

Faculté des sciences

Département d’informatique

IFT630 : Processus concurrents et parallélisme

Projet

**Jeu de simulation de chaîne de production industrielle**

Alexandre Blouin 13 094 634

Jérémie Coulombe 13 061 991

15 avril 2016

Table des matières

[Présentation du projet 3](#_Toc448434003)

[Idée de départ 3](#_Toc448434004)

[Technologies utilisées 3](#_Toc448434005)

[Utilisation du parallélisme dans le projet 4](#_Toc448434006)

[Résultat final 4](#_Toc448434007)

[Déroulement du jeu 4](#_Toc448434008)

[Ajouts possibles 5](#_Toc448434009)

[Images 6](#_Toc448434010)

# Présentation du projet

## Idée de départ

Pour ce projet, nous avons décidé de faire un jeu qui permettrait de visualiser certains principes de base de la programmation multi-fils. Pour y arriver, nous avons eu l’idée d’un jeu de simulation de chaîne de montage en millieu industriel. Notre « usine » est ainsi composée de plusieurs « ateliers » dans lesquels travaillent des « travailleurs ». Les travailleurs de chaque atelier combinent ou raffinent des ressources que l’atelier reçoit en entrée d’un fournisseur de ressources ou d’un convoyeur qui transporte les ressources d’un atelier à l’autre. Le résultat est ensuite envoyé en sortie vers un autre atelier. Tout ajout à l’usine a un coût, et les produits fabriqués par l’usine entraînent des profits différents en fonction de leur nature.

Il est à noter que notre idée initiale de projet était complètement différente : nous souhaitions essayer la nouvelle librairie graphique 3D Vulkan, qui met de l’avant le parallélisme. Par contre, elle n’est pas encore compatible avec un grand nombre de cartes graphiques, dont celles des laboratoires informatiques du département ainsi que celles de nos ordinateurs personnels, ce qui nous a poussé à trouver une autre idée.

## Technologies utilisées

Notre projet est un jeu en 2D et fait en C++ avec Visual Studio 2015. Nous avons utilisé le petit moteur de jeu 2D Allegro, que nous avons adapté et simplifié via des classes « wrapper » C++. Entre autres, nous avons exposé certaines commandes de rendu ainsi que les systèmes de gestion de caméra, de clavier et de souris. Pour ce qui en est du parallélisme, nous avons tout simplement utilisé les « threads » de la STL de C++.

# Utilisation du parallélisme dans le projet

Dans le projet, la programmation multi-fils a été utilisée de plusieurs manières. Tout d’abord, chaque travailleur exécute son code sur son propre fil d’exécution, afin de travailler simultanément. Ils convertissent les ressources d’entrée d’un atelier en ressource ou produit de sortie. Nous avons représenté l’accumulation de ces ressources en entrée et sortie par des piles, dont les travailleurs demande l’accès pour retirer ou ajouter une ressource à la fois. On a ainsi une forme du problème du producteur/consommateur, ou le producteur est le tapis roulant qui achemine la ressource vers l’atelier (ce dernier fonctionne également sur son propre fil d’exécution) et le consommateur est un travailleur. Il a donc été nécessaire d’utiliser l’exclusion mutuelle ainsi qu’une variable de condition pour gérer l’accès à ces piles.

De plus, notre projet utilise un « renderer » (moteur de rendu), lequel s’occupe entre autres d’afficher tout ce qui se trouve à l’écran. Ce dernier fonctionne sur son propre fil d’exécution, ce qui permet de le séparer de la boucle principale du jeu. Dans cette boucle principale, nous ne faisons pratiquement qu’envoyer des commandes de rendu au « renderer », lesquelles sont exécutées par celui-ci en même temps qu’il accumule les commands pour la prochaine image à dessiner.

# Résultat final

## Déroulement du jeu

Au final, voici ce qu’il est possible de faire dans notre jeu : En utilisant l’interface utilisateur, qui est composée d’un menu contenant trois boutons, on peut premièrement placer des ateliers sur la grille de l’usine, en choisissant le côté de l’atelier qui contiendra une pile de ressources de sortie. En plaçant les ateliers côte à côte, ils deviennent reliés par un tapis roulant qui transportera des ressources. Chaque atelier apparaît d’abord avec un seul travailleur, mais il est possible d’en ajouter d’autres pour accélérer le traitement des ressources, ce qui est surtout utile lorsqu’il y a plusieurs entrées à l’atelier. Il est aussi possible d’ajouter un fournisseur (ou générateur) de ressources à un atelier qui a un de ses quatre côté libre, ce que l’on peut interpréter comme un camion qui emmènerait des ressources premières à l’usine. En bas à droite de l’écran est affichée l’argent que le joueur possède, qui change lorsqu’il dépense pour produire des objets ou lorsque les produits se rendent aux piles de sorties (carrés rouges).

On peut déplacer la caméra en maintenant le clic droit de la souris enfoncé, puis il est possible de faire un zoom avec la roulette de la souris, sauf quand on utilise celle-ci pour décider de la position d’une pile de sortie ou d’un fournisseur de ressources qu’on est en train de créer.

## Ajouts possibles

Puisque nous n’avons pas eu l’occasion de mettre autant de temps qu’on aurait souhaité sur le projet, il contient peu de fonctionnalités. Toutefois, le concept de base est solide, et nous l’avons programmé d’une manière qui permettrait aisément de rajouter de nouvelles fonctionnalités. Par exemple, rajouter un nouveau type de salle d’usine serait possible, qu’on pourrait relier aux ateliers et lui assigner un bouton de création dans l’interface. Supprimer ou modifier un atelier aurait aussi été une action utile. Nous aurions également souhaité avoir le temps de concevoir et d’utiliser de vraies textures 2D ainsi que des animations, lesquelles auraient pu être gérées sur leurs propres fils d’exécution. Ajouter du texte qui explique les contrôles et les actions possibles serait aussi un ajout potentiel.

## Images



