

BattleCraft







Framework de jeu 2D

Présentation du jeu









BattleCraft est un jeu de type RTS (Real-Time Strategy). L'objectif du jeu est de détruire le château de l'adversaire tout en résistant aux assauts de l'IA.

Au lancement du jeu, il y a deux équipes (Bleu et Rouge) donc une contrôlée par un joueur. Les deux équipes possèdent un Château, ainsi que 4 casernes pour créer des unités. Chaque casernes possèdent 2 types d'actions créer une unité ou améliorer le bâtiment. L'amélioration de la caserne permet une amélioration des unités créés par la suite mais pas des unités déjà créées.



Le jeux dispose de 3 ressources récoltables, le bois (wood), la pierre (rock) et le minerais (ore). Ces ressources s'épuisent au bout de quelques coups mais réapparaissent automatiquement.

Bois (Wood)	Pierre (Rock)	Minerai(Ore)
 Plein	 Plein	 Plein
 Epuisé	 Epuisé	 Epuisé

Pour récolter ces ressources il suffit de créer l'ouvrier correspondant.

 <p>Coût amélioration: Ore(1000)</p>	<p>Caserne pour ouvriers ramassant du bois, permet de créer :</p>  <p>Coût :Wood(10)</p>
 <p>Coût amélioration: Ore(1000)</p>	<p>Caserne pour ouvriers ramassant de la pierre, permet de créer :</p>  <p>Coût :Wood(200)</p>
 <p>Coût amélioration: Ore(1000)</p>	<p>Caserne pour ouvriers ramassant du minerais, permet de créer :</p>  <p>Coût :Rock(100)</p>
 <p>Coût amélioration: Ore(1000)</p>	<p>Caserne pour soldats, permet de créer :</p>  <p>Coût :Rock(100)</p>

Pour afficher les boutons d'amélioration et de création il suffit de cliquer sur une structure. Si les boutons sont grisés cela signifie que les ressources nécessaires sont insuffisantes.

Le bouton  permet d'améliorer la structure, le bouton  permet de créer une unité.

Au dessus de chaque unité une barre bleue indique le cooldown de cet unité (c'est à dire le temps de rechargement avant la prochaine frappe, que ce soit sur une ressource pour un ouvrier ou sur un ennemi pour un soldat).

Pour les soldats une barre de vie est affichée, en vert pour les alliées et en rouge pour les ennemis. Il faut savoir que la vie des soldats remonte peu à peu automatiquement.

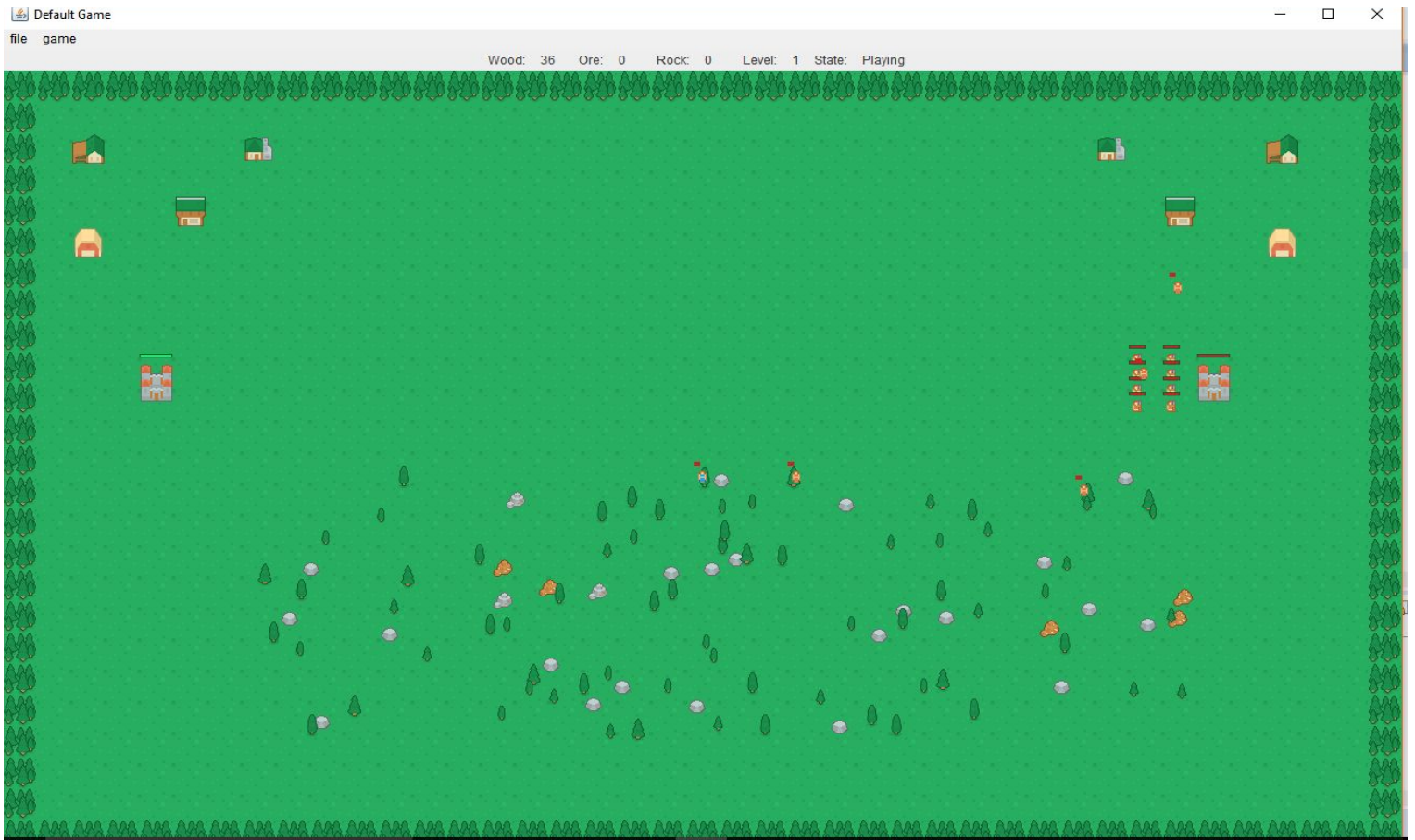
Pour déplacer les unités :

- Un soldat ou un groupe de soldat est contrôlable et sélectionnable par l'utilisateur via une zone de sélection. Une fois sélectionnées le contour de leur barre de vie s'affiche en blanc. Pour déplacer les unités sélectionnées il faut indiquer une position via un clic droit. Un clic gauche désélectionne toutes les unités sélectionnées.
- Les unités de récoltes sont divisées par catégorie de ressource (Bois, Pierre, Minerai) à récolter. Elles ne sont ni contrôlables, ni sélectionnables par le joueur à la création d'une unité, celle-ci ira automatiquement chercher la ressource liée à son métier sans besoin de recevoir un ordre. Les ressources récoltées seront automatiquement ajoutées à celle déjà disponible.

Combat :

- Si un soldat croise un ennemi, ou est placé contre le château adverse, il entre en mode combat. Il inflige automatiquement des dégâts.
- Les ouvriers ne peuvent pas être tués ni blessés.

Astuce : Au début il est recommandé de créer entre 5 et 10 ouvriers pour le bois, et ensuite dès que possible entre 5 et 10 ouvriers pour la pierre, puis de commencer la création des soldats et des ouvriers chargés des minerais.



Architecture du jeu

Package	Description
battleCraft	Contient les classes qui permettent de gérer le jeu et les sous packages correspondent à notre application de jeu 2D
battleCraft.entity	Contient les classes qui créer les entités et celles qui permettent de leur définir une action (Sélectionner, Améliorable, etc,...)
battleCraft.entity.environment	Contient les classes liées aux ressources
battleCraft.entity.structure	Contient les classes liées aux structures
battleCraft.entity.tile	Contient les classes qui remplissent le background
battleCraft.entity.unit	Contient les classes liées aux unités
battleCraft.enums	Contient les différentes énumérations utiliser dans le cadre de l'application
battleCraft.rule	Contient les classes liées aux règles de collisions (MoveBlocker, Overlap,...)
battleCraft.strategy	Contient les différentes stratégies du jeu, e.g. sélection des unités, déplace et cueillette des ouvriers,...
battleEngine	Contient les classes des soldats cf TD
gameframework	Framework du jeu

Classe	Description
GameLevelBC	Initialisation de la configuration du jeu et du GameBoard
GameBC	Créer notre application de jeu avec les menus et l'affichage des informations

EntityFactor	Regroupe les fabriques de toutes les entités en une seule fabrique, permet une généralisation plus simple d'utilisation et cache la création à l'utilisateur
Selectable	Interface, permet de rendre un objet sélectionnable (cf. soldier et barrack)
Upgradable	Interface, permet à un objet d'être amélioré lors de la partie en cours
SpriteStore	Singleton qui stocke les différents sprites à utiliser dans une HashMap, et permet de récupérer les sprites déjà utilisés sans insérer une nouvelle image en mémoire.
MoveBlockers	Action à effectuer lorsqu'un objet croise un objet de type MoveBlocker. Ici action à réaliser quand un soldat arrive sur un château. Définit la condition d'arrêt du jeu dans notre cas $\text{chateauHealth} \leq 0$
OverlapRules	Action à effectuer lorsqu'un objet Overlappable rencontre un autre objet Overlappable. E.g. Quand deux soldats se croisent ils rentrent en mode combat. Où si un ouvrier rencontre une ressource correspond à son travail alors il la récolte
BarrackStrategy	Stratégie pour les casernes, permet d'afficher le menu pour la création d'unité ou l'amélioration de la caserne après un clic
MoveStrategySelect	Stratégie qui permet de sélectionner plusieurs soldats pour faire un groupe de soldats et leur donner des ordres en même temps
MoveStrategyToPoint	Stratégie qui permet de définir une nouvelle destination
WorkerMoveStrategy	Définit une stratégie pour les ouvriers, elle va permettre à un ouvrier de se diriger vers la ressource la plus proche possible correspond à sa ressource attribuer et de la récolter automatiquement.

Choix de conception

Pour la conception de l'application, nous avons souhaité cacher une grande partie des différentes créations des objets (Casernes, Unités, etc,...). De ce fait nous avons utilisés le pattern de la Fabrique Abstraite. La classe EntityFactory implémente l'interface des 4 fabriques (structures, unités, tiles, ressources). A la création elle se charge d'instancier ces 4 fabriques avec l'équipe passées en paramètre, et utilise ensuite ces fabriques pour retourner les objets demandées. Cela permet de réduire le code nécessaires à la création et à l'utilisation de 4 fabriques différentes, et donc de créer facilement des entités appartenant à une équipe.

Les différentes équipes et les différents type de ressources sont représentés par un Enum, ce qui permet de restreindre les types passées en paramètre tout en autorisant un ajout facile d'un type de ressource ou d'une équipe.

Pour la partie qui concerne l'ajout des objets dans le GameBoard, nous avons utilisé la classe LevelManager qui est un singleton. Cela nous permet d'ajouter une entitée à l'univers du jeu (qui est de toute façon unique), avec le drivers et la stratégie correspondante, tout en évitant de "polluer" les différentes classes pour lesquelles il est nécessaire de pouvoir insérer des unités dans l'univer du jeu (casernes par ex.).

L'IA est tout simplement implémentée via une simple classe qui contient le total des ressources actuelles ainsi qu'une liste de toutes les structures qu'elle possède. A chaque ajout d'une ressource (i.e. un ouvrier récolte une ressource, ce qui ajoute le montant récolté au total des ressources à la disposition de l'IA) elle regarde si elle peut construire une unité (avec certaines priorités, comme par ex. un certain nombre d'ouvriers avant les soldats) ou améliorer un bâtiment.

Lors de l'implémentation nous avons essayé de factoriser le plus de code possible pour éviter la duplication de code dans les sous classes pour garder quelques choses de claire et compréhensible.

Problèmes et améliorations

Au niveau des améliorations possibles il y a :

- L'ajout et la création de différents types de soldats, par exemple de Tanks, Cavaliers, etc...
- Ajouter la possibilité d'acheter une amélioration pour son armée comme une épée ou un bouclier pour booster les stats de son équipe,
- Utiliser des sprites animés *.gif pour avoir un meilleur rendu des combats au lieu de simple image.png
- Meilleur gestion de l'IA (e.g. quand toutes les ressources d'un ouvrier sont prises il

s'arrête)

- Gérer la création de structures
- Génération de la map. Pour les ressources par ex. utiliser le bruit de Perlin pour générer automatiquement les ressources sur la map.
- Implémenter les méthodes de sauvegarder, charger, pause et reprendre.
- Ajouter des niveaux de difficultés.

Pour les améliorations certaines sont faciles à implémenter rapidement d'autres demandent un peu plus de temps par exemple pour l'ajout d'un type de soldat il suffit de définir une autre classe et créer ce type de soldat facilement grâce au framework de création d'une armée fourni. En suivant le même principe il est également simple d'ajouter une amélioration à un soldat.

Pour la gestion de l'IA il serait possible d'ajouter d'autres comportements, les comportements actuels des soldats étaient actuellement au nombre de 2.

De même, il est possible de générer toute la carte automatiquement, même si actuellement la position des ressources est déjà aléatoire.

Pour les problèmes, il n'y a pas de problèmes majeur rencontrés lors de l'exécution du programme.

Mais on peut noter le fait que nos unités peuvent se stacker à un même point. Et lorsque la partie est fini il est impossible de relancer une partie.