Projet d'algorithmique et d'image n°1

Cours de Synthèse d'Image I

— IMAC première année —

Imac Tower Defense

Ce projet a pour but :

- De vous familiariser avec les structures d'arbres.
- De vous faire utiliser les fonctions de dessin 2D OpenGL.
- De vous faire manipuler des images.

Nombre de personnes par projet : 2 ou 3
Date de rendu : lundi 20 mai

---

1 Introduction

Ce projet a pour but de vous faire réaliser une application SDL permettant de visualiser une carte composée d'un chemin où des monstres se déplacent d'une entrée vers une sortie. L'utilisateur pourra placer des tours pour empêcher ces derniers d'atteindre une sortie. Ce projet sera rendu pour le lundi 20 mai et accompagné d'un rapport (cf. travail à fournir). Vous enverrez donc avant cette date, le rapport, le Makefile, les fichiers .c et .h ainsi que l'exécutable à vos chargés de td/cm, le tout taré et zippé. Les projets seront réalisés en binômes, la masse de travail étant trop importante pour une seule personne. Les trinômes sont acceptés mais devront réaliser du travail supplémentaire. Enfin votre projet devra fonctionner sur les machines linux des salles de TD. Aucun code ni exécutable fonctionnant exclusivement sous Windows ou Mac ne sera accepté.

2 Présentation du projet

Votre groupe est chargé, au sein d'une entreprise de jeu vidéo, de créer un prototype de logiciel permettant, d'une part, le chargement d'une carte, et d'autre part, la construction sur cette carte de différents batîments pour empêcher des vagues de monstres tentant de trouver la sortie d'atteindre cette dernière, le tout dans un jeu nommé `Imac Tower Defense`.

2.1 Travail à fournir

Pour le projet :

- Réalisation d'un Makefile, et d'une structure ordonnée de projet (cf. section 2.2)

Initialisation d'une carte (cf. section 3) :

- Chargement d'une carte.

Création des différents élements (cf. section 4) :

- Sprites de bâtiments, de tours, de monstres.
- Création des structures de bâtiments, de tours, de monstres.

ITD : le jeu (cf. section 5) :

- Ajout, suppression et édition de bâtiments et de tours.
- Déplacement des monstres.
- Gestion des actions des tours et bâtiments.
- Gestion du temps.

Rapport contenant notament :

- Introduction.
- Présentation de l'application.
- Description globale de l'architecture de votre application (justification des découpages, etc.).
- Description des structures de données utilisées.
- Description détaillée des fonctionnalités de l'application interactive avec notamment les règles du jeu (Guide utilisateur).
- "Résultats" obtenus par votre application (capture d'écran, temps de calcul...).
- Conclusion

2.2 Structure de votre programme

Dans ce projet complexe, il est hors de question de n'utiliser qu'un seul fichier contenant toutes les fonctions nécessaires à l'application ! Il vous faut donc séparer les traitements en différents fichiers `.c` et `.h`. La découpe des fichiers est laissée à votre appréciation mais doit être logique. Globalement, le projet étant scindé en deux parties, il serait logique que la partition des fichiers en tienne compte. Afin de compiler le projet, un Makefile devra être réalisé. **Note importante :** Tout `#include "unfichier.c"` ou tout rendu de projet sans Makefile entrainera un 0 !

Le programme sera donné sous la forme d'un exécutable nommé `itd`. Il prendra 1 argument : le fichier `.itd` décrivant la carte de jeu.

Enfin votre projet sera mis dans un repertoire nommé aux noms des 2 binômes (ou des 3 trinômes le cas échéant). Dans celui ci vous suivrez la structure suivante :

```
Nom1Nom2/
  \-- bin/
  \-- include/
  \-- src/
  \-- lib/
       \-- include
       \-- src
       Makefile
  \-- data/
  \-- images/
  \-- doc/
  \-- Makefile
```

Le répertoire `bin` contient l'exécutable `itd`. Le répertoire `include` contient les fichiers d'entête `.h` de vos programmes tandis que `src` contient les fichiers sources `.c`. Le répertoire `lib` (optionnel) contiendra les fichiers `.a` ou `.so` de vos bibliothèques ainsi que tout le nécessaire pour les compiler : un Makefile, un répertoire `include` contenant les `.h` et un répertoire `src` contenant les `.c`. Le répertoire `data` contient les fichiers `.itd` et `images` contient tous les `.ppm`. Le répertoire `doc` contiendra toute la documentation, dont votre rapport. Enfin un `Makefile` permettera de compiler `itd`.

Trinôme uniquement : Les trinômes devront permettre à l'exécutable de ne prendre aucun argument. Dans ce cas, l'utilisateur pourra choisir une carte parmi celles présentes dans le répertoire `data`.

3 Les cartes Imac Tower Defense

Les éléments constitutifs d'une carte sont regroupés dans un fichier de configuration simple de suffixe `.itd`.

3.1 Fichier de description .itd

Ce fichier indique les différents paramètres nécessaires pour charger une carte dans le jeu. C'est un simple fichier texte au format suivant :

```
@ITD 1
carte fichier.ppm
chemin 255 255 255
noeud 0 0 0
construct 255 200 80
in 0 200 0
out 200 0 0
5
10 20
454 103
300 103
200 103
200 206
```

La première ligne est constitué d'un code (pour ce projet ce sera toujours `@ITD`) indiquant qu'il s'agit d'un fichier de description de carte pour ITD suivi du numéro de version de ce fichier de description (ici ce sera `1`). Ensuite il y a plusieurs lignes, constituées de deux éléments : un mot clé et une valeur.

Mot clé	Valeur	Type de la valeur
`carte`	Nom, **ne contenant pas d'espace**, de fichier image au format `ppm` représentant la carte proprement dit (cf. section 3.2)	chaine de caractères
`chemin`	Couleur clé des chemins dans l'image	Triplet R,V,B (0-255)
`noeud`	Couleur clé des noeuds	Triplet R,V,B (0-255)
`construct`	Couleur clé des zones 'constructibles'	Triplet R,V,B (0-255)
`in`	Couleur clé de la zone d'entrée	Triplet R,V,B (0-255)
`out`	Couleur clé de la zone de sortie	Triplet R,V,B (0-255)

Figure 1: Description des parametres du fichier

Le chemin emprunté par les monstres est défini par une suite de segments, eux-même définis par une suite de nœuds. Le fichier contient, sur une ligne, un entier indiquant le nombre de nœuds du chemin, suivi d'autant de lignes décrivant la position de ce noeud dans l'image. Le premier nœud représente l'entrée et le dernier représente la sortie.

3.2 Fichier image représentant la carte

Pour représenter la carte, nous utilisons une image en couleur au format `ppm` . Sur cette images, on associe certaines couleurs à des zones particulières :

- couleur : 120 120 120 → zones constructibles

- autres couleurs → zones non-constructibles

Sur la carte affichée à l'écran, la couleur de la zone constructible sera remplacée par celle spécifiée dans le fichier `.itd` . Les autres couleurs de l'image seront reportées sur la carte affichée à l'écran.

3.3 Validation d'une carte

Plusieurs conditions doivent être remplie pour qu'une carte soit valide pour le jeu ITD. Votre application devra être capable, lors du chargement d'une carte ITD, de vérifier si une telle carte est valide ou non.

Voici les différentes exigences :

- Fichier `.itd` au format correct : Présence sur la première ligne du bon code, ligne de commentaire présente, chacun des 8 paramètres existe et a une valeur "correcte", nombre de noeud correspondant au nombre de lignes restantes.
- Bonne validité des noeuds : Les coordonnées de chaque noeud dans le fichier .itf doit correspondre à un pixel de l'image.

4 Les éléments du jeu et leur représentation informatique

4.1 Résumé des règles et liste des éléments du jeu

Le jeu se déroule comme suit. Après le chargement de la carte, le joueur dispose d'une certaine somme d'argent initiale pour construire, sur les zones constructibles de la carte, des tours de défense. Ces tours sont là pour détruire des monstres, arrivant par vagues de 10, et qui cherchent à atteindre la zone de sortie. Si un monstre parvient à la zone de sortie, la partie est perdue. Si, au bout de 20 vagues de monstres, aucun n'est parvenu à la zone de sortie, alors la partie est gagné.

Chaque monstre détruit par les tours rapporte de l'argent au joueur. Cet argent permet au joueur de construire des nouvelles tours ... L'ensemble des éléments du jeu est donc :

- La carte décrite section 3 avec ses chemins
- Les monstres
- Les tours

4.2 Les chemins

Ils sont spécifiés par les nœds définis dans le fichier de configuration. Sur la carte, il faut les faire apparaitre en dessinant des segments de droites openGL, avec la couleur spécifiée dans le fichier de configuration.

4.3 Les monstres

Les monstres disposent de points de vie et d'une résistance propre à chacun des types de tour (voir section suivante). Les monstres arrivent par groupes de dix. À chaque nouvelle vague, les monstres voient leur point de vie et/ou leur résistance augmenter. C'est à vous de définir cette augmentation. De même, chaque monstre tué rapporte de l'argent au joueur. Initialement, les monstres de la première vague rapporte 5 unités d'argent. À chaque nouvelle vague, les monstres rapporteront plus d'argent.

Vous devrez implémenter au moins 2 type différents de monstres ayant des caractéristiques distinctes notamment de vitesse.

Graphiquement, les monstres seront représentés dans le jeu par ce qu'on appelle des sprites. Les sprites sont des images rectangulaires possèdant bien souvent de la transparence. Affichés sur une image de fond, ils permettent de représenter un élément (ici les monstres ou les tours) sur un fond (ici la carte).

4.4 Les tours

Le joueur peut créer des tours de défense qui tireront sur les monstres. Les tours ont trois caractéristiques : la puissance, la portée et la cadence. La puissance indique les dégats effectués par la tour. La portée indique en pixels la distance à laquelle la tour peut tirer. Enfin, la cadence indique en nombre de dizième de secondes l'intervalle entre deux tirs.

Les tours sont de quatre types :

- Les tours de type rocket : Ces tours infligent beaucoup de dégats mais ont une cadence de feu faible.
- Les tours de type laser : Elles tirent très rapidement mais ont une faible portée et occasionne des dommages moyens.
- Les tours de type mitraillettes : Elles occasionnent peu de dégat, ont une portée très limité mais une bonne cadence de tir et tirent sur tous les monstres à leur portée.
- Les tours de type hybrides : Elles ont une très bonne portée et une bonne cadence de tir mais occasionnent peu de dégats.

Les différentes paramètres de ces différentes tours (dégats, portée, cadence) sont laissés à votre appréciation. Chaque tour a également un coût d'achat pour pouvoir être placée dans la carte.

Les tours sont représentées graphiquement par des sprites d'au moins vingt pixels (laissé à votre appréciation). De plus les tours ne peuvent être construites que sur une zone constructible et pas sur une autre tour.

5 Le jeu ITD

5.1 Interface générale

L'application sera réalisée à l'aide d'OpenGL et de la SDL. Votre application contiendra au minimum une fenêtre d'affichage contenant la carte et une fenêtre interface SDL contenant au moins un bouton permettant de quitter l'application. De plus, une option dans les arguments du programme, ou une action de l'utilisateur (appui sur un bouton par exemple) affichera une aide pour l'utilisation du jeu.

5.2 Déroulement du jeu

L'application commence tout d'abord par charger la carte i.e. le fichier `.itd` (section 3) et vérifier que celle-ci est valide (voir section 3.3). Une fois la carte chargée le jeu peut commencer après une action de l'utilisateur.

Le jeu fonctionne en "continu" c'est à dire que c'est le temps qui rythme la succession des évènements. L'unité de temps est le dizième de seconde. Durant un dixième de seconde, les tours peuvent tirer au mieux une fois et les monstres peuvent se déplacer d'un pixel. En fait chaque tour tire toutes les n dizièmes de secondes où n est la cadence de tir de la tour. Les tours tirent sur le monstre le plus proche d'elle.

L'utilisateur peut par contre construire une tour quand il le souhaite. Lorsqu'il place une tour, l'application doit veiller à ce que ce placement soit valide (voir section 4.4. Le joueur doit pouvoir également mettre en pause l'application.

Votre interface devra permettre de :

- Voir la carte, les monstres et les tours
- Indiquer l'argent restant disponible
- Sélectionner une tour à construire
- Placer cette tour dans la carte
- Déplacer les monstres
- Sélectionner une tour déjà construite et afficher ces propriétés : puissance, cadence, portée.

Si un monstre arrive à une zone de sortie, le jeu est perdu. Si 20 vagues de monstres ont été éliminées alors le jeu est gagné.

6 Travail supplémentaire pour les trinômes

En plus de la mention de la section 2.2, vous devrez réaliser les fonctionnalités suivantes :

- Format de fichier : Votre application devra pouvoir lire un fichier `.itd` au format 1 ou 2 (représenté dans la première ligne par `@ITD 2`). Le format de fichier `.itd` version 2 inclus les couples mot clé valeur suivants :

Mot clé	Valeur	Type de la valeur
`powerX`	Puissance pour les tours de type X (X à remplacer par R (rocket) L (laser), M (mitraillette) et H (hybride))	entier ≥ 0
`rateX`	Cadence pour les tours de type X (R,L,M,H)	entier ≥ 0
`rangeX`	Portée pour les tours de type X (R,L,M,H)	entier ≥ 0
`costX`	Coût d'achat des tours de type X (R,L,M,H)	entier ≥ 0

- Zones non constructibles : Afin d'embellir la carte, tout pixel qui ne soit pas de la couleur d'une zone constructible, d'un chemin ou d'une zone d'entrée sortie est considéré comme une zone non constructible. Ainsi, lors de la création d'une tour ou d'un batiment, l'application devra vérifier, **en plus des intersections avec chemin et autre tour/batiment**, que les pixels recouvrant le sprite (disque ou carré) ne sont pas de type zone non constructible.

7 Des bonus

Tout outils, spécifications, fonctionnalités supplémentaires seront récompensés. On peut penser notamment à :

- Différents types de terrain pour les chemins ralentissant ou accélérant les monstres.
- Visualisation des tirs des tours sur les monstres.
- Sélection des monstres et affichage de leur propriétés.
- ...

Néanmoins, si une nouvelle fonctionnalité modifiait même de manière légère les spécifications, notamment le format des fichiers `itd`, présentes dans cette description de projet alors cette nouvelle fonctionnalité devra être validée par vos chargés de td / cm. Elle devra aussi être indiquée clairement dans le rapport.

8 Conseils et remarques

- Ce sujet de projet constitue un cahier des charges de l'application. Tout changement à propos des spécifications du projet doit être validée par vos chargés de td / cm.
- Le temps qui vous est imparti n'est pas de trop pour réaliser l'application. N'attendez pas le dernier mois pour commencer à coder.

- Il est très important que vous réfléchissiez **avant de commencer à coder** aux principaux modules, algorithmes et aux principales structures de données que vous utiliserez pour votre application . Il faut également que vous vous répartissiez le travail et que vous déterminiez les tâches à réaliser en priorité.
- Ne rédigez pas le rapport à la dernière minute sinon il sera baclé et cela se sent toujours.
- Le projet est à faire par binôme. Il est **impératif** que chacun d'entre vous travaille sur une partie et non pas tous "en même temps" (plusieurs qui regarde un travailler). sinon vous n'aurez pas le temps de tout faire. C'est encore plus vrai pour les trinômes.
- N'oubliez pas de **tester** votre application à chaque spécification implémentée. Il est impensable de tout coder puis de tout vérifier après. Pour les tests, confectionnez vous tout d'abord de petites cartes (taille 5 par 5 par exemple) avec un chemin extrêmement simple. Si cela marche vous pouvez passer à plus gros ou plus complexe.
- Vos chargés de td et cm sont là pour vous aider. Si vous ne comprenez pas un algorithme ou avez des difficultés sur un point, n'attendez pas la soutenance pour nous en parler.