Cahier des charges

Fonctionnalités :

- Moteur de rendu 3D Vulkan (Paul Boursin) ~ 3 mois
- Modèle de jeu (David Hamelin, Hugo Spinat) ~ 2 mois
- Interface 2D (David Hamelin) ~ 1 mois
- Générateur procédural de carte (Hugo Spinat) ~ 2 semaines
- I.A (Simon Pellicer) ~ 3 mois
- Pathfinding (Hugo Spinat) ~ 2 semaines
- Implémentation des différentes unités (David Hamelin) ~ 1 mois
- Pile réseau (Hugo Spinat) ~ 1 mois

Moteur de rendu 3D Vulkan

Vulkan est une API de Khronos pour communiquer avec les drivers de carte graphique, de manière à ce qu'une application puisse tourner de la même manière sur toutes les cartes graphiques et plateformes. Comme OpenGL, il est quasiment présent partout, mais existe pour donner plus de possibilités au développeur. En conséquence, l'API est très verbeuse, et requiert beaucoup de mise en place en avance.

- Mise en place de tout le code de base requis pour Vulkan.
- Implémentation d'un lecteur de fichier obj et mtl qui permet de lire des fichiers venant d'applications de création de modèle 3D.
- Implémentation du instancing pour positionner tous les objets dans la scène.
- Gérer les events envoyés du modèle pour changer l'état d'affichage tel que la position des unités.
- Affichage des cases en couleur différentes pour les cases sur lesquels l'unité sélectionnée peut aller ou attaquer.
- Animation des objets simples, en interpolant entre l'ancienne position et la nouvelle, parfois avec une courbe quadratique.
- Rendu de l'eau, un plan au niveau 0, avec une simulation simple de vagues (parce que c'est joli).

Modèle de jeu

La classe principale du modèle est `Game`. Elle contient une classe `Board`, qui est un tableau bidimensionnel d'Hexagone. Chaque Hexagone contient sa position dans le tableau, de manière à ce qu'on puisse trouver les voisins rapidement. Nous utilisons le modèle ECS, avec l'API `entt`, donc les hexagones, unités, et ville sont des entités, et leur attributs sont stocké dans des Composant, qu'on peut dynamiquement ajouter ou enlever. Les système peuvent itérer sur les composants, les modifier, les supprimer, ou encore ajouter des entités.

Interface 2D

L'interface 2D utilise `dear imgui`, une librairie permettant de créer des interfaces en utilisant vulkan comme back-end. Le menu principal est animé, et permet de lancer une partie, ou en charger une nouvelle. Une fois la partie lancé, on peut appuyer sur la touche Échap afin de mettre en pause le jeu, ainsi que sauvegarder, charger, ou quitter.

Générateur procédural de carte

Le générateur procédural de carte, génère une carte aléatoire. Pour chaque case, le générateur doit décider;

- La hauteur
- La présence ou pas de ville

Pathfinding

Le Pathfinding sert à indiquer où est ce que les unités peuvent aller. On profite de l'ECS pour avoir une implémentation élégante.

Implémentation des différentes unités

Les unités auront des statistiques différentes, par exemple, le lancier de base sera moins mobile qu'un cavalier, mais les cavalier ne pourront pas grimper aux montagnes. Voici la liste des unités:

- Lancier (soldat à pied basique)
- Guerrier (soldat à pied, avec plus d'attaque)
- Cavalier (soldat à pied, avec plus de mobilité et moins de défense)
- Archer (soldat à pied, moins de mobilité, moins d'attaque, mais permet d'attaquer à distance)

IA

L'IA permettra au joueur de se confronter contre un robot. Il faudra donc proposer trois niveaux de difficultés afin de pouvoir affronter un robot adapté à son niveau. Un niveau de difficulté sera caractérisé par la précision de la fonction d'évaluation. La fonction d'évaluation prendra en argument la position actuelle du jeu, et retournera un réel indiquant si la position nous est favorable. Il s'agira alors d'explorer un graphe pondéré afin de choisir le coup qui maximisera notre fonction d'évaluation. La profondeur de calcul de L'IA sera aussi un paramètre important à prendre en compte et dépendra du nombre de possibilités qui s'offrent au joueur, et des résultats empiriques en temps de calcul que l'on observe.