Programmation avancée en C :

Présentation du projet 2023–2024

Licence informatique 3^{e} année

Université Gustave Eiffel

Tower defense, un pseudo-démo

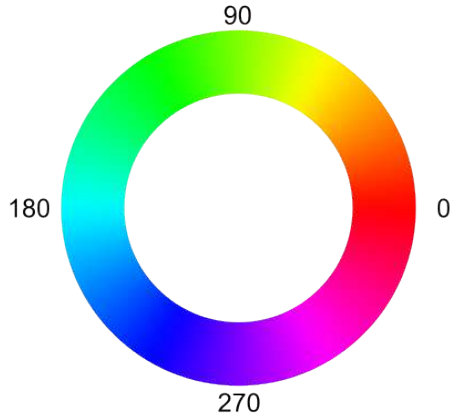
Capture d'écran de *Gemcraft Chapter One : The Forgotten*

Les ingrédients

- Vagues de monstres
  - Tous les 35 secondes
  - Quadre type : normal, agile, foule, boss
  - Teinte aléatoire pour chaque monstre
- Gemmes et tours
  - Teinte, qui influence le dégât généré par un tir
  - Deux types : pure et mixte
  - Gemmes pures avec des effets spéciaux d'élément
  - Gemmes mixtes avec un dégât de base doublé
  - Placée dans les tours pour pouvoir tirer
- Déroulement du jeu
  - Mouvement continu, avec framerate à contrôler
  - Se défendre contre les monstres
  - Bien gérer le mana et les gemmes

Teintes et éléments

Les monstres et les gemmes viennent avec des teintes (0–359).



Plus les deux teintes sont proches, moins le dégât est généré.

Trois types de gemme pure :

- Rouge : élément *Pyro*, effet d'éclaboussure
- Verte : élément *Dendro*, effet de parasite
- Bleue : élément *Hydro*, effet de ralentissement

Détaillé dans le sujet !

Mana et actions

Toute action du joueur consomme du mana stocké dans une réserve.

La capacité de la réserve est déterminée par son niveau.

Le joueur peut :

- Augmenter le niveau de sa réserve,
- Bâtir une tour sur une case vide,
- Générer une gemme pure,
- Fusionner deux gemmes en une de niveau plus grand.

La gestion de mana est cruciale pour la survie !

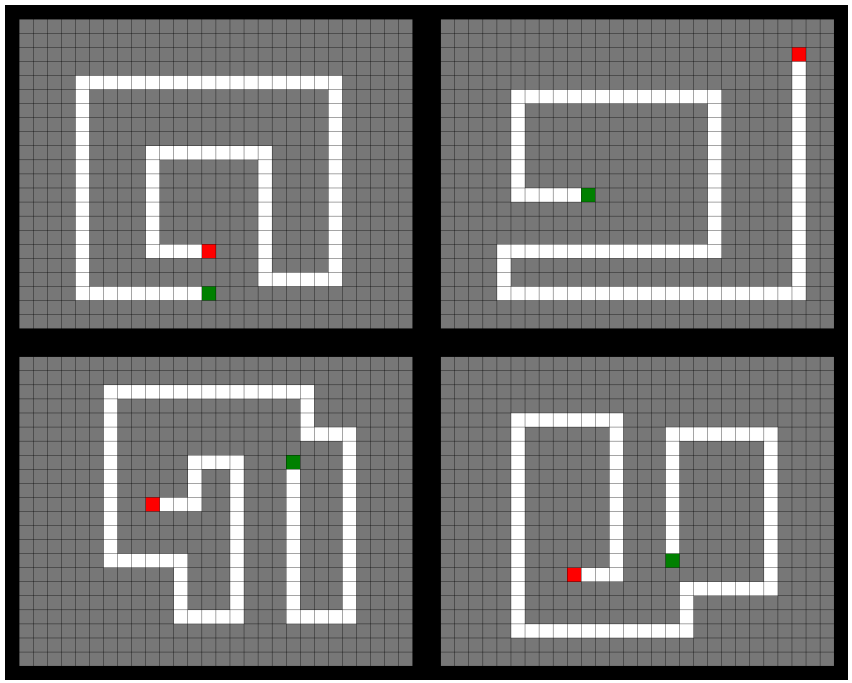
Représentation du terrain

Le terrain est de taille fixé (28×22), avec :

- Le nid des monstres (départ des vagues)
- Le camp du joueur (à défendre)
- Le chemin (généré aléatoirement)

Il est conseillé de représenter les cases avec un `enum`.

L'algorithme de génération est donné dans le sujet.



Interface graphique

- Le terrain représentée sur une grille
- Interface des actions du joueur
- Réserve de mana et stockage de gemmes
- Monstres avec barre de HP
- Tirs animés
- ...

Avec libmlv !

Contrôle de framerate

Il faut exactement 1 mise-à-jour tous les $1/60$ secondes.

- Mesurer le temps t en seconde de chaque mise-à-jour.
- Si $t < 1/60$, alors faire une pause de $1/60 - t$ secondes.

Avantage de framerate contrôlé :

- Fluidité et cohérence de vitesse assurée
- Réduction de consommation de ressource

Un pseudo-code est donné dans le sujet.

Modularisation

Plusieurs choses à gérer !

- Graphique
- Gestion des tours
- Gestion des gemmes
- Gestion des vagues de monstre
- Gestion des tirs (dégât, calcul de position, ...)
- Génération du terrain
- ... potentiellement les autres (ou vous subdivisez encore)

Makefile, et modularisation !

Condition de développement

Le projet sera effectué **en binôme**.

- Collaboration avec **Git**
- Formation de binôme libre **dans le même groupe de TP**
- **Il faut signaler toute anomalie**

Le travail demandé :

- Un code qui roule (bah oui !) et bien organisé
- Un Makefile qui marche
- Une documentation adéquate et des bons commentaires
- Un rapport en PDF
- Un fichier `log_dev` de l'historique de commits

Tout dans fichier zip !

Questions ?