Sujets Projet Long TOB
Groupe AB2

---

1 - Jeu à la Roguelike

Description :

Le but ici est de créer un jeu vidéo d'exploration de donjons à plusieurs étages générés procéduralement. Le joueur débarque dans un donjon généré procéduralement (donc chaque partie est unique) et il doit explorer un étage de ce donjon pour trouver une salle dans laquelle se trouve une clé qui lui permettra ensuite d'ouvrir une autre salle (ailleurs dans le même étage du donjon) afin de pouvoir passer à l'étage suivant. Un boss sera présent tous les 5 étages. Il sera nécessaire de l'affronter et le vaincre pour passer à l'étage suivant.

Le joueur possède 100 points de vie (abrégé en PV) au début de la partie. Si le joueur se fait toucher par un monstre, il perd un certain nombre de PV en fonction de la puissance du monstre. Si le joueur perd ses 100 PV, la partie est finie. Il est à noter que certains objets pourront redonner des PV voire augmenter le nombre total de PV du joueur.

Le joueur possède un inventaire, qui lui permettra de stocker des pièces (monnaie du jeu, réinitialisée à chaque partie), des objets et des armes.

Les objets seront soit des consommables (c'est-à-dire des objets à usage unique, comme par exemple un bandage qui redonne un 20 PV au joueur), soit des objets à effet passif (par exemple, un objet armure qui donne au joueur +5% de résistance face aux monstres).

Les armes auront des effets différents. Certaines armes ne s'utilisent qu'au corps à corps (comme une épée par exemple) et certaines armes peuvent permettre au joueur d'attaquer à distance (comme un pistolet par exemple).

Dans (quasiment) chaque salle d'un étage se trouvent des monstres qu'il faut vaincre afin d'ouvrir l'accès aux salles voisines. Chaque monstre vaincu donnera l'opportunité au joueur de gagner des pièces. Par exemple, un monstre de bas niveau (c'est-à-dire un monstre peu puissant et facile à vaincre) pourra donner 0 ou 1 pièce. Un monstre de niveau un peu supérieur pourra donner entre 0 et 3 pièces. Un monstre d'encore plus haut niveau pourra donner entre 1 et 4 pièces, etc.

Cependant, quelques salles sont spéciales, et chacune de ces salles spéciales ne peuvent apparaître qu'au maximum une fois par étage (sauf pour la salle de téléportation qui apparaît obligatoirement en paire). Chacune de ces salles spéciales a une probabilité

d’apparition différente (elles sont évidemment moins susceptibles d’être générées que les salles classiques qui contiennent les monstres). Ces salles spéciales sont les suivantes :

- Une boutique : salle qui permet d’acheter un objet parmi 3 proposés. La sélection des 3 objets se fait aléatoirement selon le numéro de l’étage. C’est-à-dire que pour les 5 premiers étages, les boutiques proposeront 3 objets choisis aléatoirement parmi un catalogue de 10 objets. Ensuite, pour les étages de 6 à 10, les boutiques proposeront toujours 3 objets mais cette fois-ci ils seront choisis aléatoirement parmi un catalogue de 20 objets (qui englobe les 10 objets précédents). Chaque objet aura bien entendu un prix qui lui est attribué mais ce prix pourra varier aléatoirement de plus ou moins 5% (pour ajouter une mini touche d’aléatoire). Un des objets peut être remplacé par une arme avec une probabilité d’environ 1/5.
- Une salle “coffre” : cette salle contient uniquement un coffre, que l’on peut ouvrir grâce à une petite clé. Cette petite clé peut être soit achetée dans une salle boutique si celle-ci en propose, soit récupérée après avoir battu un monstre (faible taux d’apparition, environ 2%). Si le joueur possède une petite clé, il pourra ouvrir le coffre et obtenir un objet choisi aléatoirement parmi une sélection d’objets. Il aura 70% de chance d’avoir un objet aléatoire parmi une sélection d’objets (en fonction du numéro de l’étage actuel, tout comme pour les salles boutiques) et 30% de chance d’avoir une arme choisie aléatoirement parmi une sélection d’armes (la sélection varie aussi, tout comme pour les objets, selon le numéro de l’étage actuel). Dans le cas où le joueur ne possède pas de petite clé, il ne pourra tout simplement pas ouvrir le coffre.
- Une salle de téléportation : dans chaque étage se trouvera obligatoirement 2 salles de téléportation. Ces salles contiennent, comme leur nom l’indique, un téléporteur qui permettra au joueur de se téléporter à l’autre salle de téléportation (du même étage). Ces salles devront être espacées d’au moins 3 salles (pour éviter que les deux salles de téléportation se trouvent l’une à côté de l’autre, ce qui n’a aucun intérêt).
- Une salle “clé d’étage” : dans cette salle se trouve l’unique clé permettant d’ouvrir la salle “étage suivant”. Il va de soi que cette salle est générée exactement une fois par étage.
- Une salle “étage suivant” : comme son nom l’indique, cette salle est l’unique salle de l’étage qui permet de quitter l’étage actuel pour se rendre à l’étage suivant. Aucun monstre ne se trouve dans cette salle. Elle est générée à chaque étage. Elle est verrouillée et nécessite la clé trouvée dans la salle “clé d’étage” pour l’ouvrir.
- Une salle “boss” : comme son nom l’indique, la salle “boss” est l’unique salle de l’étage qui contient le boss. Elle apparaît tous les 5 étages, c’est-à-dire à

l'étage 5, 10, 15, etc. Cette salle est obligatoirement succédée par une salle "étage suivant".

On se limitera dans un premier temps à cette sélection de salles spéciales mais d'autres pourraient voir le jour.

***PS :** Si les combats en temps réel s'avèrent trop compliqués à mettre en place, on pourra plutôt utiliser un système de combat en tour par tour. De même, si un déplacement libre d'un joueur s'avère trop complexe à mettre en place, on pourra s'appuyer sur un déplacement case par case (ce qui nécessite de découper la carte en petite cases).*

Interface :

Afin de jouer, il est bien entendu nécessaire de fournir une interface graphique, qui traite en temps réel les actions du joueur pour les retranscrire sur l'écran. Les actions du joueur seront données au programme par l'intermédiaire du clavier de l'ordinateur.

On adoptera une vue du dessus (comme si on était au-dessus du personnage) pour simplifier les choses.

Documentation :

- Vidéos YouTube :
  (Procedurally Generated 2D Dungeons)
  https://www.youtube.com/watch?v=rBY2Dzej03A

  (Herbert Wolverson - Procedural Map Generation Techniques)
  https://www.youtube.com/watch?v=TlLIOgWYVpI

  (Procedural Generation: Programming The Universe)
  https://www.youtube.com/watch?v=ZZY9YE7rZJw

---

2 - Jeu de construction de ville :

L'idée ici est de créer un jeu de construction de ville. Globalement, le joueur pourra créer et gérer une ville (ou plusieurs). Dans ce dernier nous avons différentes entités : les habitants, les ressources, les bâtiments, les objets/animaux…

- Les bâtiments ont besoin de ressources pour être construits. On pourra définir des maisons, des hôpitaux, des usines, des supermarchés, des salles de cinéma, des refuges animaliers… Et chacun de ces bâtiments aura des fonctionnalités différentes. Ces derniers peuvent être définis par : la capacité d'accueil, le coût de construction, les revenus qu'il génère…
- Les ressources telles que l'argent, le bois, la pierre doivent être collectées grâce à des activités commerciales gérées par les habitants. Le joueur pourra par exemple planter des légumes, les récolter, les envoyer à l'usine pour en fabriquer de la nourriture, les vendre… Plus généralement, il pourra les collecter, les stocker et les dépenser.
- Chaque habitant a des besoins de base tels que le logement, la santé, le divertissement… il sera défini des caractéristiques telles que le bonheur, l'emploi, la santé…
- La ville pourra être basée sur une économie simulée où les citoyens consomment et produisent des biens.
- Les objets/animaux seront des éléments tels que des animaux de compagnie, des éléments décoratifs, des terrains pour planter des légumes…

De plus, l'utilisateur pourra créer plusieurs villes qui ont chacune des caractéristiques différentes (une cité, une île, une campagne, une ville dans l'espace…). Ces caractéristiques ainsi que d'autres pourront être débloquées grâce à un système de points commun à toutes les villes.

Si les caractéristiques des habitants sont au-dessus d'un certain seuil (les habitants sont contents, en bonne santé, ont des emplois…) le joueur gagne des points. Au cas contraire, il en perd. Les points permettent aux utilisateurs d'agrandir leurs villes, de débloquer des bâtiments, de nouvelles villes… et les ressources leur permettent de les acheter ou de les construire.

Pour rajouter un côté amusant au jeu, des événements aléatoires et inattendus tels que des catastrophes naturelles, des migrations de peuples… peuvent changer le cours du jeu.

Finalement, il faudra aussi intégrer un système de transport en commun et individuel, pour se déplacer dans une même ville et entre les villes (des voitures, des trains, des avions, des fusées…)

Afin de pouvoir jouer à ce jeu, une interface graphique pourra être créée.

---

3 - Déchiffrer la théorie musicale

Le but est de créer un outil qui se basera sur la théorie musicale pour générer des partitions, les transposer, ou même les déchiffrer et les lire.

Dans un premier temps, on permettra à l'utilisateur de générer des gammes de son choix (gamme majeur, mineure, chromatique, décatonique, par ton…) et à partir d'une note de son choix (do, do#, re…). Ces gammes pourront nous servir par la suite à générer des accords; nous proposerons ainsi à l'utilisateur plusieurs méthodes par lesquelles il pourra générer un accord : à partir d'une gamme, et un type d'accord (majeur, mineur, diminué…), ou tout simplement à partir d'une seule note (ce qu'on appelle une fondamentale), ou à partir d'une note et d'une gamme… La construction d'un accord pourra donc être définie par des constructeurs ou par des méthodes de classe.

Nous permettrons à l'utilisateur de générer des bouts de partitions à partir de ces accords ainsi définis, en respectant la théorie musicale et les gammes, tout en lui laissant le choix du style de musique qu'il a en tête (dramatique, jazz, hyped…), ainsi que le nombre de mesures souhaité, ainsi que le nombre d'accord souhaité dans ce bout de partition.

Après avoir créé ces bouts de partitions, nous pourrons ainsi générer des partitions complètes en laissant à l'utilisateur le choix de nombre de mesure total souhaité, et le nombre de changement d'accord, ainsi la possibilité de rajouter des riffs ou des bridges dans la partition.

Nous pouvons aussi essayer de lire des partitions déjà existantes, et essayer de déchiffrer les accords utilisés, ainsi que les gammes utilisées dans ces partitions.

Nous offrons à l'utilisateur de transposer des partitions préexistantes, ce qui est utile pour mélanger des instruments à cordes et des cuivres par exemple, ou même pour changer la tonalité d'une partition.

4 - Logiciel d'aide à l'écriture de Programme de Test

Description :

Le concept de ce projet est de développer un logiciel qui permet d'automatiser une partie des procédures de test d'un module ou d'une classe à partir de son implantation dans un langage de programmation et de ses commentaires.

On pourrait dans un premier temps se restreindre uniquement au langage de programmation Java. Ainsi, on pourrait s'appuyer sur la javadoc pour nous aider à générer de manière automatique une partie des procédures de tests.

Interface :

L'objectif est de réaliser une interface graphique pour l'utilisateur afin de lui offrir différentes fonctionnalités. L'utilisateur n'aurait qu'à fournir au logiciel le fichier source du code de son module ou de sa classe (en supposant que les commentaires soient correctement tapés) pour que le service lui affiche une page avec une première partie des procédures de tests ainsi que d'autres sections vides délimitées par des commentaires lui suggérant par quoi il pourrait compléter le code. Et ce, dans le but d'automatiser des procédures de tests et d'aider l'utilisateur à avoir un programme de test le plus exhaustif possible.