UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH
Facultat d'Informàtica de Barcelona

FIB

Jocs per Computador

JC-MEI

June 2, 2025

Carles Matoses Gimenez *
Carles Matoses Gimenez †

*Email: carles.matoses@estudiantat@upc.edu
†Email: carles.matoses@estudiantat@upc.edu

Contents

1 Introducció

2 Objectius

2.1 Mapa

El jugador ha de conseguir claus, objectes equipables i resoldre trenca closques per a poder accedir a l'ultima pantalla on enfrontar-se al ultim enemic. El recorregut del joc es mostra a la figura 1. El jugador comença en el "overworld" i ha d'accedir a la "dungeon1". Les posibles rutes del personatge estan marcades en una linea rossa. Alguns nivells presenten linees de color groc, taronja i blau que indiquen una dependencia per a poder avançar.

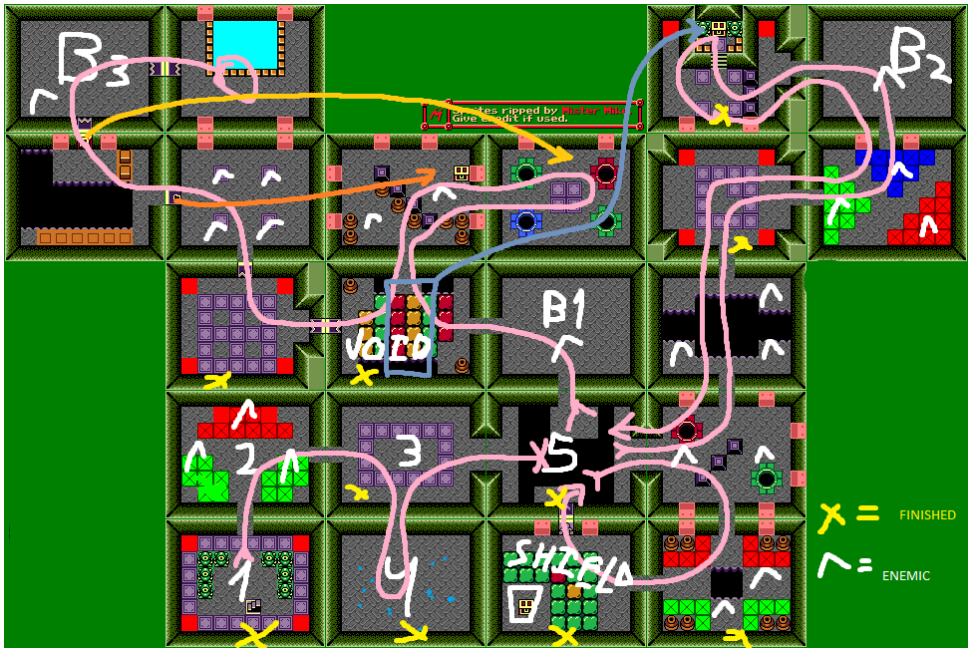

Figure 1: Mapa del joc amb el recorregut i els requerriments per a cada nivell.

3 Disseny

Per al disseny del joc no hem seguit un patró de disseny concret. El "loop" del joc incorpora una clase auxiliar que anomenada **gamestatemanager**. Aquesta conté una llista amb els estats necessaris. L'objectiu de gastar aquesta classe és poder anyadir i eliminar estats dinamicament. Per exemple, qual el jugador apreta "escape" i s'obre el menú, el que realment s'està fent és concatenar dos estats i sols permitir que el més alt en la jerarquia puga llegir els inputs i actualitzar-se en funció del temps.

Cada estat es responsable del seu propi update, draw i inputs. Estan dissenyats de

manera que poden ser creats i destruits sense afetar als elements globals del joc com els nivells o el jugador. El jugador es persistent encara que es surti del nivell (escape -¿ Exit -¿ Start) per aquesta mateixa raó.

3.1 Detecció de colisions

Els elements poseeixen una "bounding box". L'escena itera sobre els objectes i comprova si hi ha solapament. En cas de solapament, el moviment es revertit. Quan hi ha col·lisió s'executa onCollide(item) amb una referencia del objecte colisionat. Aquest sistema afecta tant al Player com als enemics i els projectils. Adicionalment hem anyadit altres tipus de colisions com el "SteptOn" que permiteix saber sobre que element es troba el jugador.

Els atacs del jugador generen una bounding box enfront d'ell i es comproba si colisiona amb algun enemic o element interactiu. Si es detecta una collisió, s'execute "onCollided" de cada objecte.

3.2 Nivells, elements i entitats

La classe `World` guarda un diccionari amb els mapes. La classe `Mapa` conté una llista amb els nivells. La classe `Level` contente una llista amb els elements i entitats que s'han de carregar. Per al projecte s'ha creat un World que conte dos mapes, "dungeon1" i "overworld". El mapa "dungeon1" conté 29 nivells. Cada nivell es responsable de contindre els estats per defecte de les entitats i també de customitzar events com "entrar" i "sortir" de les entitats.

Scene es encarregada d'actualitzar els elements en cada fotograma. En primer lloc, Scene demana a World el contingut del nivell en el que es troba el jugador. A continuació, guarda una copia local d'aquests elements i permiteix al usuari interactuar amb ells. Tots els canvis que es realizen en els elements locals no es traslladen directament a la referencia global, sinó que s'actualitzen en funció de callbacks especials o per events externs. La naturaletza del joc requereix que els nivells tornen al seu estat original quan el jugador surt d'ells. Per això, implementar aquest comportament com el funcionament per defecte pareixia una idea acceptable. Aquesta estrategia comporta els seus inconvenients, com tindre que crear moltes funcions auxiliars per a comprtaments especifics o tindre que crear copies que influencien en la memoria del joc.

3.2.1 Categories d'elements i entitats

A continuació es mostren les diferents categories d'elements i entitats presents al joc, juntament amb exemples de cada tipus:

- **Items equipables**

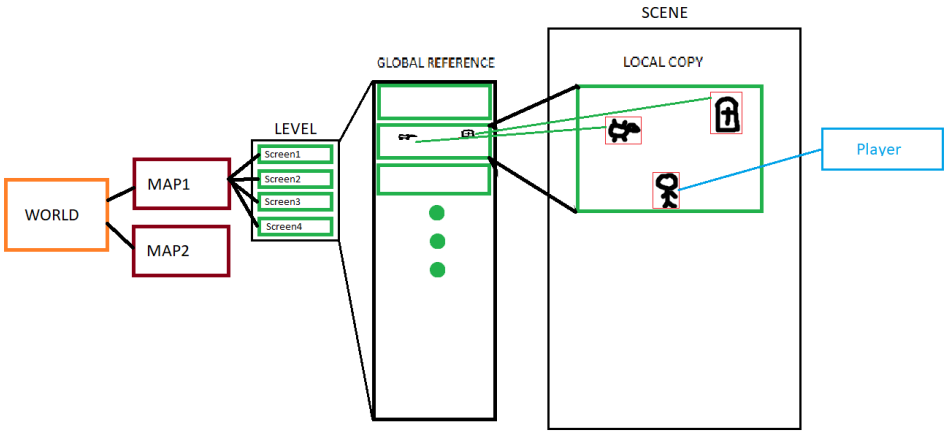

Figure 2: Jerarquia dels elements durant l'execució del joc.

  - Espasa
  - Escut
  - Ploma (Feather)
  - Braçalet (Bracelet)
- **Items**
  - Claus
- **Blocs**
  - Blocks invisibles.
  - Portcullis: portes que s'obrin i tanquen per scripts.
  - Lights: block decoratiu animat.
  - Vase: block decoratiu animat.
  - Fire Place: block decoratiu animat.
  - Animated Floor: block decoratiu animat.
- **Blocs interactius**
  - Sòls flotants (també representen el buit): es trenquen al passar per damunt.
  - Làpides: es poden empentar.
  - Cofres: es poden obrir i contenen objectes.
  - Rotors: trenca closques que es poden resoldre al colpejar-los.
  - Portes: es poden obrir amb claus.
  - Portes en el pis: Teletransporten al jugador a un altre nivell.
  - Estatua: Activa un dialeg.
  - Floating Heart: Consumible que incrementa la vida del jugador.
  - Floating Money: Consumible que incrementa la quantitat de diners del jugador.
- **Entitats**
  - Jugador
  - Enemics comuns (slimes, ratpenats, etc.)
  - Bosses
  - NPCs

3.3 Mecàniques

3.3.1 Equipar Items

Els items poseeixen atributs o effectes necessaris per a resoldre trenca closques o per a poder accedir a nivells. El objectes s'equipen mitjançant l'inventari.

3.3.2 Atacar

Atacar es un altra mecanica del joc. El jugador pot atacar als enemics i a alguns objectes.

3.3.3 Interactuar

Alguns objectes com portes o estatues poden ser interactuats amb la tecla principal. Els objectes responen a l'interacció executant un callback.

3.3.4 Tocar

Gracies a la detecció de colisions, podem ejecutar callbacks en funció de quin objecte esta colisionant amb el jugador. Per exemple, trencar el pis o empentar un objecte.

3.4 Triggers

Els levels poden contenir els següents triggers:

- **onEnter**: s'executa quan el jugador entra al nivell. Permiteix anyadir enemics a l'escena, moure al personatge amb un script, o qualsevol altre acció que es vulga realitzar quan el jugador entra al nivell. Adicionalment permet saber si es la primera vegada que el jugador entra al nivell.
- **onExit**: s'executa quan el jugador surt del nivell. S'utilitza sobretot per a moure el personatge a una posició concreta.
- **onAllEnemiesDead**: s'executa quan tots els enemics del nivell estan morts. Per defecte, es desbloqueja la porta de sortida del nivell.

Amb aquests triggers podem fer apareixer un "boss" una unica vegada

Per a més control, també podem anyadir triggers personalitzats tant als objectes com als nivells. Normalment son utilitzats per a modificar l'estat d'un objecte permanentment.

- **Tombstone**: La làpida de la cova de la mort té un trigger que es dispara quan el jugador la mou. Aquest trigger posiciona la làpida en una posició concreta de forma permanent.

4 Trenca closques

4.1 Lights Out

Aquest trenca closques es basa en un joc de taula que consisteix a apagar totes les llums d'una graella. Cada vegada que es prem una llum, aquesta i les seves veïnes canvien d'estat (encenent-se si estaven apagades i apagant-se si estaven encenent-se). L'objectiu és trobar la combinació correcta de premudes per apagar totes les llums.

Per a poder generar diferents trenca closques, s'ha creat una classe "Rotor" que contindrà: una id unica per nivell, una llista dels rotors als que afectarà (ids) i l'estat amb el que comença.

Figure 3: Exemple visual d'un trenca-closques "Lights Out" implementat al joc.

4.2 Objectes Pesats

Durant el joc, l'usuari es trobarà amb objectes que podrà empentar en algunes direccions. Aquest es el cas de les làpides i algunes pedres. Per a poder implementar aquest comportament, s'ha creat un item equipable (al igual que en el joc original) que incrementa les estadístiques de força del jugador. Si el jugador intenta desplaçar-se en una direcció constant i l'objecte obsstruint el pas es pot moure, aquest event es detectat per la classe `Player` i es realitza un moviment.

Aques trenca closuqes bloqueja pantalles necessaries per a poder avançar en el joc:

- Làpida de la cova de la mort: per a poder accedir a la cova de la mort, el jugador ha de moure una làpida que bloqueja l'entrada. Aquesta làpida es pot moure cap a

dalt, baix i esquerra. En la mateixa cella on es trobava la làpida, hi ha una porta cap a la "world.mapa['dungeon1']"
- Pedres pesades la vencer a "B2": hi ha que moureles per a poder accedir al cofre que conté la "Feather", objecte necessari per a volar.

4.3 Preguntes i Respostes

Algunes estatues poden requrir d'una resposta per part del jugador. La resposta activarà un trigger.

5 Enemics

6 Menus

Per a poder començar o eixir el joc, s'ha creat un conjunt de pantalles que permeten al jugador navegar per les opcions del joc. Aquestes pantalles es tracten com a estats del joc i es poden accedir mitjançant la tecla "escape" o alguns events.

- Start: Pantalla inicial del joc. Permet iniciar una partida o tancar el joc.
- Menu In Game: Pantalla que permet al jugador accedir als controls, reanudar o eixir del joc.
- Controls: Pantalla que mostra els controls del joc.
- Credits: Pantalla que mostra els crèdits del joc.
- Menu Final: Igual que el menú principal però amb una transició per a donar a entendre al jugador que ha guanyat.
- Dialegs: Encara que no són un "menu" en el sentit estricte, s'ha implementat un sistema de diàlegs que permet al jugador interactuar amb els NPCs i seleccionar diferent opcions.

7 No Planejat

La mort del jugador no esta implementada de forma que es considere part del loop jugable. Si la vida del personatge baixa al maxim, simplement apareix en alguna pantalla amb tot igual que abanç. Alguns triggers poden tenir problemes amb aquesta implementació. Si el jugador mor mentres lluita contra un "boss", la seguent vegada que entre pensarà que no es la primera vegada. La solució a aquest problema es poc elegant, però es soluciona fent varies comprobacions de si el boss ha sigut derrotat o no.

8 Examples

Code examples:

8.1 Text

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Donec sit amet arcu aliquam, mattis sapien quis, faucibus ligula. Praesent facilisis felis libero, nec scelerisque est vestibulum a. Proin gravida tempor neque eget maximus. Aliquam vel ex arcu. Maecenas porta blandit leo, eget dapibus libero faucibus quis. Curabitur vel sapien pharetra, ultricies nulla sed, ullamcorper metus. Sed mauris sapien, dapibus et elit in, bibendum pellentesque dui. Donec consequat quam ut pulvinar sodales. Duis commodo ex pharetra mauris dictum, in ultricies purus facilisis. Donec id ornare sem. Nam diam erat, imperdiet et orci id, lacinia condimentum justo. Integer laoreet pulvinar nibh. Vivamus vel tellus lacus.

- one
- one
- one
- one
- one

Phasellus congue, massa in aliquet elementum, ipsum felis dictum enim, id maximus ante orci sit amet dui. Suspendisse blandit iaculis feugiat. Curabitur ante elit, tristique luctus hendrerit sed, posuere quis massa. Vivamus eu nunc vel neque egestas porttitor ac at nibh. Morbi odio justo, maximus sit amet elementum nec, finibus sed purus. Mauris eu posuere neque, at consequat elit. Praesent eu venenatis mauris, vel lobortis eros. Donec suscipit congue augue, ut aliquet metus volutpat ac. Vivamus mattis, nisl vel placerat euismod, tellus velit sollicitudin massa, nec sagittis nunc elit vel mauris. In id justo non nunc condimentum bibendum. Mauris scelerisque urna nisi, et lacinia justo posuere eget.

8.2 Code

```
#define NB 2 // selected based on the number of threads
...
double relax_jacobi (double *u, double *utmp, unsigned sizex, unsigned sizey)
{
...
    #pragma omp parallel for reduction(+:sum) private(diff)
    for (int ii=0; ii<nbx; ii++) {
        ...
    }
```

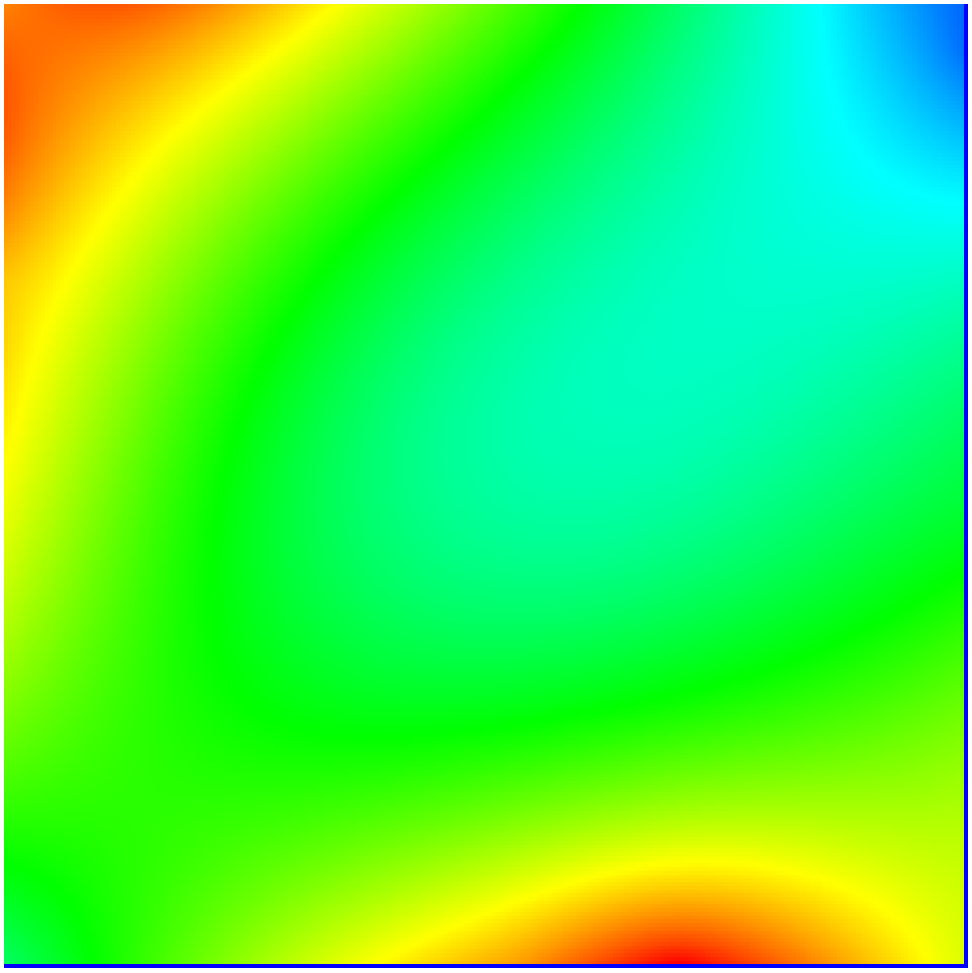

Figure 4: Test results for circuit 1

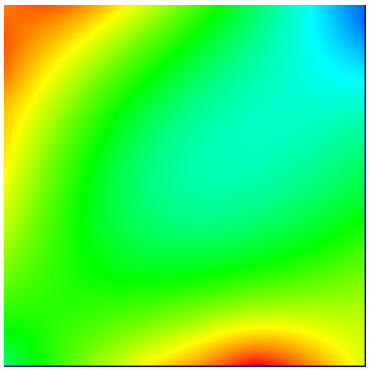

Figure 5: subimage1

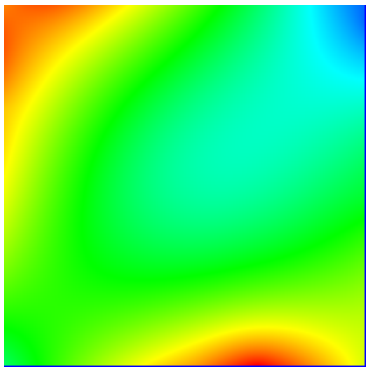

Figure 6: subimage2

```
    return sum;
}
```

References