Projet Logiciel Transversal

Hikaru GOTO – Benjamin HYON

Table des matières

1 Objectif

1.1 Présentation générale

Le but ce projet est la création d'un jeu de stratégie tour par tour basé sur Age of Empires : Age of Kings version DS. Le jeu se concentra sur l'aspect de la bataille et la création d'armée, avec une arborescence des technologies grandement simplifiée par rapport au jeu publié.

1.2 Règles du jeu

Présenter ici une description des principales règles du jeu. Il doit y avoir suffisamment d'éléments pour pouvoir former entièrement le jeu, sans pour autant entrer dans les détails . Notez que c'est une description en « français » qui est demandé, il n'est pas question d'informatique et de programmation dans cette section.

1.2.1 But du jeu

Le principe du jeu est simple : une civilisation à développer, en technologie et en armée, pour but de détruire la civilisation adverse.

1.2.2 Principaux apsects du jeu

- Le joueur doit choisir une civilisation, sous forme d'un choix de **races**. Plusieurs races sont disponibles :

les elfes, les orcs, les nains ou les humains.

Toutes les races forment les mêmes unités : les villageois, l'unité d'infatrie, de cavalrie et de distance.

Cependant, chaque race a une **affinité particulière** :

Par exemple, les orcs forment une infantrie avec une attaque accrue, pendant que les elfes forment des archers plus habiles.

- Le jeu se déroule sur une carte prédéfinie avec différentes **zones**, telles que le marais, la forêt, la montagne, la plaine, la route ou la mer. Chacunes possèdant des attributs différents, la connaissance du terrain est un facteur clé pour la victoire.
  - Marais : maluce de déplacement, passage impossible pour les unités cavalières.
  - Forêt : maluce de vision*, bonus de défense.
  - Montagne : maluce de déplacement, bonus de vision* et de défence.
  - Plaine : 0 attribut.
  - Route : bonus de déplacement.
  - Mer : Aucune troupe autorisée.

  (la vision influt sur la distance de tir des archers)

- Sur cette carte est présente différentes **ressources**, de type constante et instantanée : leur repérage et leur prise de contrôle sont les aspects les plus importants du jeu.
  - Ressource instantannée : Les ruines, les trésors et les animaux permettent d'obtenir des ressources dés la prise du contrôle de la zone. Cependant, après l'obtention de la ressource, la ressource disparaît.
  - Ressource constante : les champs et les gisements d'ors ; ils nécéssitent leur prise de contrôle à l'aide d'une construction dédiée (un moulin pour les champs, une mine pour l'or). Après contrôle de la zone, une quantité constante de ressource sera obtenue chaque tour.

1.2.3 Déroulement du jeu

- Les joueurs jouent un jour chacun leur tour. Chaque unité pourra se déplacer et/ou attaquer une fois par jour.
- Tous les joueurs commencent par une unité d'infantrie, et une unité de villageois.
- Les villageois sont les seules unités capable de construire. Seul les centres-villes peuvent créer les villageois.
- Le nombre de construction et d'unités autorisé varie en fonction de l'avancée technologique et du nombre de ressources constantes contrôlées par le joueur.
- Le partie se termine lorsque tous les centres-villes adverses sont détruits.

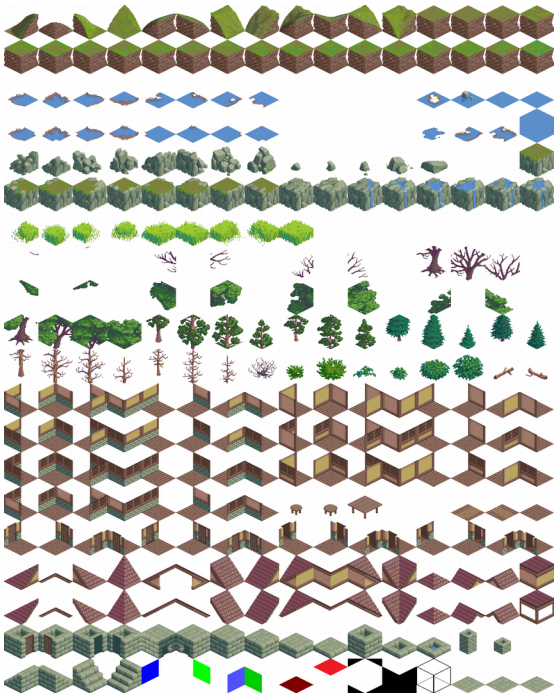
1.3 Conception Logiciel

Présenter ici les packages de votre solution, ainsi que leurs dépendances.

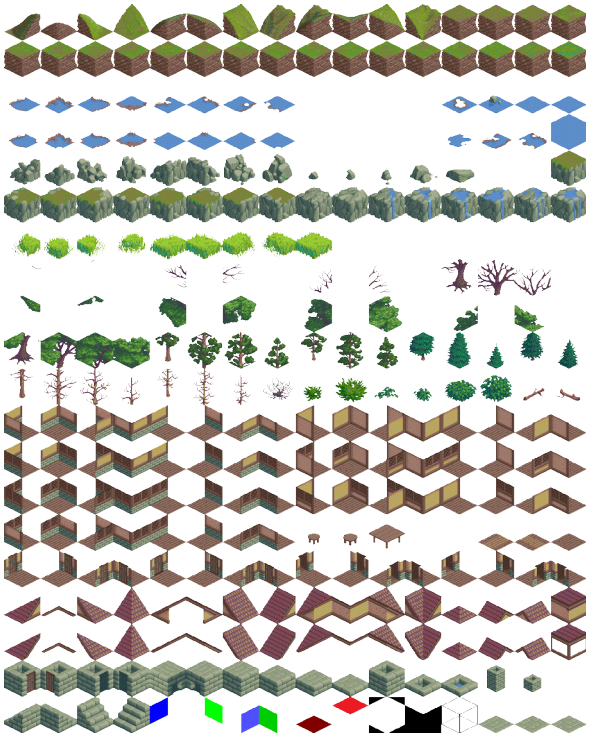
La carte est d'une taille 32x32 pixels, un pixel déterminant une zone.

1.3.1 Tiles pour le terrain :

Version original trouvée sur google image :



Version trasnparente utilisée dans le projet:

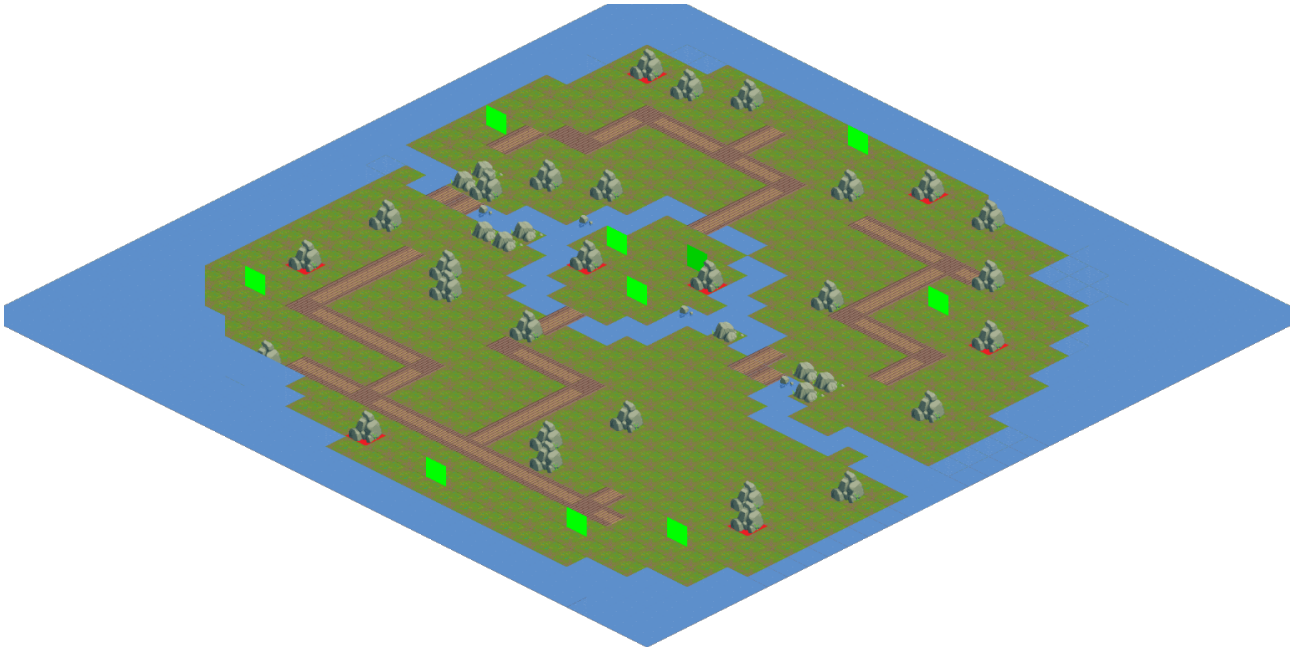


1.3.2 Tiles pour les ressources :

création de tiles sur blender (à confirmer)

1.3.3 Tiles pour les unités :

Voici notre prototype de notre première carte : InrushIsland



-Les carrés verts désignent une zone de champ.
-Les carrés rouges désigent une zone de gisement d’or.

2 Description et conception des états

L'objectif de cette section est une description très fine des états dans le projet. Plusieurs niveaux de descriptions sont attendus. Le premier doit être général, afin que le lecteur puisse comprendre les éléments et principes en jeux. Le niveau suivant est celui de la conception logiciel. Pour ce faire, on présente à la fois un diagramme des classes, ainsi qu'un commentaire détaillé de ce diagramme. Indiquer l'utilisation de patron de conception sera très appriécé. Notez bien que les règles de changement d'état ne sont pas attendues dans cette section, même s'il n'est pas interdit d'illustrer de temps à autre des états par leur possibles changements.

2.1 Description des états

2.2 Conception logiciel

2.3 Conception logiciel : extension pour le rendu

2.4 Conception logiciel : extension pour le moteur de jeu

2.5 Ressources

Illustration 1: Diagramme des classes d'état

3 Rendu : Stratégie et Conception

Présentez ici la stratégie générale que vous comptez suivre pour rendre un état. Cela doit tenir compte des problématiques de synchronisation entre les changements d'états et la vitesse d'affichage à l'écran. Puis, lorsque vous serez rendu à la partie client/serveur, expliquez comment vous aller gérer les problèmes liés à la latence. Après cette description, présentez la conception logicielle. Pour celle-ci, il est fortement recommandé de former une première partie indépendante de toute librairie graphique, puis de présenter d'autres parties qui l'implémente pour une librairie particulière. Enfin, toutes les classes de la première partie doivent avoir pour unique dépendance les classes d'état de la section précédente.

3.1 Stratégie de rendu d'un état

3.2 Conception logiciel

3.3 Conception logiciel : extension pour les animations

3.4 Ressources

3.5 Exemple de rendu

Illustration 2: Diagramme de classes pour le rendu

4 Règles de changement d'états et moteur de jeu

Dans cette section, il faut présenter les événements qui peuvent faire passer d'un état à un autre. Il faut également décrire les aspects lié au temps, comme la chronologie des événements et les aspects de synchronisation. Une fois ceci présenté, on propose une conception logiciel pour pouvoir mettre en œuvre ces règles, autrement dit le moteur de jeu.

4.1 Horloge globale

4.2 Changements extérieurs

4.3 Changements autonomes

4.4 Conception logiciel

4.5 Conception logiciel : extension pour l'IA

4.6 Conception logiciel : extension pour la parallélisation

Illustration 3: Diagrammes des classes pour le moteur de jeu

5 Intelligence Artificielle

Cette section est dédiée aux stratégies et outils développés pour créer un joueur artificiel. Ce robot doit utiliser les mêmes commandes qu'un joueur humain, ie utiliser les mêmes actions/ordres que ceux produit par le clavier ou la souris. Le robot ne doit pas avoir accès à plus information qu'un joueur humain. Comme pour les autres sections, commencez par présenter la stratégie, puis la conception logicielle.

5.1 Stratégies

5.1.1 Intelligence minimale

5.1.2 Intelligence basée sur des heuristiques

5.1.3 Intelligence basée sur les arbres de recherche

5.2 Conception logiciel

5.3 Conception logiciel : extension pour l'IA composée

5.4 Conception logiciel : extension pour IA avancée

5.5 Conception logiciel : extension pour la parallélisation

6 Modularisation

Cette section se concentre sur la répartition des différents modules du jeu dans différents processus. Deux niveaux doivent être considérés. Le premier est la répartition des modules sur différents threads. Notons bien que ce qui est attendu est un parallélisation maximale des traitements: il faut bien démontrer que l'intersection des processus communs ou bloquant est minimale. Le deuxième niveau est la répartition des modules sur différentes machines, via une interface réseau. Dans tous les cas, motivez vos choix, et indiquez également les latences qui en résulte.

6.1 Organisation des modules

6.1.1 Répartition sur différents threads

6.1.2 Répartition sur différentes machines

6.2 Conception logiciel

6.3 Conception logiciel : extension réseau

6.4 Conception logiciel : client Android

Illustration 4: Diagramme de classes pour la modularisation