada en el treball que aprenguessin a mesura que van enfrontant-se amb el jugador.
- En l'apartat gràfic, es podria millorar totes les "sprites" i les seves animacions, creant d'aquesta manera una millora atmosfera pel videojoc.
- En l'apartat de l'àudio, es podria afegir música d'ambient i efectes de so, fent així el joc molt més immersiu.
- Es podria implementar un menú per la configuració de totes les variables del joc, podent així personalitzar totalment l'experiència de joc.

Agraïments

M'agradaria agrair a la meva família pel suport incondicional que sempre m'han donat i al meu tutor per fer el projecte amè, alhora que motivar-me cada setmana per millorar-lo al màxim possible. Ha estat un plaer treballar amb ell.

Bibliogràfia

[1] Bandai Namco. (7 d' Octubre de 2011). *Dark Souls*. Recuperat de https://en.bandainamcoent.eu/dark-souls/dark-souls

[2] Super Giant Games. (7 de Desembre de 2018). *SuperGiantGames*. Recuperat de https://www.supergiantgames.com/blog/hades-faq/

[3] Motion Twin. (2018). *Dead Cells*. Recuperat de https://dead-cells.com/

[4] Mossmouth. (2012). *Spelunky World*. Recuperat de https://spelunkyworld.com/whatis.html

[5] Kazemi, D. (2013). *Tiny Subversions*. Recuperat de http://tinysubversions.com/spelunkyGen/

[6] Hoekstra, C. (2008). *Adaptive Artificially Intelligent Agents in Video Games: A Survey* . Waterloo, IA 50701 . Recuperat de http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.137.4978&rep=rep1&type=pdf

[7] Calinou. (2018). *Godot*. Recuperat de https://godotengine.org/qa/23206/what-specs-are-good-for-computer-that-planning-use-godot-with

[8] APD. (30 de Gener de 2019). *Metodología Kanban*. Recuperat de https://www.apd.es/metodologia-kanban/

[9] Godot Community. (2020). *Godot Scene Organization*. Recuperat de https://docs.godotengine.org/en/stable/getting_started/workflow/best_practices/scene_organization.html

[10] Godot Community. (2020). *Godot Nodes*. Recuperat de https://docs.godotengine.org/en/stable/classes/class_node.html

[11] Godot Community. (2020). *Godot Node2D*. Recuperat de https://docs.godotengine.org/en/stable/classes/class_node2d.html

[12] Godot Community. (2020). *Godot TileMap*. Recuperat de https://docs.godotengine.org/en/stable/tutorials/2d/using_tilemaps.html

[13] Godot Community. (2020). *Godot KinematicBody2D*. Recuperat de https://docs.godotengine.org/en/stable/tutorials/physics/kinematic_character_2d.html

[14] Godot Community. (2020). *Godot CollisionShape2D*. Recuperat de https://docs.godotengine.org/en/stable/classes/class_collisionshape2d.html

[15] Godot Community. (2020). *Godot Area2D*. Recuperat de https://docs.godotengine.org/en/stable/classes/class_area2d.html#class-area2d

[16] Godot Community. (2020). *Godot Signals*. Recuperat de https://docs.godotengine.org/en/stable/getting_started/step_by_step/signals.html

[17] Godot Community. (2020). *Godot Timer*. Recuperat de https://docs.godotengine.org/en/stable/classes/class_timer.html#class-timer

[18] Godot Community. (2020). *Godot Packed Scene*. Recuperat de https://docs.godotengine.org/en/stable/classes/class_packedscene.html#class-packedscene

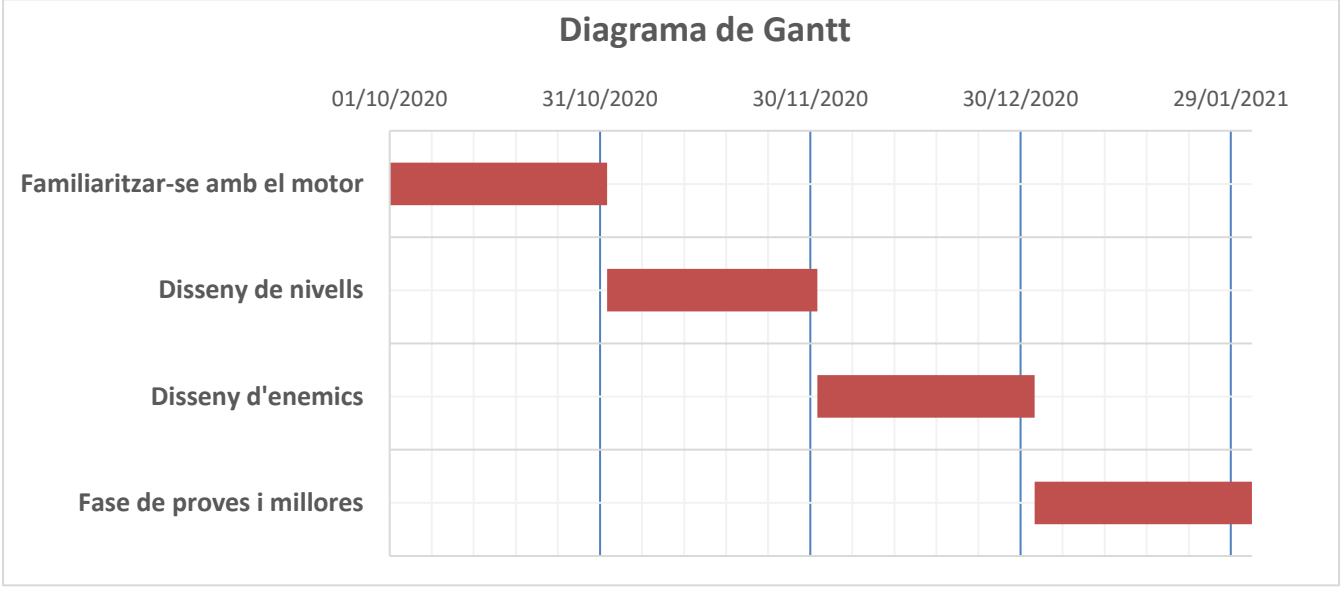
[19] Godot Community. (2020). *Godot Groups*. Recuperat de https://docs.godotengine.org/en/latest/tutorials/scripting/groups.html

[20] Godot Community. (2020). *Godot GDSCript basics*. Recuperat de https://docs.godotengine.org/en/stable/getting_started/scripting/gdscript/gdscript_basics.html

[21] Hiulit. (9 de Desembre de 2019). *Github*. Recuperat de https://github.com/hiulit/Godot-3-2D-Fake-Explosion-Particles

[21] Yuri Sarudiansky. (7 d'Octubre de 2020). *Kehom's Forge*. Recuperat de http://kehomsforge.com/tutorials/single/GDWeightedRandom/

APÈNDIX



Apèndix 1: Diagrama de Gantt amb totes les fases del projecte.