

Gurnaud Jade

Turan Baturay

Kruzic Charlotte

NOM\_DU\_PROJET

Jeu d’infiltration stratégique

Projet de 3ème année de licence informatique

Encadré par Julien Bernard, maître de conférences

Université de Franche-Comté

2022-2023

Table des matières

[I. Introduction 3](#_Toc123742920)

[II. Modélisation 3](#_Toc123742921)

[Les règles du jeu (implémentées ou non) 3](#_Toc123742922)

[Structures et classes 3](#_Toc123742923)

[Physique du jeu 5](#_Toc123742924)

[III. Implémentation 5](#_Toc123742925)

[Initialisation 5](#_Toc123742926)

[Principe de la boucle de jeu 5](#_Toc123742927)

[Conditions de fin 5](#_Toc123742928)

[IV. Conclusion 5](#_Toc123742929)

[Gestion de projet 5](#_Toc123742930)

[Idées non implémentées 6](#_Toc123742931)

# Introduction

L’objectif de ce projet était de réaliser un jeu vidéo de type infiltration stratégique, en C++, à l’aide de la bibliothèque Gamedev Framework (gf). Le projet était encadré par Julien Bernard, maître de conférences à l’Université de Franche-Comté.

Un jeu d’infiltration est un jeu vidéo dans lequel le joueur doit tenter de s'infiltrer dans un lieu, en toute discrétion, en évitant d’être repéré par des ennemis, pour réaliser une mission. Pour cela, il dispose généralement de gadgets permettant de l’aider dans sa tâche. En général, cela implique une progression lente et réfléchie.

Nous avons créé NOM\_DU\_JEU, un jeu d’infiltration se déroulant dans un musée, dans lequel le joueur incarne un voleur, ayant comme but de dérober un OBJET, sans se faire repérer.

# Modélisation

## Les règles du jeu (implémentées ou non)

NOM\_DU\_JEU est un jeu se jouant en solitaire.

Le joueur commence au point de départ (ELEMENT). Grâce aux touches directionnelles, il peut se déplacer dans le musée. Il doit d’abord récupérer l’OBJECT, puis rejoindre le point d’arrivée (ELEMENT). Tout cela sans se faire repérer par les différents gardes du musée, qui réalise des rondes.

Ajouter image jeu prototype

Les gadgets / pouvoirs du joueur

Si le joueur sort du musée sans avoir récupéré l’OBJET ou s’il se fait attraper par un des gardes, alors il a perdu. Si le joueur sort avec l’OBJET, il a gagné.

## Structures et classes

### Game

La classe Game permet de représenter l’ensemble du jeu.

Lors de son initialisation, la fenêtre de jeu, un joueur et un niveau sont créés.

Elle contient aussi une boucle de jeu permettant de mettre à jour et d’afficher les différents éléments du jeu.

La fenêtre de jeu peut être fermée en cliquant sur le bouton avec la croix en haut à droite. Lorsque celle-ci est fermée, le jeu est supprimé.

### Level

La classe Level permet de représenter un niveau du jeu.

Un niveau à une largeur et une hauteur, il possède un début et une fin, des murs, des gardes et un joueur.

Lors de sa création, le niveau ne possède pas de murs, il faut les ajouter par la suite.

### Wall

La classe Wall permet de représenter les murs.

Les murs sont représentés par les carrés, ils ont une position, une couleur et un type.

Ils ne changent pas au cours d’un niveau, ils restent fixes.

### Player

La classe Player permet de représenter un joueur du jeu.

Un joueur possède une position, une vélocité, une vitesse, une couleur et est représenté par un carré.

La position du joueur, ainsi que sa vitesse changent au cours du jeu. Pour le déplacer, l’utilisateur doit utiliser les touches directionnelles.

Le joueur reste le même tout au long de la partie.

### Guard

La classe Guard permet de représenter un garde d’un niveau.

Un garde possède une position, une vélocité, une vitesse, une couleur et est représenté par un carré.

Il possède aussi des rectangles de vision et une routine prédéfinie, lui permettant de surveiller le musée.

### Main

La fonction main permet de lancer NOM\_DU\_JEU, en initialisant une partie.

## Physique du jeu

L’ensemble des collisions de NOM\_DU\_JEU étant des collisions rectangle-rectangle, nous avons choisi de ne pas utiliser de moteurs physiques et de gérer les collisions grâce à la bibliothèque gf.

# Implémentation

Comment sont implémentés les trucs de la partie 1

## Initialisation

Chargement du niveau

Les classes / entités

## Principe de la boucle de jeu

## Conditions de fin

# Conclusion

## Gestion de projet

Nous avons utilisé différents outils pour nous organiser lors de ce projet.

Pour la partie communication, nous avons créé un groupe privé sur Discord. Cela nous a permis d’échanger nos idées, de poser des questions et de se faire aider lors des blocages. Nous avons aussi fait des appels, afin de suivre les avancées de chacun.

Pour la partie programmation, nous avons créé un dépôt sur GitHub. Celui-ci nous a permis de sauvegarder notre projet et de pouvoir nous partager le code facilement. Nous avons créé différentes branches afin de pouvoir merger et gérer les conflits plus facilement.

* Master : version qui fonctionne ? (quand on aura le prototype C++)
* Development : évolution des fonctionnalités (ensuite sur master quand c’est bon)

Le cahier des charges (validé ou non, si non pourquoi ?)

Paragraphe 'Solutions trouvées'

## Idées non implémentées

Estimation de temps requis, implems possibles

A faire :

* Ajouter liens en bas de page :
  + Gamedev framework (p1 + lien sujet projet)
  + Discord (conclusion)
  + GitHub (conclusion)

Idées :

* Créer un logo de jeu
* Nom du jeu NOM\_DU\_JEU : Musée + objet à voler
* Ajouter ELEMENT : dire par quoi est représenté (capture écran du jeu de base)
* Ajouter OBJET : nom objet
* Diagramme organisation des différentes classes entre elles
* Game faire singleton ?
* Faire un calendrier
* Remplacer player par voleur pour pas s’embrouiller entre le joueur physique et virtuelle