Tamed Souls
Un joc de plataformes 2D amb esperit souls-like

Jan Moros-Esteban

Resum– Aquest article recull el desenvolupament de *Tamed Souls*, un videojoc de plataformes en dues dimensions amb elements d'altres gèneres com el *Souls-like* que va ser creat utilitzant el motor *Godot*. El món en què es juga es genera de manera procedural a cada nova partida, i s'hi poden trobar presoners a rescatar i fogueres on descansar dels enemics que el plaguen. Els rivals tenen el seu propi sistema d'intel·ligència artificial que els permet crear estratègies d'atac en funció de la del jugador. Al final del nivell apareix un enemic més fort i amb patrons de moviment i atac diferents que s'ha de derrotar per acabar el joc. Els resultats obtinguts son força satisfactoris, tenint ja una primera versió funcional, assolint tots els objectius plantejats al principi del TFG.

Paraules clau– Generació procedural, Godot Engine, IA de Videojocs, Plataformes, Videojoc.

Abstract– This article explains the development of *Tamed Souls*, a two-dimensional platformer videogame with elements from other genres such as *Souls-like*, created through the *Godot* engine. The world the game takes place in is proceduraly generated in each new game, and it contains prisoners to be rescued and campfires to find some rest after fighting against the enemies that plague the environment. Rivals have their own Artificial Intelligence, which allows them to develop attack strategies based on the player's actions. At the end of the level a stronger enemy appears, with different movement and attack patterns: it must be defeated to finish the game. Current state of the game is rather satisfactory, having already a fully functional version that fulfills all the requirements planned at the beginning of this TFG.

Keywords– Godot Engine, Platformer, Procedural generation, Videogame, Videogame AI.

1 Introducció

1.1 Concepte general

Tamed Souls és un videojoc que surt de la visió de Jorge Bernal, tutor d'aquest treball. El joc parla de l'aventura de dos germans per derrotar Seoh, un déu d'una altra dimensió que ha arribat al món per treure el poder de les ànimes dels seus habitants. Per fer-ho, hauran d'endinsar-se a les masmorres de Sachrol, un poderós mag del que Seoh va fer-ne servir el cos com a hoste. Allà trobaran 5 criptes -una per cada raça que habita el planeta-, i hi hauran de destruir el segell que dóna poder a Seoh.

El joc està dividit en dues parts amb jugabilitats molt diferents. La primera està ambientada a una de les masmorres, on el jugador haurà de trobar l'entrada a la cripta, tot passant per sales amb enemics portats per Seoh i misteriosos artefactes que li milloraran les habilitats. No només haurà de derrotar enemics per sobreviure, també en podrà recol·lectar les ànimes, que li serviran com a moneda per a la segona part. El jugador canviarà a aquesta quan trobi la cripta i s'hi endinsi, i és la que es desenvolupa en aquest projecte.

Es podria definir la seva jugabilitat com un *action-platformer*. Com a subgènere dels videojocs de plataformes, implica que una part important de la interacció es basa en caminar i saltar entre diverses plataformes [1]. D'aquest gènere també n'extreu característiques com un disseny de nivell en dues dimensions però amb una única direcció (en aquest cas, cap a la dreta) o l'ús de *scroll* lateral, és a dir que la càmera es mou cap a la dreta independentment del jugador, per forçar-lo a avançar. A *Tamed Souls* no es mou sola, però sí que es mou únicament a la dreta, impedint-li retrocedir.

Es diferencia, però, en les accions que pot fer el jugador i en com es comporten els enemics. Als jocs de plataformes clàssics, la jugabilitat es centra en el moviment, i els enemics es solen derrotar saltant-hi a sobre o amb l'ús d'elements de l'entorn. A *Tamed Souls* el jugador pot atacar tant amb l'espasa com amb l'arc o llançant projectils màgics. De la mateixa manera ho poden fer els enemics, canviant completament l'estratègia a l'hora de jugar.

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

Com s'indica al nom del treball, també té elements que recorden al gènere *souls-like* [2]: al final de cada cripta el jugador es trobarà amb un *boss* (un enemic més important, i més difícil de vèncer) i es penalitza la mort del jugador fins al punt de perdre la partida.

Pel que fa a les competències de la menció de Computació, gran part del desenvolupament del projecte es centra en el disseny de dos algorismes intel·ligents.

El primer s'encarrega de la generació del nivell, que serà principalment aleatòria amb certs paràmetres com la longitud horitzontal del nivell. Haurà de tenir en compte restriccions com que les plataformes siguin assolibles pel jugador però tampoc massa fàcils o que el nombre d'enemics sigui raonable però difícil i que estiguin ben posicionats al terreny.

L'altre es tracta de la Intel·ligència Artificial dels enemics. Aquests poden ser de tres nivells, cada un amb més vida i més habilitats (p.ex. els de nivell 1 només disposen d'una arma cos a cos). El comportament dels enemics dependrà del seu nivell i, sobretot, de l'historial d'atacs del jugador. Allà es tindran en compte factors com la distància a l'enemic que ha rebut l'atac, el temps des de l'últim atac o els punts de vida restants.

1.2 Objectius del projecte

L'objectiu principal del projecte és el correcte desenvolupament del videojoc descrit a l'apartat anterior, prioritzant els elements d'algorísmia però també amb atenció a que el resultat final sigui no només jugable si no el més divertit possible.

Per assolir-lo, s'han definit tres objectius específics:

- Implementar un algorisme generador de nivells capaç de crear un nivell aleatori que sigui jugable i amb tots els elements que hi han d'aparèixer en posicions coherents.
- Implementar un agent intel·ligent per als enemics, específic per les característiques de Tamed Souls i que suposi un repte per al jugador.
- Integrar aquests dos mòduls en un videojoc amb les mecàniques descrites, que tingui una meta assolible i sigui divertit per al jugador alhora que un repte.

2 Estat de l'art

Pel que fa al disseny de la jugabilitat de la cripta s'ha tret inspiració de videojocs actuals però també de clàssics.

Dead Cells [3] (2018) i *Blasphemous* [4] (2019) van ser les primeres inspiracions per al disseny. Tot i que tenen estètiques i jugabilitat diferents, ambdós son *action-plaformers* en dues dimensions. Es diferencien, però, del disseny final de *Tamed Souls* en que el seu sistema de plataformes és no-lineal. Això vol dir que, enlloc de ser en una única dimensió com els plataformes clàssics, el disseny de nivells també té certa verticalitat i incorpora elements del gènere *Metroidvania* com el fet d'haver d'aconseguir claus per obrir portes en altres zones del mapa, que obliguen al jugador a tornar per on ja ha passat.

A part, *Dead Cells* també té part de *Souls-like*, amb enemics i *bosses* que ataquen amb patrons o el concepte de

Fig. 1: Personatge de Dead Cells disparant a un enemic.

PermaDeath (mort permanent), és a dir que si el personatge mor, el jugador ha de tornar a començar des de l'inici.

Finalment, però, el sistema de plataformes és més similar al de la clàssica saga *Megaman* [5]. La seva primera entrega, *Rockman I* al japó, *Megaman 1* a la resta del món (1987), és considerat el primer *action-platformer* de la història.

Aquest segueix el disseny de nivells típic del gènere, amb una única direcció, però afegeix al final de cada nivell un *boss* amb un disseny específic i unes habilitats i debilitats característiques.

A cada nivell, es van afegint enemics de diferents tipus, i el jugador guanya diversos tipus d'armes. Part de la jugabilitat es basa en que certes armes son més poderoses contra algun tipus d'enemic. El sistema de combat de *Tamed Souls* conté elements similars.

Pel que fa a l'ús d'una intel·ligència artificial que s'adapti als moviments i a les estratègies del jugador, existeixen models basats en *scripting dinàmic* [6], on una base de dades de *Normes* determina l'script de la IA. Aquesta base de dades s'actualitza segons els moviments del jugador. També existeixen els models *Case-Based* [7], que incorporen *Opponent Modeling*.

Degut a la seva naturalesa, aquests últims son més exitosos en videojocs multijugador i per això a *Tamed Souls* s'ha optat per un model basat en *Normes*. La intel·ligència artificial dels enemics té en compte l'estructura pedra-paper-tisores, el mode d'atac del Personatge i la distància a la que es troba per decidir la seva estratègia.

3 Requisits

3.1 Requisits software

Per al desenvolupament d'aquest projecte existien tres grans opcions amb llicència gratuïta pel que fa al motor del videojoc: Godot [8], Unity [9] i Unreal Engine [10]. Degut a que un objectiu secundari és fusionar les dues parts del joc en un mateix projecte, es va acordar utilitzar el mateix motor.

Aquest acord va descartar Unreal Engine, perquè cap dels dos estudiants hi tenia experiència. Finalment es va escollir Godot per diverses raons:

- Un dels estudiants ja havia treballat amb ell.
- El seu llenguatge, GDScript, és senzill i basat en python, aquest últim molt utilitzat per aplicacions d'algorísmia.
- És Open Source.

TAULA 1: RISCOS DEL PROJECTE

Risc	Impacte	Possibilitat	Pla de Contingència
Els enemics són massa difícils de derrotar, esdevenint una experiència frustrant.	Mitjà-Alt	Probable-Alta	Editar els paràmetres de l'agent intel·ligent per baixar-ne la dificultat.
L'scroll lateral dificulta massa la jugabilitat.	Alt	Probable	Disminuir-ne la velocitat màxima. Si el problema persisteix, substituir l'Scroll lateral per un temporitzador que faci tornar al jugador a la masmorra quan s'acabi.
El joc no és divertit per a l'usuari.	Mitjà	Probable	Analitzar el focus del problema i fer els canvis possibles sense haver de redissenyar el joc completament.
El termini no permet la implementació d'un boss final.	Mitjà	Probable	Eliminar aquest element del disseny del joc o bé substituir-lo per un enemic ja existent amb més punts de vida i més poderós.
L'algorisme de generació de nivells no genera nivells assolibles, son massa difícils o massa fàcils.	Alt	Remota	Si el problema és la dificultat, es pot jugar amb els paràmetres fins trobar un punt satisfactori. Si no, s'hauria de descartar l'algorisme i dissenyar el nivell manualment.

- És més intuïtiu per als projectes en dues dimensions que Unity, l'altra opció.

Pel que fa a les eines, l'editor gràfic de Godot compta amb un IDE força complet i dissenyat per ser integrat amb la resta de components. A més, les prediccions de l'editor de text obtenen informació de l'editor gràfic, facilitant així la tasca de programació. És per això que en aquest projecte no s'utilitzen eines de desenvolupament externes a l'editor de Godot.

3.2 Requisits hardware

Per al desenvolupament, les especificacions hardware requerides corresponen a les del motor Godot. Segons la pàgina oficial [11], l'únic requisit és tenir una targeta gràfica compatible amb OpenGL 2.1, i a la botiga de contingut digital Steam [12] els desenvolupadors recomanen que també sigui compatible amb OpenGL 3.3.

Per poder jugar al joc final es requereix Windows 7 o posterior o una distribució amb interfície gràfica de Linux. Quant a hardware, els requisits mínims son una targeta gràfica compatible amb OpenGL 3.3 i 35 MB de memòria disponible al disc.

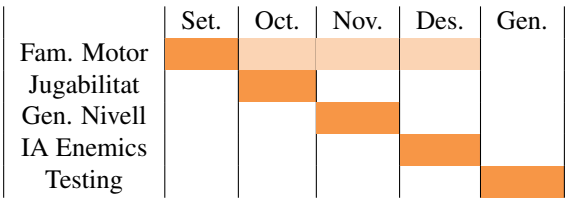
4 PLANIFICACIÓ DEL PROJECTE

El projecte consta de cinc fases principals, cada una d'una durada aproximada d'un mes. Son les següents:

1. **Familiarització amb el motor**. Durant el mes de setembre el focus principal serà l'aprenentatge de l'editor de Godot i del llenguatge que fa servir el seu motor, GDScript. Aquesta fase, però, s'estendrà per tot el desenvolupament.
2. **Mecàniques de joc.** L'octubre es dedica a implementar el moviment i atacs del jugador, el sistema de col·lisions per la generació del nivell, el sistema de punts de vida i estamina i els punts de reaparició. També s'implementa una versió inicial reduïda de l'agent intel·ligent que controla els enemics.
3. **Disseny procedural del nivell.** Al novembre s'inicia la part algorísmica del treball. Es dissenya i implementa un algorisme capaç de generar un nivell jugable amb cert grau d'aleatorietat i col·locar-ne enemics, punts de reaparició i aliats empresonats seguint un patró lògic.
4. **Intel·ligència artificial dels enemics.** L'última etapa de desenvolupament és la que es du a terme al desembre, i es centra en dissenyar un agent intel·ligent per als enemics, que tingui en compte el comportament anterior del jugador per planejar els seus moviments.
5. **Testing i refinament.** Finalment, al gener es fan proves per comprovar el correcte funcionament del joc, s'arreglen els errors i es fan els petits ajustaments que calguin per millorar l'experiència de joc.

La planificació temporal d'aquestes fases es pot representar gràficament en un Diagrama de Gantt, com el que es pot veure a la Figura 2.

TAULA 2: DIAGRAMA DE GANTT SIMPLIFICAT



5 RISCOS DEL PROJECTE

A la Taula 1 es poden veure els riscos inicials plantejats inicialment pel projecte i el seu pla de contingència.

6 METODOLOGIA

Degut a la naturalesa iterativa del desenvolupament de videojocs, durant el transcurs del projecte s'han fet reunions set-

manals amb el tutor, per poder veure la progressió, i definir nous objectius setmana a setmana. Per adaptar-se fàcilment als canvis de requeriments s'ha escollit una metodologia *Àgil* [13]. Més concretament, s'han utilitzat taulers *Kanban* amb elements d'*Scrum* [14].

Es parteix d'un *backlog* de tasques, que s'actualitza setmanalment a les reunions. Aquestes, dividides per temes, s'afegeixen al tauler Kanban, i allà es mouen del seu tauler al de "En progrés" i finalment al de "Completada". Per crear el tauler s'ha utilitzat la eina web *Trello* [15].

7 DESENVOLUPAMENT DEL PROJECTE

En aquest apartat s'explicarà en detall el procés que s'ha seguit durant el desenvolupament del projecte, i el seu funcionament. Es divideix en Entorn, Personatge, Món, Enemics i *Boss*.

7.1 Entorn

Seguint els primers vídeos de les sèries de tutorials enfocats a crear videojocs senzills de Plataformes 2D a Godot de UmaiPixel [16] i HeartBeast [17], es va crear una primera escena del joc amb un terra i un personatge que es veu afectat per la gravetat i pot moure's i saltar segons l'input del jugador (a través del teclat o d'un comandament de videoconsola).

Continuant amb la llista de l'usuari HeartBeast, es va crear un *Tilemap* a partir d'una imatge proveïda a la descripció del mateix vídeo i que es pot veure a la Figura 2. Aquest element permet tenir estructurats, en una quadrícula altres elements visuals (en aquest cas, el terra), i assignar-ne una forma de col·lisió. Això permet crear nivells molt fàcilment perquè crea la col·lisió directament quan es col·loca un bloc de terra.

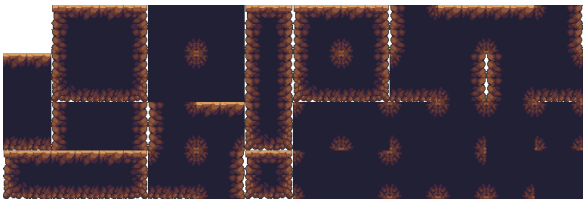


Fig. 2: Imatge a partir de la qual es genera el Tilemap

A més, es pot crear una màscara de bits que representa la connexió amb altres blocs del *Tilemap*, aconseguint que únicament col·locant els blocs amb la forma que es vol, aquests es mostrin orientats automàticament. Aixo es pot veure a la Figura 3.

Per acabar amb la sèrie de tutorials de HeartBeast, es va crear un fons *Parallax*. Això significa que es mou amb cert retard de temps relatiu al moviment de la càmera, fet que crea un efecte de profunditat.

A part de la jugabilitat principal, s'ha implementat un sistema de menús. Així, quan s'inicia *Tamed Souls* el jugador pot triar: Crear una nova partida, Continuar una partida existent, veure el Rànquing o Sortir del joc. Si escull el primer, passa a un altra pantalla on decideix la mida del nou món, ja sigui Curt, Mitjà o Llarg. Un cop s'ha creat, s'activarà l'opció de Continuar des de l'últim punt de guardat al menú inicial.

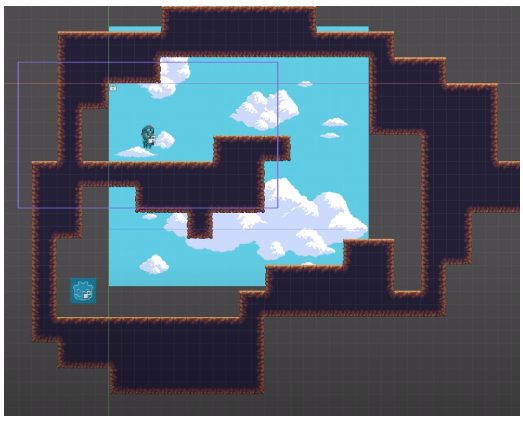


Fig. 3: Exemple del resultat de la utilització d'Autotiles

Pel que fa al Rànquing, emmagatzema els 5 millors temps per a cada mida del món, i els mostra a la pantalla corresponent.

7.2 Personatge

L'heroi compta amb tres modes d'atac diferents, entre els quals el jugador pot alternar a través de dos botons assignats. Aquests son els modes *Melee* (Cos a cos), *Ranged* (a distància) i *Magic*. Encara que no s'ha implementat a la versió actual, es planteja que en futures versions l'*sprite* del personatge canviarà amb el mode d'atac per facilitar la identificació d'aquest.

En la versió que s'ha completat per a aquest projecte, s'utilitzen sprites adaptats dels que proporciona HeartBeast, amb text sobre el cap del jugador indicant el mode. Es pot veure a les figures 7 i 8.

Pel que fa a la implementació dels atacs, existeixen dues escenes pels dos tipus de projectil: boles de foc i fletxes. Les escenes a Godot permeten crear noves instàncies de l'objecte des-del codi. En aquest cas, contenen un *sprite* i una forma de col·lisió que s'adapta a l'objecte corresponent.

Ambdós projectils tenen un funcionament senzill i similar. Les boles de foc es mouen infinitament en una direcció, fins que algun element extern les fa desaparèixer. Les fletxes es diferencien en el fet que no tenen rang infinit, si no que és un nombre determinat. Quan es separa certa distància del jugador, se n'activa la gravetat i es destrueix quan toca el terra.

Per altra banda, a l'escena del personatge hi ha un element de tipus *Position2D* que emmagatzema una posició dins del món. Aquesta està col·locada estratègicament a l'alçada de la mà del personatge quan aquest fa l'animació de disparar.

A més, es mou amb el personatge i també canvia de banda quan aquest es gira. D'aquesta manera, a l'*script* que controla al personatge, quan el jugador pitja el botó de disparar es crea una instància del projectil a la posició indicada pel node *Position2D*. S'assigna també a la nova instància la direcció que ha de seguir, segons la del personatge.

Ambdós projectils desapareixen quan entren en contacte amb qualsevol altre cos. Però si aquest és un enemic o el personatge jugador (PJ), abans de fer-ho n'hi criden la funció *hit*, que s'encarrega de restar-ne els punts de vida corresponents i determinar si ha mort.

L'atac *Melee* funciona força diferent. Aquí s'utilitza d'un *RayCast2D*. Com es pot veure a la Figura 4, és representat dins de l'editor com una fletxa, i té dos elements importants: el punt on comença i al que apunta. El tipus d'element *RayCast2D* té un mètode *is_colliding*, que retorna *true* si hi ha algun objecte entre els dos punts, i *false* si no n'hi ha cap.

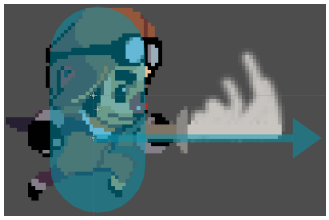


Fig. 4: Personatge a l'editor, amb la caixa de col·lisió i el *RayCast2D* visibles

En aquest cas, la fletxa surt del cos del personatge i arriba fins la punta de l'espasa en el moment de màxima extensió de la seva animació. Quan el jugador prem el botó d'atacar, es reprodueix l'animació i es comprova si hi ha algun objecte col·lidint amb el *RayCast2D*. Si n'hi ha i és del grup "enemics", en crida la seva funció *hit*.

Els tres tipus d'atac, a part de reduir els punts de vida dels enemics, els empenyen lleugerament en la seva direcció, aturant-los i allunyant-los lleugerament del personatge.

Per acabar la secció d'atacs es troba la jerarquia "pedra-paper-tisores", que ve determinada per l'esquema de la Figura 5. Dins del joc, els enemics també poden canviar entre els tres modes, pel que la jerarquia té efecte entre l'atacant i qui rep l'atac. Per exemple, si el PJ fa un atac *melee* a un enemic en mode *magic*, l'atac farà el seu valor de dany multiplicat per 1,5. En canvi, si qui el rep està en mode *ranged*, es veurà reduït fins la meitat. Si els dos elements estan en el mateix mode, no s'apliquen modificacions al dany.

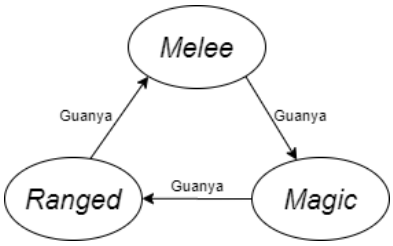


Fig. 5: Esquema de la jerarquia pedra-paper-tisores

El personatge també compta amb un sistema d'estat, que en monitoritza els punts de vida (*hp*, de *Health Points*) i els d'*estamina*. Estamina és un anglicisme molt popular dins la indústria dels videojocs, i es refereix a la resistència al cansament, o a l'energia del personatge. A *Tamed Souls*, representa l'energia necessària per realitzar els atacs.

Respectivament, els atacs màgic, a distància i cos a cos consumeixen 3, 2 i 1 punts d'estamina. Si no es disposa de suficients punts, no es realitza cap atac. Per recuperar-la, el jugador únicament ha d'esperar: es guanya 1 punt cada segon si no s'està atacant.

Tant per a la vida com per a l'estamina existeix una variable que té un valor flotant entre 0 i 10. Perquè l'usuari pugui visualitzar-les, es representen com un percentatge sobre el valor màxim, en forma de les barres de la Figura 6. Formen part de la *GUI*, sigles de "Interfície Gràfica de l'Usuari" en anglès. Godot permet crear-la amb un tipus de node específic [18], que no interactua a nivell físic amb el món i sempre es sobreposa a la visió de la càmera.

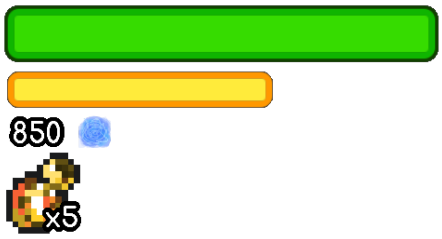


Fig. 6: *GUI* del joc, amb les barres i els comptadors

Els altres dos elements de la *GUI* son comptadors, un per les ànimes i un altre pels càntirs d'*estus*, per ordre descendent.

L'*estus* és un element característic de la saga *Dark Souls*, i tant allà com a *Tamed Souls* és una substància desconeguda que cura aquell que la consumeix. El seu funcionament dins del joc és simple: si es pitja el botó configurat, es resta 1 al comptador de càntirs d'*estus* i es suma 5 *hp* a la vida actual del personatge, sense superar la màxima. De la mateixa manera, el botó deixa de funcionar si s'arriba als 0 càntirs.

L'*sprite* que el representa s'ha adaptat d'una imatge penjada a la xarxa social *Twitter* per l'usuari pleiadvm [19].

Finalment es troba l'element que dóna nom al joc, les ànimes. Aquestes també son una variable a l'*script* del jugador, i serveixen com a moneda dins la cripta.

Quan el jugador entri per primer cop a la cripta, ho farà amb la quantitat d'ànimes que hagi recol·lectat a la masmorra. Un cop dins, l'única manera de guanyar-ne és agafant la dels enemics, quan moren.

Per gastar-les, però, hi ha diversos elements repartits pel nivell que requereixen certa quantitat d'ànimes a canvi dels seus serveis. Aquests s'expliquen més detalladament a l'apartat 7.3. A part, quan el PJ mor, pot pagar un nombre concret d'ànimes per evitar-ho, tornant així a l'últim punt de guardat. Si el jugador no es pot permetre aquest preu, mor definitivament: s'esborren les dades de la partida i es tanca el joc.

Per últim, a la cantonada superior dreta es troba un temporitzador, pel temps que es tindrà en compte al rànquing.

7.3 Món

7.3.1 Elements

El món i la posició de tots els seus elements venen determinats per tres *Tilemaps*. El primer és el que s'ha explicat a la introducció i conté els blocs de terra que formen el nivell.

Respecte al segon, cada cel·la pintada representa una punxa. L'objectiu d'aquestes és crear un terra que mati el personatge si aquest cau de les plataformes. Per fer-ho, totes les cel·les del *Tilemap*, a part de tenir la seva caixa de col·lisió, pertanyen al grup "*instakill*". A l'*script* del personatge, quan es calculen els elements amb els quals col·lideix, mor directament si algun pertany al grup.

Per últim hi ha el *Tilemap* Elements. En aquest cas, cada cel·la pintada pot tenir diversos valors. Cada un d'ells

representa algun element que s'ha d'instanciar: el Personatge, un Enemic, una Gàbia, una Foguera o el *Boss*. Tant del primer com de l'últim l'algorisme s'encarrega que n'hi hagi només un.

Les gàbies, com la de la Figura 7, representen elfs empresonats. Quan el personatge s'hi apropa, apareix un text que especifica la quantitat d'ànimes necessària per trencarles. Si el jugador prem el botó indicat i es pot permetre el preu, es resta aquest a les ànimes, es reprodueix l'animació d'alliberar el presoner i s'elimina el text i la possibilitat de pagar. Alliberar presoners redueix la vida inicial del *Boss*.

Fig. 7: PJ al costat d'una gàbia, amb el text corresponent

De manera similar, les fogueres també tenen un text a sobre indicant la seva funció i el preu, però es diferencia del de les gàbies en què aquest es comporta com un menú on el jugador pot escollir entre tres opcions: curar-se, omplir els càntirs d'estus o guardar la partida amb la foguera com a nou punt d'aparició.

Fig. 8: Foguera amb la opció d'establir un punt de guardat

La partida es guarda des d'un *script* global, que conté funcions per desar i carregar informació d'un fitxer. Així, a *Save.dat* s'hi emmagatzema informació sobre el personatge en el moment de guardar: els punts de vida i d'estamina, la quantitat de càntirs d'estus, el nombre d'ànimes i el de presoners alliberats.

Quan es decideix guardar a una foguera, a part de la informació que es desa a *Save.dat*, també s'edita el *Tilemap* Elements. Primer es mou la cel·la que representa al PJ al nou punt de reaparició, just a dalt de la foguera. Després s'eliminen totes les cel·les a l'esquerra d'aquesta. Així, si el PJ mor i reapareix, tots aquells elements no s'instancien innecessàriament.

Per poder mantenir aquests canvis, existeix el fitxer *Tilemaps.dat*. Té un funcionament similar al de guardat, i s'hi emmagatzemen sencers els tres *Tilemaps* que formen el món.

A banda dels *Tilemaps*, existeix un element més al món que és la paret que fa la funció d'*scroll* lateral. Es tracta d'una paret invisible, amb la seva caixa de col·lisió. Es mou alhora que la càmera, sempre a la seva aresta esquerra. D'aquesta manera, bloqueja al PJ (amb l'únic amb el que col·lisiona) d'anar endarrere.

7.3.2 Generació procedural del terreny

Quan s'executa el joc, l'escena del món comença completament buida. Llavors es comprova si el fitxer *Tilemaps.dat* existeix. Si ho fa vol dir que ja hi ha una partida començada, i per tant es copia el seu contingut als *Tilemaps* corresponents. Si no, és una partida nova i s'ha de generar el mapa.

El següent pas per ambdós casos és iterar sobre totes les cel·les pintades del *Tilemap* Elements, i instanciar a cada posició l'objecte corresponent. Això posiciona al mapa tots els elements, inclòs el jugador.

L'últim pas està relacionat amb el fitxer *Save.dat*. Si es tracta d'una partida ja començada, es carrega la informació del fitxer a les variables corresponents del personatge. En canvi, si és una partida nova, aquest fitxer tampoc existeix, pel que es crea amb la informació per defecte i la posició d'aparició del PJ generada per l'algorisme.

La generació del mapa, que s'executa quan es crea una partida nova, es pot dividir en tres: generació del terra, generació del sostre i elements estètics, i posicionament dels elements. El primer es pot llegir al següent pseudocodi:

Algorithm 1 Algorisme Caminador

```
1:  procedure GENERAR_MÓN
2:      posició ← CREARPLATAFORMA(POSINI, MIDAINI)
3:      posicióJugador.x ← POSINI.x + 1
4:      posicióJugador.y ← POSINI.y − 1
5:      posició.x ← posició.x + MINXBUIT
6:      posició ← CREARPLATAFORMA(posició, MINSPAWN)
7:      anterior ← plataforma
8:      terra ← posició.y
9:      while posició.x < MIDAMÓN do
10:         if anterior = buit then
11:             midaPlat ← aleatori entre MINPLAT i MAXPLAT
12:             posició ← CREARPLATAFORMA(posició, midaPlat)
13:             if posició.y = terra then
14:                 terra ← posició.y
15:             anterior ← plataforma
16:         else
17:             midaBuit ← aleatori entre MINXBUIT i MAXXBUIT
18:             posició ← CREARPLATAFORMA(posició, midaBuit)
19:             anterior ← buit
20:     GENERARTERRA(terra)
21:     GENERARSOSTRE( )
22:     GENERARSALABOSS( )
```

Es tracta d'un algorisme "caminador" [20]. Això vol dir que hi ha un element, en aquest cas la variable *posició*, que es mou per la quadrícula generant el terreny al seu pas. Això passa dins de les funcions *crear_plataforma* i *crear_buit*. La segona únicament fa avançar la *posició* l'espai calculat. L'altra, a mesura que avança, pinta una cel·la del *Tilemap* del terra. Com es pot observar al pseudocodi, les crides es van alternant entre plataforma i buit, fins que s'arriba a la mida del món preestablerta. A més, cada crida retorna la posició on s'ha quedat el caminador, i aquesta serveix com a punt de partida per la següent. Perquè el terra sigui visible gairebé permanentment per al jugador, el caminador està restringit també verticalment per uns límits preestablerts.

La successió de crides a *crear_plataforma* abans de començar el bucle serveix perquè les dues primeres pla-

taformes serveixin com a petit tutorial, on el jugador pot aprendre els controls i es trobi amb un primer enemic fàcil de vèncer.

Per últim, s'hi poden veure dues crides, a *generar_terra* i a *generar_sostre*. La primera, com el nom indica, crea una plataforma de la llargada del mapa a l'alçada del terra, indicada per la variable *terra*, que és l'alçada de la plataforma més baixa. Un cop fet això, genera pilars des del terra fins les plataformes. Finalment, omple el terra de punxes allà on no hi ha un pilar. Es pot veure la diferència que marca aquesta funció en el terreny a la Figura 9.

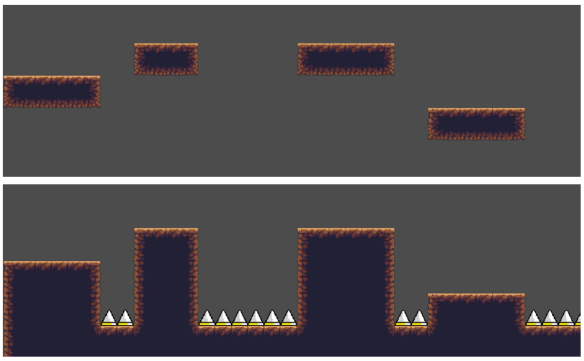


Fig. 9: Nivell abans i després de la generació del terra.

La funció *generar_sostre* crea més terreny sobre el nivell, per crear la sensació de que el personatge està sota terra. Aquesta funció té com a objectiu crear camins el més baixos possible sense augmentar la dificultat dels salts. Per assolir-lo, itera sobre les plataformes. Allà considera les dues opcions s'indiquen a la Figura 10

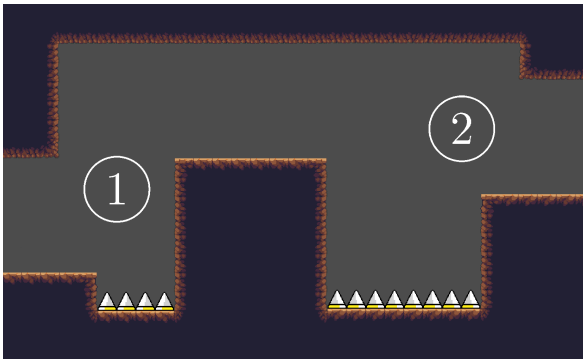


Fig. 10: Generació del sostre amb les dues situacions indicades.

A l'escenari indicat amb un 1 es troba el que es podria anomenar un salt. És a dir, la primera plataforma té una alçada inferior a la segona. En aquest cas, perquè el PJ no xoqui amb el cap quan salti, el sostre es col·loca a partir de l'última cel·la de la primera plataforma a l'alçada determinada per la segona.

Per altra banda, a la situació 2 o caiguda, la primera plataforma es troba més amunt que la segona. Aquí la restricció és que el PJ xoqui amb la paret que baixa del sostre quan aquest canvia d'alçada. Per esquivar-la, es mou la paret una cel·la en direcció al centre de la segona plataforma. Per permetre el salt, durant tot el buit el sostre està a l'alçada de la primera plataforma.

Gràcies a com està implementat, si les dues plataformes estan al mateix nivell el sostre es manté recte, estalviant la necessitat de crear un tercer escenari.

Finalment, *generar_sala_boss* crea un passadís a l'alçada de l'última plataforma, al final del qual es troba la sala del *Boss*. Dins hi pinta la cel·la *Boss* al *Tilemap* Elements.

7.3.3 Distribució dels elements

Un cop generat el terreny, el següent pas de l'algorisme és escollir ón col·locar els elements al *Tilemap* homònim. Primer es divideix el mapa en un nombre pre-determinat de seccions. A les plataformes frontereres s'hi col·loquen fogueres, de manera que queden equilibrats els possibles punts de guardat. Seguidament, es col·loca una gàbia aleatòriament entre totes les plataformes de cada secció, i un enemic de nivell aleatori a cada plataforma que compleixi una llargada mínima. Per últim, pinta el personatge a la posició calculada a *generar_món*.

Per acabar, es crida a la funció *spawn_elements*. *Spawn* és un anglicisme que es tradueix com "fer aparèixer". En aquest cas, el que fa és iterar sobre totes les cel·les pintades del *Tilemap* Elements, en transforma les coordenades i instancia l'escena corresponent segons el valor de cada cel·la.

7.4 Enemics

7.4.1 Visió general

L'enemic es regeix per les mateixes lleis físiques que el personatge, ambdós col·lideixen amb el terra i també entre ells. A més, també tenen associades barres de vida i estamina.

Els seus *sprites* per la versió actual s'han adaptat d'un conjunt adquirit a través del portal web *OpenGameArt* [21].

Pel que fa al moviment, inicialment segueix el que s'anomena una *patrulla*: es mou per la plataforma on està situat en una direcció fins que es troba amb una paret o amb el final de la plataforma, allà canvia de direcció i va fins l'altre extrem de la plataforma, on torna a fer el mateix.

Per la detecció del final de la plataforma, es fa servir un *RayCast2D*. En aquest cas està posicionat davant dels peus, i apunta cap al terra. D'aquesta manera, si el *RayCast2D* no detecta cap col·lisió vol dir que està a punt de caure de la plataforma, i és llavors quan es canvia de direcció. Per la detecció de les parets fa servir un mètode de la classe *KinematicBody2D* (a la que pertanyen tant el jugador com els enemics) anomenada *is_on_wall*. Com el nom indica, determina si està col·lidint amb alguna paret, que es pot definir com qualsevol objecte perpendicular al terra.

Per representar el seu camp de visió, cada enemic té a la seva escena un element *Area2D* en forma de con. Aquest és capaç de detectar si un objecte entra o surt de la seva caixa de col·lisió. Així, si el jugador hi entra, s'alerta l'enemic en qüestió, i canvia del moviment en patrulla a la posició d'atac. En aquesta, es decideix quin mode d'atac és l'òptim, s'hi canvia i s'inicia el moviment corresponent.

Existeixen tres nivells diferents d'enemics, cada cop més forts. El primer pot atacar únicament cos a cos. El segon, a més, té un arc amb el que dispara fletxes. I l'últim té les mateixes capacitats d'atac que el PJ, pel que també llança boles de foc.

A més, el de nivell 3 compta amb un escut bombolla que el permet cobrir-se. Si detecta que un projectil del PJ entra al seu camp de visió i té prou estamina, l'activa. Això el fa

invencible un període curt de temps, durant el qual tampoc pot atacar. Els seus sprites s'han adaptat d'una animació anònima al web *Pixilart* [22].

Fig. 11: Enemic de nivell 3 amb l'escut bombolla activat.

Aquí els atacs funcionen de manera similar a l'atac *melee* del PJ. L'enemic també alterna entre els diversos modes, i a la seva escena escena hi ha un *RayCast2D* per a cada un, de mides diferents. Així, si es detecta que el PJ col·lideix amb el *RayCast2D* corresponent, es llança l'atac.

7.4.2 Intel·ligència artificial

Quan el camp de visió d'un enemic detecta el PJ, es pot considerar que s'activa la seva intel·ligència artificial. Es pot dividir en dues grans parts, que son independents entre elles. Aquestes son la tria del mode d'atac segons l'entorn i els moviments passats del jugador, i el moviment a fer un cop triat.

En futures versions, la tria del mode d'atac serà un algorisme complex que tindrà en compte el comportament previ del jugador i les posicions relatives per calcular el mode òptim. Actualment, canvia al que guanya directament al mode actual del jugador a la jerarquia pedra-paper-tisores.

Es segueix un patró de moviment determinat segons el mode d'atac escollit.

El moviment del mode *melee* té com a objectiu apropar-se al personatge, per poder-li fer un atac cos a cos. Per això la direcció en la que es mou l'enemic en aquest mode és la direcció en la que es troba el personatge. Aquesta es troba fàcilment determinant si la resta entre la posició a l'eix horitzontal dels dos elements és positiva o negativa. L'únic cas especial a tenir en compte és el final de la plataforma. Allà l'enemic es queda quiet esperant al jugador, per evitar saltar de la plataforma i morir a les punxes.

A cada cicle d'execució, també es comprova si el *Melee-RayCast* detecta al jugador. Si ho fa, ataca.

Per als atacs *ranged* i *magic* la lògica és una mica més complicada. Per aquests, l'enemic es posiciona a la dreta del tot de la seva plataforma, s'estableix allà i dispara al personatge quan està a l'abast. Ho fa així ja que la càmera força al jugador a moure's cap a la dreta constantment, per l'*Scroll lateral*. Així s'evita que els enemics que ataquen a distància morin massa ràpid o ataquin al jugador des de fora del camp de visió d'aquest.

Si encara no està posicionat, es mou incondicionalment cap a la dreta fins que arriba a l'extrem de la plataforma, i allà es marca com a posicionat. En canvi, si ja està posicionat l'únic que es fa és canviar la direcció (sense moure), per apuntar sempre al personatge.

Si està posicionat, compta també amb un sistema per evitar que l'empenta dels projectils facin caure l'enemic directament a les punxes. Si hauria d'estar posicionat i detecta que no està en contacte amb el terra, intenta salvar-se movent-se cap a l'esquerra per contrarestar l'empenta.

De nou, si els respectius *RayCasts* detecten al jugador, l'enemic ataca.

7.5 Boss

7.5.1 Visió general

El *Boss* és l'enemic més fort de tota la cripta, i es troba al final d'aquesta protegint el segell que dóna poder a Seoh.

Visualment és un ós, com els enemics però el doble de gran i amb una armadura daurada. Té també una barra de vida, que representa força més punts.

El seu comportament passa per tres etapes, úniques per aquest enemic. Tenen cadascuna el seu patró, tant pel moviment com per l'atac. Son, per ordre, l'etapa de Salt, la de Llançament de *Minions* i la d'Invocació de Llamps. El *Boss* es mou entre elles en funció dels seus punts de vida; Cada terç va assignat a una etapa.

L'etapa inicial és la de Salt, la més senzilla de totes, i l'única que no utilitza màgia. En aquesta l'objectiu del *Boss* és aixafar el Personatge saltant-hi a sobre. Després d'un segon al terra, calcula la trajectòria per caure sobre el seu cap, i l'executa. Si col·lisiona amb ell durant la caiguda, li fa mal. Per al PJ, és rebut com un atac *Melee* més fort i amb més empenta.

Si el jugador aconsegueix reduir la vida del *Boss* a $\frac{2}{3}$ de la inicial, aquest canvia a l'estat de Llançament de *Minions*. La paraula *Minion* prové de l'anglès i significa esbirro. A *Tamed Souls*, es tracta d'enemics més dèbils creats amb màgia. En aquest estat, el *Boss* té la capacitat d'engendrar fins a 7 d'aquestes criatures, que van directament a atacar al Jugador.

Fig. 12: *Boss* llançant un *Minion* al Personatge.

Finalment, a l'últim terç de la vida el *Boss* passa a l'estat d'Invocació de Llamps. En aquest, es col·loca a l'esquerra de la sala i passa a controlar una diana màgica que apareix al terra. Cada tres segons, és l'objectiu de la caiguda d'un llamp. Si hi ha qualsevol ésser a sobre d'ella en el moment de l'impacte, rep un atac de tipus màgic.

Als dos últims modes, a part dels nous comportaments, si el Personatge s'apropa massa al *Boss*, aquest li farà un atac cos a cos per allunyar-lo.

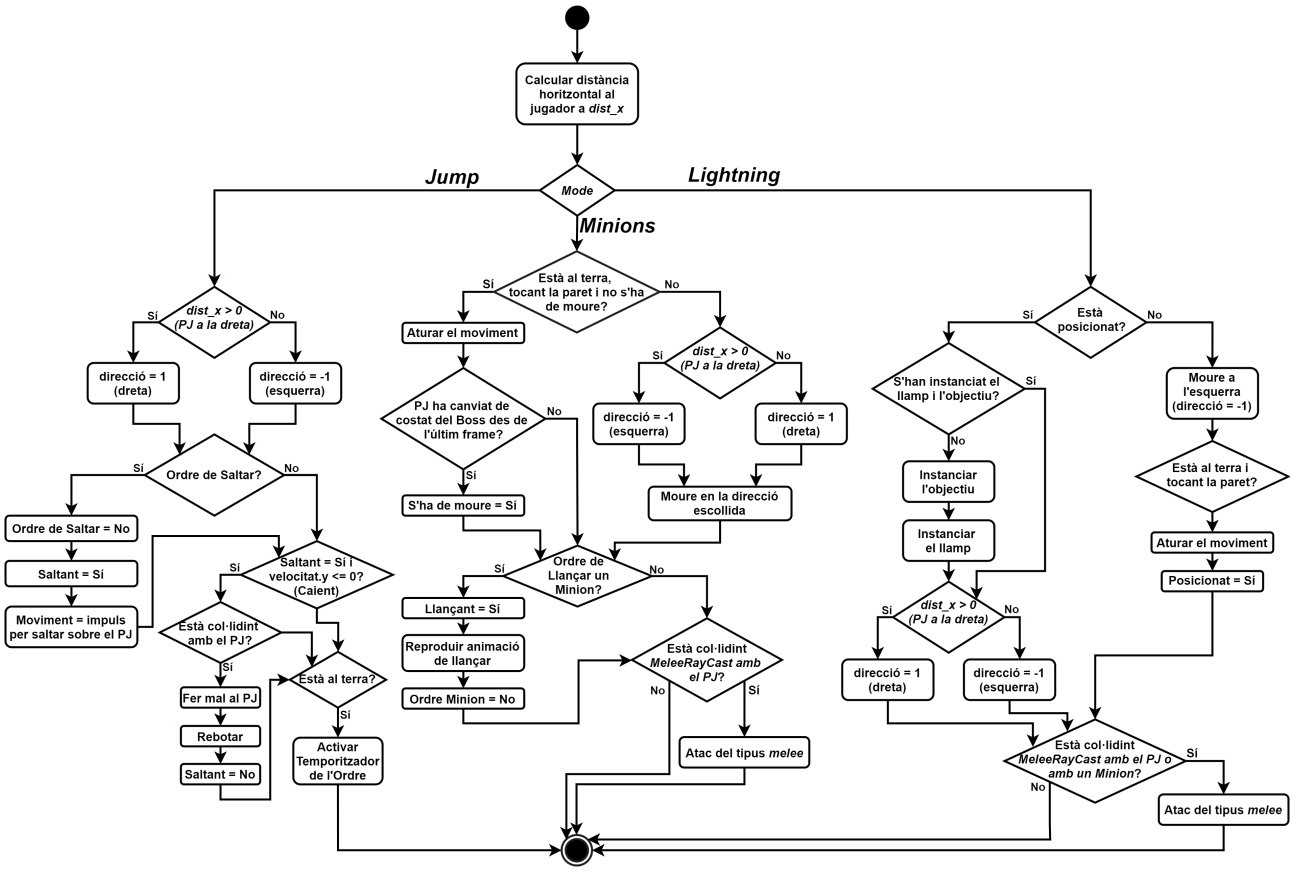


Fig. 13: Diagrama de flux de la Intel·ligència Artificial del *Boss*

7.5.2 Intel·ligència artificial

El diagrama de flux de la Figura 13 representa gràficament com funciona l'algorisme de moviment i atac dels diversos modes del *Boss*. En aquest apartat s'explicarà i es donaran més detalls del seu funcionament.

Per a tots els estats el primer que es fa és calcular la distància fins al Personatge, ja sigui per saber en quina direcció es troba com per calcular la trajectòria fins a ell.

Pel mode de Salt, es fan servir ambdues. Primer, canvia de direcció per mirar sempre al PJ. Després, si té l'ordre corresponent, calcula la trajectòria fins al seu cap, i s'aplica la força necessària per arribar-hi. La fórmula de l'impuls és la següent, sent $dist_x$ la distància entre el PJ i el *Boss*:

$$\overrightarrow{moviment} = \langle dist_x \times 0.75, -400 \rangle$$

Pel que fa a l'ordre, és negativa per defecte. El *Boss*, però, compta amb un temporitzador que activa l'Ordre de saltar quan arriba a 1 segon. Aquest s'inicia quan detecta que ha arribat al terra després de saltar.

Abans d'això, si mentre està caient (si té una velocitat en l'eix *Y* positiva i està saltant) col·lisiona amb el PJ, es crida a la seva funció *hit*, indicant-hi que és un atac *melee* amb un dany superior a l'habitual.

Pel que fa al llançament de *Minions*, el primer que es fa és comprovar si el *Boss* està correctament posicionat, és a dir, si toca de peus a terra i d'esquena a una paret. Ho detecta a través de dos *Raycasts2D* que apunten, respectivament, a sota i a darrere del cos. Si està ben posicionat, deixa de moure's a no ser que el PJ canvii a l'altre costat del *Boss*, cosa que faria que es desplacés fins la paret contrària.

Per altra banda, si no ho està es mou fugint del Personatge fins que topa amb una paret. Cal destacar que camina endarrere per poder llançar esbirros mentre fuig.

Llavors, si rep l'Ordre de llançar un *Minion*, activa l'animació del llançament. El reproductor d'animacions és l'encarregat d'instanciar el petit enemic en aquest cas. Ho fa en una posició designada per un element *Position2D*, com als atacs del PJ. La diferència és que als *Minions* se'ls dóna una empenta inicial per llançar-los cap al Personatge. La seva trajectòria segueix la mateixa formula que el Salt, però amb menys alçada. Això fa que no caigui directament a sobre del PJ i que tingui una paràbola menys pronunciada.

Per últim es troba el mode d'Invocació de Llamps. Aquest també comença comprovant si està a posició. La diferència en aquest cas és que un cop a lloc ja no s'hi mou, de manera similar als estats *Ranged* i *Magic* dels enemics.

Com es pot veure al diagrama, en aquest estat la càrrega algorísmica més gran no la porta el *Boss* sinó que ho fa l'objectiu. Des de l'enemic final únicament s'instancia, juntament amb el llamp, quan s'ha posicionat.

Així doncs, l'objectiu en forma de diana quan apareix a l'escena fa dues coses. La primera és col·locar-se, al mateix nivell que el personatge a l'eix *X*, i al del terra a l'eix *Y*. L'altra és iniciar l'animació. Aquesta comença amb l'opacitat al 0%, i puja progressivament fins a ser completament opac. Llavors, fa pampallugues durant gairebé un segon per alertar al jugador.

Juntament amb la diana es mou un element *Area2D*. Quan finalitza l'animació, si el PJ hi està dins rep un atac màgic. A més, es genera un llamp que comença a la cantonada superior esquerra de la sala i impacta aproximadament

a la posició de la diana. Pel que fa al moviment, la diana té una velocitat fixa i persegueix constantment al Personatge, per intentar que estigui dins l'*Area2D* quan caigui el llamp.

El llamp és únicament un efecte visual i el codi que el genera s'ha adaptat d'un projecte de l'usuari de *YouTube* Gingerageous Games [23].

8 Resultats

En general, el projecte ha estat un èxit. S'ha construït un joc divertit però desafiant, i que incorpora competències de la menció gràcies a la generació procedural de nivells i a les intel·ligències artificials dels enemics i del *Boss*. Un vídeo de demostració dels assoliments es pot trobar al següent enllaç: https://youtu.be/BScunYbMC2M.

9 Conclusions

S'han complert tots els objectius satisfactòriament: S'ha desenvolupat en la seva totalitat la idea inicial del projecte, fins i tot amb més funcionalitats. També s'ha creat un joc jugable, divertit i amb èmfasi als elements algorísmics. Els objectius específics també s'han complert, amb la generació procedural del nivell, la intel·ligència artificial dels enemics i la del *Boss*, i fent de *Tamed Souls* un repte. Com s'ha comentat durant l'article, en versions futures la intel·ligència artificial dels enemics podria tenir en compte el comportament anterior del jugador. També es podria millorar l'estètica i afegir més nivells, un per a cada raça.

Agraïments

Al tutor d'aquest Treball de Final de Grau, Jorge Bernal, per la motivació aportada, les ganes de tirar el projecte endavant i l'entusiasme transmès a cada reunió.

Referències

[1] Definición de Plataformas [en línia]. (4 d'Abril, 2013). Recuperat de http://www.gamerdic.es/termino/plataformas

[2] Definición de Soulslike [en línia]. (14 d'Octubre, 2019). Recuperat de http://www.gamerdic.es/termino/soulslike

[3] Dead Cells - Rogue-lite metroidvania with some souls-lite combat on top! Kill, die, learn, repeat [en línia]. (s.d.). Recuperat de https://dead-cells.com

[4] Blasphemous Home - The Game Kitchen [en línia]. (s.d.). Recuperat de https://thegamekitchen.com/blasphemous/

[5] Mega Man [en línia]. (s.d.). Recuperat de https://megaman.capcom.com

[6] P. Spronk, M. Ponsen, I. Sprinkhuizen-Kuyper, E. Postma. Adaptive game AI with dynamic scripting. (9 de Març, 2006). Recuperat de https://link.springer.com/article/10.1007/s10994-006-6205-6

[7] S. C.J. Bakkes, P. H.M. Spronck, H. J. van den Herik. Opponent modelling for case-based adaptive game AI. (10 de Setembre, 2009). Recuperat de https://www.spronck.net/pubs/bakkes_journalOM.pdf

[8] Godot Engine - Free and open source 2D and 3D game engine [en línia]. Recuperat de https://godotengine.org

[9] Crea juegos exitosos desde el concepto hasta la comercialización | Unity [en línia]. (s.d.). Recuperat de https://unity.com/es/solutions/game

[10] The most powerful real-time 3d creation platform - Unreal Engine [en línia]. (s.d.). Recuperat de https://www.unrealengine.com/en-US/

[11] Godot Engine – Download | Windows [en línia]. (s.d.). Recuperat de https://godotengine.org/download/windows

[12] Godot Engine on Steam [en línia]. (s.d.). Recuperat de https://store.steampowered.com/app/404790/Godot_Engine

[13] Manifiesto por el Desarrollo Ágil de Software [en línia]. (2001). Recuperat de http://agilemanifesto.org/iso/es/manifesto.html

[14] Metodologías ágiles. Las 3 más usadas actualmente. (6 de Març, 2019). Recuperat de https://blog.conectart.com/metodologias-agiles/

[15] Trello [en línia]. (s.d.). Recuperat de https://trello.com

[16] UmaiPixel. Godot 3 – Platformer Tutorial Series. [llista de vídeos] (14 d'Abril, 2019). Recuperat de https://www.youtube.com/playlist?list=PLyckz_-Rzq6ClGevL2fneJ5YJnMPKWa4M

[17] HeartBeast. Godot 3 2d Platform Game. [llista de vídeos] (13 d'Abril, 2018). Recuperat de https://www.youtube.com/playlist?list=PL9FzW-m48fn2jlBu_0DRh7PvAt-GULEmd

[18] Control the game's UI with code – Godot Engine latest documentation [en línia]. (s.d.). Recuperat de https://godot-es-docs.readthedocs.io/en/latest/getting_started/step_by_step/ui_code_a_life_bar.html

[19] pleiadvm. Estus flask. (1 de Gener, 2020). Recuperat de https://tinyurl.com/56pknbc5

[20] T. Batcher. Procedural Level Generation Algorithms. (29 d'Agost, 2018). Recuperat de http://jthomasbacher.com/bacherJuniorISWriting.pdf

[21] doudoulolita. Bearsum, pixel art bear. (24 de Gener, 2013). Recuperat de https://tinyurl.com/1fry39zy

[22] Anònim. Shield Bubble. (2018). Recuperat de https://tinyurl.com/1t4mmz6f

[23] Gingerageous Games. Lightning Tutorial (Godot 3.2) Isometric Tower Defense [8] [vídeo]. (11 d'Abril, 2020). Recuperat de https://www.youtube.com/watch?v=QGReh8eF69w