Projet Long de Technologie Objet

Présentation des sujets

Groupe EF02

Élèves :

ARRIX-POUGET Baptiste

DEMAZURE Clement

DREUMONT Evann

GIULIANI Astrid

GUTIERREZ Tom

LAGRANGE Angel

SABLAYROLLES Guillaume

THEVENET Louis

12 Février 2024

1. Rogue-like

Une partie de jeu rogue-like se déroulerait de la manière suivante. Un personnage (protagoniste) entre dans un donjon, à la recherche d'un trésor caché dans ses entrailles. Pour l'atteindre il va devoir explorer un par un tous les étages de ce donjon. Pour arriver au bout, il devra :

- accumuler des objets.
- améliorer ses caractéristiques et son équipement pour faire face au danger toujours grandissant.
- survivre dans ce milieu hostile (gestion de la faim, du sommeil et de la soif).

Les caractéristiques du jeu sont les suivantes :

- Vue en 2D du dessus.
- Pas de progression sauvegardée entre les parties, tout est remis à zéros quand le joueur échoue.
- Présence d'ennemis avec des comportements contrôlé par une IA simpliste
- Une très grande place à l'aléatoire : Le plan du donjon est aléatoire. Emplacement des portes, tracé des couloirs, forme et taille des salles sont générés aléatoirement. Les effets des objet rencontrés par le joueur varient selon parties, ils sont tirés au sort pendant la création du donjon parmis une liste de d'effets possibles.
- Gestion de la vision : le personnage n'a accès qu'à son environnement direct; il doit faire attention aux potentiels pièges, salles, trésors ou ennemis cachés dans l'ombre.
- Une carte mise à jour en continu permet au joueur de se repérer dans les parties du donjon qu'il a déjà explorées.

L'un des points-clef du projet est la gestion de l'aléatoire, i.e. comment réaliser un enchaînement de pièce ou de couloir aléatoire tout en s'assurant que toute les pièces soient cohérentes (pas de superpositions, de portes menant sur le vide) et en s'assurant que tous les points importants soient atteignables de n'importe où par des déplacements simples.

Un autre point clef réside dans les intéractions entre le personnage et son environnement : se déplacer dans les salles, désamorcer les pièges, se battre avec les individus hostiles rencontrés ou parlementer avec des personnages amicaux. Etudier des murs pour trouver une porte dérobée ou le sol pour un piège mortel.

Cette idée de roguelike permet d'ajouter facilement des mécaniques de jeu, ou des fonctionnalité comme un éditeur de donjon pour le créer à la main, ou modifier des paramètres influant sur sa génération (nombre de salle par étage, dimension des couloirs), de nouveaux ennemis / objets / pièges, l'ajout de personnages amicaux avec des quêtes à accomplir. Ce qui nous garantit d'avoir suffisamment de matière pour un projet de cette envergure.

2. Jeu de survie (bac à sable)

L'objectif est de programmer un jeu simpliste de survie, sur l'exemple de Minecraft.

Dans un jeu de survie, il y a plusieurs objectifs :

- se nourrir.
- fabriquer des outils, armes et équipements avec les matériaux récupérés.
- améliorer les statistiques de son personnage avec des points d'expérience (points de vie, d'endurance, de force …).
- explorer la carte.

Le joueur se déplace dans une zone en 2D (vue de dessus ou de côté) générée procéduralement, c'est-à-dire de manière automatisée et à grande échelle tout en répondant à un ensemble de règles. L'idée originale est de proposer un jeu en 2D, mais il peut être intéressant de pousser le projet vers un jeu encore plus proche de Minecraft, en 3D, en utilisant des voxels.

Il peut rencontrer diverses choses extérieures :

- des arbres, des rochers, des minerais (qu'il peut détruire pour obtenir des matériaux).
- des animaux (que l'on peut tuer pour se nourrir).
- des ennemis (qu'il faut abattre pour obtenir des points d'expérience).
- des coffres au trésor, contenant des matériaux, armes, équipements.
- des zones spéciales, à définir …

Le joueur a donc un inventaire dans lequel il peut placer ses matériaux et les combiner pour crafter. Par exemple, à partir de bois et de pierre, il peut concevoir une pioche ou bien une épée ; avec une pioche, il peut extraire du charbon et du fer. Avec certaines matières telles que le fer, le diamant ou encore le cuir, le joueur pourra assembler des armures pour résister aux monstres. Chaque outil est construit à partir d'éléments basiques récoltés.

Ainsi, il semble pertinent de réaliser ce projet pour approfondir notre connaissance de Java. L'utilisation de classes est cohérente et intuitive dans un tel jeu vidéo (ex : une classe être vivant, dont des classes personnage et ennemi peuvent hériter, avec différentes méthodes associées). D'autre part, tout ajout de nouvelles fonctionnalités sera envisageable, ce qui nous donnera un travail suffisamment conséquent pour la réalisation d'un projet long.

Fig. 1. – Exemple d'inventaire dans le jeu Minecraft

3. Doom engine

Doom est un jeu de combat à la première personne sorti en 1993 qui a révolutionné le monde du jeu vidéo par ses graphismes. En effet, le jeu réalise un semblant de 3D dans lequel le personnage se déplace.

C'est un choix de projet pertinent car il nous permettra d'explorer différents domaines de la programmation en Java tels que la gestion des entrées/sorties, l'affichage à l'écran, le calcul avancé ou l'interface utilisateur.

Le jeu original utilisait une méthode appelée Ray Casting pour afficher l'univers à l'écran. Cette technique consiste à simuler la projection de rayons depuis le point de vue du joueur pour déterminer quels éléments du monde lui sont visibles. Ainsi, les ennemis, murs et objets sont représentés par un plan en 2D dans un espace virtuel (la carte), les intersections entre ces plans et les rayons déterminent si l'élément est affiché à l'écran ou non. Cet univers en trois dimensions est donc une illusion qui permet de conserver la simplicité d'un monde en deux dimensions.

Le caractère jeu-vidéo de ce sujet nous permettra, une fois la partie graphismes terminée, d'ajouter des éléments et des fonctionnalités supplémentaires. Par exemple, les jeux rogue-like et survival proposés en Chapitre 1 et Chapitre 2 pourraient être implémentés graphiquement en l'utilisant.

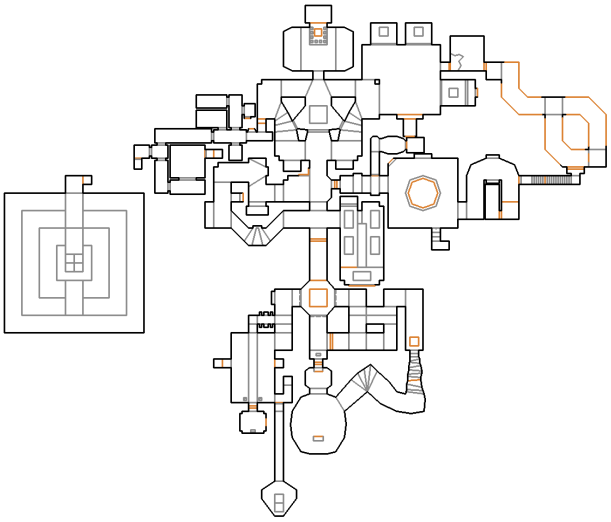

Fig. 2. – Représentation en 2D d'une carte du jeu Doom

Fig. 3. – Représentation en 3D d'une carte du jeu Doom

4. Logiciel de dessin vectoriel

L'idée derrière ce projet est de créer un logiciel de dessin. Il s'agirait de créer une interface graphique depuis laquelle l'utilisateur pourrait sélectionner différents outils nécessaires à la production de dessin tels que la création de lignes et de formes géométriques.

La taille du trait et la couleur seront modifiables, permettant des dessins divers et variés. Cera également possible l'utilisation d'outils tels qu'un pinceau, un surligneur, ou encore une gomme, avec une taille en pixels variable, qui permettra de supprimer certaines parties de la production en effaçant les pixels dessinés afin que l'utilisateur se rapproche au maximum d'une expérience de dessin réel.

Le logiciel sera aussi équipé d'une fonction de calques sur plusieurs couches afin de pouvoir travailler sur plusieurs parties du projet sans modifier le reste de l'ensemble.

De plus le logiciel gérera des images au format SVG durant le processus de création, afin d'avoir une image nette qui puisse s'agrandir et se réduire sans perte de qualité et que les fichiers de l'image soient beaucoup moins volumineux. Il pourra d'autre part permettre l'insertion d'images au format SVG dans un travail afin de pouvoir combiner plusieurs fichiers et les superposer à l'aide des calques.

Ce logiciel pourra aussi permettre à son utilisateur de transformer un dessin depuis certains autres formats en format SVG. Une option de filtre existera, proposant plusieurs filtres différents tels que des filtres de rangements de couleurs, des filtres qui transforment le dessin en noir et blanc, etc.

Enfin, lorsque l'utilisateur aura terminé sa production, il pourra choisir le format d'enregistrement final parmi plusieurs options (jpg, png, ppm, tiff…).

Un tutoriel sera intégré dans l'application afin de pouvoir offrir à l'utilisateur une compréhension profonde du logiciel et une bonne maîtrise.

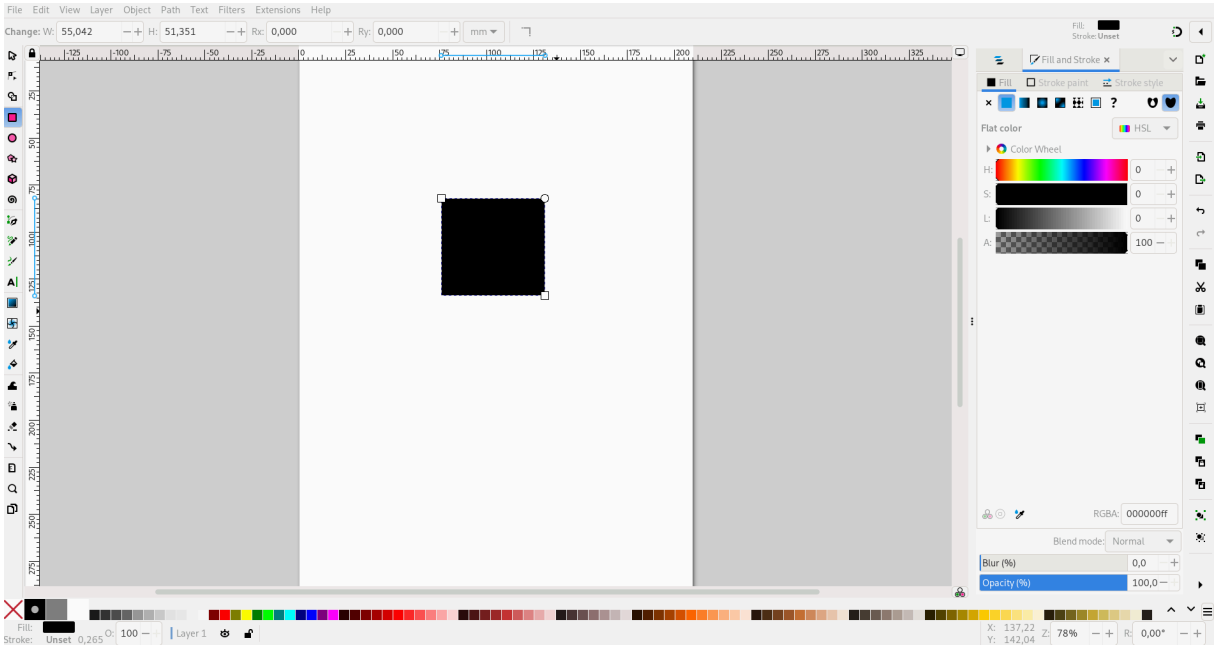

Fig. 4. – Exemple de logiciel de dessin vectoriel Inkscape