12

Développer un éditeur de niveaux

Ce chapitre est optionnel mais fortement recommandé (et très utile !). Nous allons voir comment développer l'éditeur de niveaux que je vous ai présenté au chapitre précédent. Comme je vous l'ai fourni avec les sources, vous pouvez aussi directement l'utiliser et passer au chapitre suivant. Cependant je vous encourage à suivre ce chapitre afin d'apprendre à créer un tel éditeur par vous-même. Cela vous aidera à créer vos propres outils, à personnaliser votre éditeur, à adapter le mien, à le faire évoluer et à aller beaucoup plus loin. Ce sera très formateur et très pratique pour vos futurs projets. Tous les studios de création de jeux développent des outils en interne pour justement se faciliter la vie dans le développement de jeux.

12.1. Préparation du projet

Avant de coder notre éditeur de niveaux, nous allons créer un nouveau dossier. Cette fois-ci, ne mettez pas de chiffre au nom du dossier car il s'agit d'un projet à part. Créez simplement un nouveau dossier `LevelEditor` dans lequel nous allons travailler. Dans ce dossier, ajoutez l'image du TileSet (voir Figure 12.1). Je vous l'ai fournie avec les sources du livre, elle s'appelle `tileset.png`.

Figure 12.1 : *Le TileSet à utiliser*

Créez ensuite un fichier `conf.lua` et ajoutez le code suivant :

```
function love.conf(w)
  w.window.width = 1356
  w.window.height = 864
  w.title = "Tilemap Editor - Anthony Cardinale"
  w.console = true
end
```

Ici nous donnons une taille à notre fenêtre. Remarquez que celle-ci est plus grande que le jeu. Nous avons besoin de marge pour afficher, en plus de la map, le TileSet ainsi qu'un futur menu qui sera placé à droite de l'écran. J'ajoute également un titre et j'active l'option permettant d'afficher la console pour déboguer.

Toujours dans votre projet, ajoutez un fichier `settings.lua` dans lequel vous allez créer des variables pour la taille de la grille et des tuiles. Il s'agit de choses que nous avons déjà faites ensemble dans les chapitres précédents.

```
mapW = 32
mapH = 24
tileSize = 32
```

Nous avons maintenant besoin du fichier `main.lua`, créez-le. Ajoutez la ligne `require "settings"` pour y inclure notre fichier d'options.

Notre dossier de travail est prêt, nous avons ce qu'il nous faut pour commencer à développer notre outil.

12.2. Création de la grille

La première chose à faire est de dessiner la grille à l'écran. La grille correspond à la taille de la fenêtre de notre jeu et est composée des 768 tuiles affichées à l'écran. Cette grille servira de repère et c'est dans celle-ci que le développeur (ou le designer) créera ses maps.

Comme nous le savons maintenant, pour dessiner à l'écran, nous avons besoin de la fonction `draw` de LÖVE. Créez donc cette fonction puis appelez la fonction `DrawGrid()` qui sera une fonction personnelle que nous n'avons pas encore créée. Elle aura pour rôle de dessiner la grille c'est-à-dire les lignes horizontales et verticales.

```
function love.draw()
  DrawGrid() -- Dessiner la grille
end
```

Nous devons maintenant créer notre fonction `DrawGrid`.

Nous avons vu que le module `graphics` de LÖVE est capable de dessiner beaucoup de formes simples. Pour dessiner une ligne, vous utiliserez ainsi la fonction `love.graphics.line` et passerez en paramètres `(x1,y1,x2,y2)` qui correspondent aux deux extrémités du segment. Pour tester, vous pouvez par exemple écrire `love.graphics.line(0, 0, 2000, 2000)`. Si vous testez, vous verrez qu'une ligne barrera votre

écran, c'est que cela fonctionne. Supprimez cette ligne car ce n'était qu'un exemple pour illustrer.

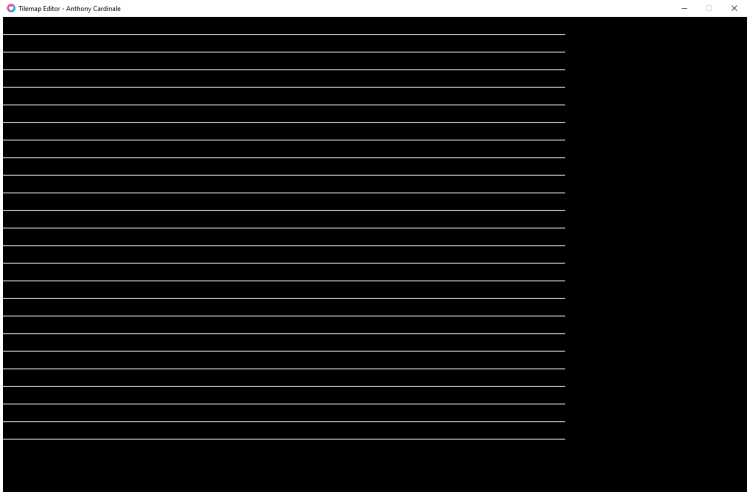
Nous allons créer un ensemble de lignes qui constitueront notre grille. Nous commencerons par les lignes horizontales. Ce que nous devons faire, c'est dessiner des lignes espacées de 32 pixels. Il nous faudra un total de 24 lignes horizontales pour représenter les 24 lignes de la grille.

Pour dessiner ces lignes, utilisez la fonction `line` dans une boucle `for` qui se répétera le nombre de fois nécessaire :

```
-- Lignes horizontales
for y=tileSize, mapH*tileSize, tileSize do
  love.graphics.line(0, y, mapW*tileSize, y)
end
```

Pour ce bout de code, j'utilise les variables de notre fichier `settings`. Nous avons la variable `y` qui sera égale à 32 et qui s'incrémentera de 32 jusqu'à atteindre 24×32=768. Pour chaque valeur de `y` nous dessinons une ligne qui aura la taille de la largeur de la fenêtre de notre jeu :

Figure 12.2 : *Nous dessinons les lignes horizontales*



Copiez cette boucle et adaptez-la pour les lignes verticales.