Isaac - Jeu Roguelike 2D d'action-aventure en Java
Projet de programmation objet

Theo Losekoot
Josie Signe

2021

1 Introduction

The Binding of Isaac est un jeu d'action-aventure rogue-like en 2D en vue de dessus. L'objectif de ce projet est de concevoir et implémenter une version simplifiée de ce jeu.

Le personnage principal de ce jeu, lui donnant son nom, est Isaac. L'aventure démarre lorsque Isaac, pour fuir sa mère, s'engouffre dans un donjon. Ce donjon est composé de plusieurs salles dans lesquelles il doit essayer de survivre face à divers monstres et pièges. Pour se défendre, le personnage tire des larmes qui lui servent de projectiles pour blesser les monstres. En débarrassant une salle de ses monstres, Isaac peut obtenir certains objets, qui pourront être consommés ou gardés par le personnage. Le monde est persistant : les monstres ne réapparaissent jamais une fois battus et les objets laissés dans une salle y sont encore quand Isaac y revient.

Le jeu se termine lorsque le joueur a vaincu le boss du donjon ou que son personnage a perdu tous ses points de vie.

Figure 1: Une salle du jeu, avec Isaac se battant contre des araignées

2 Règles détaillées

2.1 Personnage

Le joueur contrôlera le personnage d'Isaac.

- *Déplacement:* À l'aide des touches 'z', 'q', 's' et 'd', le joueur peut déplacer Isaac dans 8 directions: haut, bas, gauche, droite, et en diagonale, dans le cas de l'appui simultané de deux touches.
- *Tir:* À l'aide des touches directionnelles, le joueur peut faire tirer des larmes à Isaac dans 4 directions: haut, bas, gauche et droite.
- *Caractéristiques:* Isaac est caractérisé, entre autres, par
  - sa vitesse de déplacement ;
  - ses points de vie maximum ;
  - les dégâts qu'il inflige par larme ;
  - la portée de tir de ses larmes ;
  - la vitesse de tir de ses larmes.
- *Inventaire:* Isaac possède une certaine quantité de pièces d'or et d'équipements lui donnant un bonus passif et permanent (augmentation de caractéristiques), qu'il pourra ramasser au sol lorsque le jeu sera suffisamment clément pour lui en donner.

2.2 Donjon

Isaac se déplacera dans un donjon.

- Un donjon est un ensemble de salles reliées entre elles par des portes. Une salle peut avoir jusqu'à 4 portes qui mènent à d'autres salles : une en haut, une en bas, une à gauche et une à droite. Une porte ne doit apparaître, d'abord fermée puis ouverte, que lorsqu'il y a effectivement une salle à rejoindre derrière cette porte.

  Tout le donjon n'est pas affiché en même temps. N'apparaîtra sur l'écran qu'une salle à la fois: celle dans laquelle se situe Isaac.
- Dans cette version simplifiée du jeu, 4 types de salles sont a représenter :
  - 1 seule salle de spawn, dans laquelle rien ne se passe. C'est là qu'Isaac commence son aventure.
  - Au moins 2 salles de monstres, dans laquelle des monstres apparaissent lorsqu'Isaac y entre. Les portes sortant de cette salle sont bloquées jusqu'à ce qu'Isaac se soit débarrassé de tous les monstres présents, après quoi les portes s'ouvrent. En même temps que la réouverture des portes, un objet peut apparaître à un emplacement libre dans la salle, et Isaac peut le ramasser.
  - Une seule salle magasin, dans laquelle des objets sont disposés au sol accompagnés de leur prix en pièces d'or. Tous les objets du jeu ne sont pas disponibles au magasin : il y aura au plus 3 objets à vendre dans le magasin parmi tous les objets possibles du jeu. Isaac peut marcher sur un objet pour l'acheter. L'objet est ramassé si Isaac possède un nombre de pièces suffisant pour l'acheter. Le magasin ne se renouvelle pas : les objets achetés ne sont pas remplacés par d'autres objets lorsqu'ils sont achetés.

  - Une seule salle de boss, dans laquelle est présent un type de monstre particulier tel que présenté dans la section suivante. Le jeu se termine une fois le boss vaincu.

  Vous êtes libres de placer ces salles comme vous le souhaitez en respectant les contraintes données ci-dessus. Dans la version de base de ce jeu, vous pouvez commencer par fixer un plan de salle. Vous pourrez par la suite améliorer votre jeu avec une génération de donjon aléatoire.

- Une salle peut contenir des obstacles comme des rochers ou des piques. Les rochers bloquent le passage d'Isaac et des monstres, tandis que les piques infligent des dégâts à Isaac lorsqu'il marche dessus. Les monstres y sont immunisés. Attention, les obstacles ne doivent pas empêcher l'accès à une porte.

2.3 Monstres

Chaque monstre a les caractéristiques suivantes :

- sa vitesse de déplacement ;
- ses points de vie ;
- les dégâts qu'il inflige au corps à corps ;
- les dégâts, portée et vitesse de tir de ses projectiles (s'il tire) ;

Dans ce projet, 3 monstres sont à représenter :

- *Spider:* Une araignée est un monstre qui se déplace rapidement et aléatoirement. Elle reste immobile durant une certaine durée puis se déplace dans une direction aléatoire, avant de s'arrêter à nouveau et de recommencer son cycle. Elle inflige des dégâts lorsqu'elle touche Isaac.
- *Fly:* Une mouche est un monstre volant, donc pouvant se déplacer par dessus les obstacles, s'approchant doucement d'Isaac tout en lui tirant des projectiles. Ce monstre peut donc infliger des dégâts par contact physique ainsi que par projectile.
- *Boss:* Le Boss est un monstre imposant, qui va choisir parmi les comportements des *Spider* et des *Fly* selon ses envies. Il possède de meilleures caractéristiques que ceux-ci.

2.4 Objets

Il y aura parfois des objets sur le sol des salles. Un objet peut être ramassé par Isaac en marchant dessus ou laissé à son emplacement pour être éventuellement ramassé plus tard. Nous voulons représenter 2 types d'objets :

- *Les consommables :* Ce sont des objets qui ont un effet instantané lorsqu'il sont ramassés. Il y en a de deux types :
  - Les pièces : Elles permettent à Isaac d'acheter des objets dans une salle magasin. Ramasser une pièce augmente le solde d'Isaac si ce dernier n'a pas encore atteint le solde maximal. Lorsqu'un objet est acheté au magasin, le nombre de pièce de Isaac diminue de son coût d'achat. Il existe plusieurs types de pièces ayant des valeurs différentes, par exemple 1, 5 et 10. Une pièce, comme tout objet, a des chances d'apparaître au sol en terminant une salle de monstre.

  - Les cœurs : Ils permettent à Isaac de se soigner. Lorsque Isaac marche dessus, un coeur est ramassé uniquement s'il manque des points de vie à Isaac. Il rend alors à Isaac un certain nombre de point de vie, 1 pour un demi cœur et 2 pour un cœur complet, sans dépasser son nombre maximal de points de vie. Un cœur a également des chances d'apparaître au sol en terminant une salle de monstre. Il peut aussi être possible d'acheter des cœurs au magasin.

- *Passifs :* Les objets passifs de notre jeu permettent de modifier les caractéristiques d'Isaac. Bien souvent, ils permettent de les augmenter, mais il est parfois possible qu'ils les diminuent en échange d'une importante amélioration par exemple.
  Les objets passifs ont une chance d'apparaître au sol en terminant une salle de monstre, mais elle est beaucoup plus faible que celle des consommables. Lorsque Isaac marche sur un objet passif au sol, il est aussitôt ramassé. Il est aussi possible d'acheter des objets passifs au magasin pour 15 pièces. Dans ce cas, l'objet n'est ramassé que si Isaac a suffisamment de pièces pour l'acheter. Il doit y avoir au moins 1 objet passif achetable dans le magasin de votre jeu.
  Une fois ramassé, les objets ne peuvent être enlevés. Ils appliquent donc immédiatement un effet passif qui modifiera les caractéristiques de Isaac de façon permanente.
  Nous vous demandons de représenter au moins les objets *<3* et *Blood Of The Martyr* dont les détails sont donnés dans la section "Quelques chiffres".

2.5 Quelques chiffres

Les valeurs qui sont données dans ces règles ne sont que des valeurs pour vous guider dans l'élaboration du jeu. Rien ne vous empêche donc, sauf contre-indication, de les modifier pour rendre le jeu plus intéressant.

- *Donjon:*
  - Une salle se compose de 49 carreaux (un carré de 7*7) ainsi que d'un mur entourant ces 49 carreaux, dans lequel se situent les portes.
  - Un obstacle classique (rocher, piques, ...) mesure un carreau. Rien ne vous empêche d'en faire d'autres de taille différente.

- *Isaac:*
  - Isaac se déplace à une vitesse de 0.01. L'unité est "proportion de l'écran par cycle du jeu". La plupart des unités sont exprimées vis-à-vis des cycles de jeu plutôt que du temps, de manière à conserver l'équilibrage du jeu sur un ordinateur plus lent. Avec cette valeur, cela signifie que Isaac traversera 10% de l'écran en 10 cycles de jeu, ce qui correspond à 40% de l'écran par seconde si le jeu tourne à 40 FPS (Frames Par Seconde).
  - Isaac recharge ses larmes à une vitesse de 20. L'unité ici est "Cycle de jeu par larme". Cela signifie qu'une attaque pourra avoir lieu tous les 20 cycles de jeu. Cela s'exprime de manière équivalente en "larme par cycle de jeu" en disant qu'Isaac a une vitesse de tir de 1/20.
  - Isaac mesure un peu moins d'un carreau de largeur et de hauteur.
  - Isaac a 6 points de vie, ce qui correspond à 3 cœurs (chaque cœur vaut 2 points de vie).

- *Monstres:*
  - Une araignée a 5 points de vie, et une mouche 3.
  - Une araignée, comme une mouche, inflige 1 point de dommage.
  - Une araignée se déplace, lorsqu'elle le fait, à une vitesse de 0.02. Une mouche de déplace à un huitième de la vitesse d'Isaac.
- *Équipements:*
  - Le *<3* permet d'augmenter les points de vie max de Isaac de 2 (soit 1 cœur complet), puis le soigne entièrement.
  - Le *Blood Of The Martyr* permet d'augmenter les dégâts qu'inflige Isaac de 1.
- *Affichage:*
  - Le jeu fonctionnera à 40 FPS. Cette valeur ne doit pas être augmentée afin que le jeu reste assez fluide, même sur des ordinateurs moins performants. Cette valeur a été choisie relativement basse car StdDraw n'est pas une bibliothèque graphique très efficace.
  - La fenêtre d'affichage fera $65 * 9$ (pixels par carreau * nombre de carreaux) par $65 * 9$. Une fenêtre carrée (ici $585 * 585$) simplifie les déplacements, surtout lorsque les vitesses sont exprimées en proportion d'écran. Cette fenêtre doit rester assez petite, toujours pour permettre au jeu d'être fluide.

2.6 Triche

Pour simplifier la correction, vous devez implémenter quelques fonctionnalités de triche. Nous attendons que :

- Appuyer sur la touche 'i' rend Isaac invincible ;
- Appuyer sur la touche 'l' rend Isaac rapide ;
- Appuyer sur la touche 'k' permet de tuer instantanément tous les monstres de la salle ;
- Appuyer sur la touche 'p' rend Isaac si puissant que chacune de ses larmes tue instantanément un monstre touché ;
- Appuyer sur la touche 'o' donne 10 pièces à Isaac.

3 Implémentation

Nous vous fournissons un squelette de code qui gère certaines mécaniques de base de notre jeu en Java. Les classes qui vous sont fournies vous donnent une base solide sur laquelle construire votre code. Nous les expliquons un peu ici, et elles sont également commentées. Ce code s'appuie sur la bibliothèque StdDraw, que vous connaissez déjà, et qui a été choisie pour sa simplicité d'utilisation.

Voici l'architecture du squelette fourni ainsi qu'une description rapide de chacun des fichiers.

```
.
|-- gameWorld
|   |-- GameWorld.java
|   `-- Room.java
|-- gameloop
|   `-- Main.java
|-- gameobjects
|   `-- Hero.java
|-- libraries
|   |-- Keybinding.java
|   |-- Physics.java
|   |-- StdDraw.java
|   |-- Timer.java
|   `-- Vector2.java
`-- resources
    |-- Controls.java
    |-- DisplaySettings.java
    |-- HeroInfos.java
    |-- ImagePaths.java
    `-- RoomInfos.java
```

- La classe *Main* s'occupe de l'initialisation du monde, et de lancer la boucle principale du jeu qui a pour rôle de mettre à jour les entités et de les afficher.
- La classe *GameWorld* s'occupe de la gestion du monde et de ce qui le compose, par exemple les salles.
- La classe *Room* représente une pièce du donjon, pour l'instant bien vide.
- Dans le package *libraries*, on peut trouver les classes
  - *StdDraw* et *Keybinding*: La bibliothèque graphique *StdDraw* et un fichier donnant les opcode des touches.
  - *Vector2*: Une classe représentant une paire de *double*, accompagné des opérations vectorielles classiques. Bien pratique pour représenter des positions ou des tailles.
  - *Timer*: Une classe permettant la gestion du temps de la boucle principale du jeu.
  - *Physics*: Une classe fournissant la collision entre rectangles.
- Dans le package *resources*, on peut trouver des classes ne fournissant que des variables statiques et finales, ce qui permet d'isoler les constantes que l'on va utiliser plutôt que de les laisser s'éparpiller dans le code.

Pour ce qui est de la logique du programme, nous avons la classe *Main* qui gère la boucle principale du jeu. Dans le code fourni, cette boucle est exécutée toutes les 25 millisecondes. À chaque cycle, la fonction de mise à jour du monde est appelée. Le monde met à jour la pièce courante (celle dans laquelle se situe Isaac) qui, elle, exécute la méthode de mise à jour de toutes les entités de la pièce, ce qui a pour effet de gérer leur déplacements et actions. Ensuite, le monde appelle la fonction *drawRoom()* de la pièce, ce qui dessine le sol de la pièce et toutes les entités présentes (pour l'instant seulement Isaac).

C'est à vous de concevoir le reste de la hiérarchie de classes permettant de réaliser ce jeu. Vous pouvez toutefois remarquer que les règles fournies ci-dessus peuvent vous donner une bonne intuition d'une partie des classes dont vous avez besoin !

Nous vous conseillons de passer un peu de temps à bien comprendre ce squelette de code avant de commencer à l'étendre. Aussi, n'hésitez pas à modifier une partie du code fourni lorsque vous commencerez à le compléter.

De nombreuses images sont aussi fournies dans le dossier *images*. Vous pouvez les utiliser ou non, mais sachez qu'elles sont là et prêtes à l'emploi.

4 Notation

Le projet est à réaliser en binôme. Vous déposerez votre travail sur Moodle pour début janvier (cf Moodle pour la date exacte). Vous devrez déposer un dossier compressé *.zip* avec :

- Le code source de votre jeu (ainsi que les images), en tant que projet Eclipse. Vos

évaluateurs utiliseront la fonctionnalité “File - Import... - Existing project into workspace” de Eclipse pour charger votre projet. Si ça ne marche pas car vous avez rendu sous le format d'un autre IDE, vous aurez la note de 0/20. Avant de rendre votre projet, faites vous-même la démarche de l'importer dans Eclipse afin de vérifier que tout fonctionne.
NB: cela ne vous empêche pas d'utiliser votre IDE préféré, et il se peut que ce ne soit pas Eclipse. Vous devez juste réserver un peu de temps à la fin pour préparer un projet Eclipse pour le rendu.

- Un fichier README.txt, donnant vos noms et une description de votre jeu: ce que vous avez fait, les commandes de votre jeu, les bonus que vous avez ajoutés. Écrivez ici tout ce à quoi vous voulez que votre évaluateur fasse attention.

Si plusieurs binômes rendent des projets *trop* similaires, tous les participants de la triche auront la note de 0/20 (cette note compte pour 1/4 de votre moyenne totale de PO). Vous aurez la note de 14/20 si vous implémentez un “jeu de base” correspondant exactement aux règles indiquées ci-dessus. Le jeu doit être totalement fonctionnel (aucune erreur). Il n'a pas besoin d'être joli, mais doit être jouable et simple à utiliser (testez le vous-même ou faites le tester par vos amis). Le code doit présenter une conception objet propre et des commentaires Javadoc pour les classes et méthodes principales. L'implémentation doit impérativement utiliser des collections Java.

Si vous désirez avoir plus de 14/20, vous pouvez gagner des points bonus en améliorant le “jeu de base”. Laissez parler votre créativité! En sachant que n'étant pas une formation artistique, nous récompenserons mieux les améliorations démontrant votre maîtrise de la programmation que votre capacité à coller de jolies images.

5 Idées d'amélioration

Voici certaines idées d'amélioration, certaines étant plus dures que d'autres. Vous n'êtes pas limités aux améliorations citées ci-dessous (inspirées du jeu de base Isaac).

- Faire en sorte de respecter l'idée d'un roguelike en générant aléatoirement (avec des règles pour que le jeu soit entièrement jouable) la carte, les objets, monstres & obstacles qui apparaissent.

- Ajouter des objets actifs. Ces objets possèdent un compteur de charge active qui augmente de 1 à chaque fois qu'Isaac nettoie une salle de ses occupants monstres. Lors de l'appui sur une touche, par exemple *espace*, si l'objet est chargé, il se décharge et applique alors un effet.

- Bien que le jeu ne propose actuellement que de jouer Isaac, il existe en vérité plusieurs personnages, chacun avec des particularités bien à eux. Vous pouvez vous en inspirer ou même créer les vôtres.

- Il existe beaucoup d'autres types de consommables ramassables par Isaac : Des cœurs différents (bleu/noir pour les plus basiques), des clés qui permettent l'ouverture de portes spéciales et des bombes qui permettent à la fois d'infliger des dégâts après un délai et à la fois de briser des obstacles

- Certaines salles ne possèdent aucune porte pour y accéder. Il faut deviner son emplacement (ce qui est possible car elles n'apparaissent qu'à des endroits spécifiques) puis faire exploser le mur à l'endroit où devrait normalement être la porte pour pouvoir y accéder, ce qui aura pour effet de trouer le mur qui agira alors comme une porte.

- Le jeu se déroule habituellement sur plusieurs étages, qui correspondent à des niveaux. Lors de la victoire contre un Boss intermédiaire, on débloque un chemin qui permet d'accéder à l'étage suivant et ainsi de suite jusqu'au Boss Final. Chaque étage augmente la difficulté du jeu en faisant apparaître des ennemis de plus en plus redoutables !