

Sommaire

Table des matières

[Introduction 2](#_Toc155075007)

[En quoi consiste le projet 2](#_Toc155075008)

[Comment se joue le Tower Defense 2](#_Toc155075009)

[Outils utilisés 3](#_Toc155075010)

[Plateforme collaborative 3](#_Toc155075011)

[L’utilisation de l’IA 3](#_Toc155075012)

[Choix de l’IDE 3](#_Toc155075013)

[Contraintes 3](#_Toc155075014)

[Utilisation de Java SwingFX 3](#_Toc155075015)

[Architecture en MV 3](#_Toc155075016)

[Conception 4](#_Toc155075017)

[Utilisation du MVC et explication de son principe 4](#_Toc155075018)

[Diagramme des classes (demandé par le sujet) 4](#_Toc155075019)

[Implémentations 5](#_Toc155075020)

[Gestion de la map 5](#_Toc155075021)

[Gestion des ennemis et des projectiles 5](#_Toc155075022)

[Gestion des vagues 5](#_Toc155075023)

[Test 5](#_Toc155075024)

[Observation et Améliorations possibles 5](#_Toc155075025)

[Objectifs initiaux 5](#_Toc155075026)

[Améliorations possibles 5](#_Toc155075027)

[Conclusion 5](#_Toc155075028)

# Introduction

## En quoi consiste le projet

L’objectif de ce projet est de créer un Tower Defense avec comme seule base, les connaissances acquises lors du premier semestre de la deuxième année de licence informatique en Programmation Orientée Objet. L’état final de ce projet témoignera ainsi de la compréhension des principes fondamentaux (l’héritage, le MVC, …) des membres du groupes.

## Comment se joue le Tower Defense

Dans ce projet, le joueur pourra éditer sa propre map et ainsi son propre chemin\* et placer le point d’apparition des monstres (le Spawn) et la tour (la Target). Une fois la map finie, les monstres suivront le chemin dessiné dans le but d’atteindre la tour. L’objectif du joueur sera de neutraliser les monstres en achetant des tours de défense qu’il devra placer dans des endroits stratégiques.

\*Le chemin ne doit pas être coupé entre le Spawn et la Target, le jeu perd son intérêt si les monstres ne peuvent pas atteindre leur cible.

# Outils utilisés

## Plateforme collaborative

Github a été utilisé comme plateforme de codage collaborative, discord a été utilisé comme moyen de communication.

Parsec a été également utilisé occasionnellement pour des sessions de codage en simultané.

## L’utilisation de l’IA

L’aide de l’Intelligence Artificielle a été sollicitée durant ce projet, notamment le plus connu actuellement, ChatGPT qui donne la plupart du temps de bonnes suggestions mais ses réponses deviennent génériques et inutiles lorsque les questions deviennent plus spécifiques, cela s’explique par son fonctionnement qui dépend de la quantité de données analysées pour fournir les réponses. C’est d’ailleurs pour cela qu’il a tendance à mieux nous satisfaire lorsqu’il s’agit de web design et surtout le css, car le principe reste plus ou moins la même peu importe le projet.

Une image contenant texte, capture d’écran, Police

Description générée automatiquementGithub Copilot a également été utilisé, majoritairement pour sa fonctionnalité d’autocomplétions qui analyse le code / le projet et propose automatiquement une suite du code que le programmeur a entamé en fonction de ce qu’il vient de faire où du commentaire/nom de la fonction qu’il vient de mettre. Permettant ainsi de gagner du temps.

Github Copilot générant la suite du code

## Une image contenant capture d’écran, printemps, nature, typographie Description générée automatiquement Choix de l’IDE

Le projet a été codé avec l’IDE Microsoft Visual Studio Code avec le pack d’extensions Java et Git lens/graph\* pour pouvoir commit/push, pull et merge directement depuis l’IDE, le rendant ainsi plus polyvalent.

\*L’application Github Desktop a également été utilisé pour push/pull pour son interface graphique le rendant simple et intuitif.

Contraintes (demandées par le sujet)

## Utilisation de Java SwingFX

L’utilisation autre que Swing est prohibé, seule l’utilisation de Swing et d’autres librairies vues en cours est autorisée. L’utilisation des librairies vues dans les autres UE informatiques sont interdits (comme Gradle)

## Architecture en MV

Le projet devra être structuré en MV (modèle et la vue), cette fragmentation rend une éventuelle évolution du projet beaucoup plus simple à faire.

# Conception

## Utilisation du MVC et explication de son principe

Afin de répondre aux contraintes structurelles, nous avons décidé d’utiliser le MVC (Modèle – Vue – Contrôleur) qui est un design pattern (patron de conception) souvent utilisé pour le web. La vue ne pouvant pas directement interagir avec la base de données, elle se charge des formulaires remplis par l’utilisateur, qui sont récupérés par le contrôleur. Le contrôleur va ainsi interroger le modèle avec les données du formulaire qui va à son tour interroger la BDD (base de données). Les données renvoyées par la BDD suivent le chemin inverse.

En JAVA, tout est sous forme d’objets qui sont alors accessibles partout (sans prendre en compte public, private, protected, …), le contrôleur n’est donc plus essentiel, cependant, le MVC n’est pas pour autant inutile en JAVA. En effet le code MVC permet de :

* Séparer le traitement des données et l’affichage
* Prévoir le futur et favoriser l’évolutivité
* Mieux diviser le code pour faciliter la compréhension et l’édition en mode projet
* D’alléger certains fichiers (car le projet a d’abord été en MV)

En outre, le MVC/MV rend le code plus lisible, plus compréhensible, plus évolutive, mais quant à son implémentation, elle n’en est pas moins compliquée. En effet, savoir comment structurer son code en amont est une difficulté majeure lorsque l’on crée un projet en partant de zéro. De plus, certaines sessions de codages, notamment aux premières semaines, ont été inutiles et ont dû être supprimés à la suite d’une restructuration complète du code.

Une session Parsec a été initiée lors d'une seconde phase de restructuration du code, visant à améliorer la structure du projet sans perturber excessivement le code préexistant et à éviter les défis rencontrés lors de la première restructuration. Cette démarche vise également à faciliter la resynchronisation des membres du groupe.

Une fois le code structuré, le code devient plus explicit et l’implémentation de nouvelles fonctionnalités devient plus triviale.

## Diagramme des classes (demandé par le sujet)

# Implémentations

## Gestion de la map

## Gestion des ennemis et des projectiles

## Gestion des vagues

# Test

Une image contenant texte, capture d’écran, Police, nombre

Description générée automatiquement À la différence du projet de préprofessionnalisation, les tests unitaires et d'intégration n'ont pas été employés dans cette phase.

Des déclarations d'impression (print) ont été utilisées comme moyen de vérification du bon fonctionnement des fonctionnalités. Lorsqu'une erreur n'était pas immédiatement identifiable, des points d'arrêt (break points) ont été utilisés pour isoler le problème.

# Observation et Améliorations possibles

## Objectifs initiaux

## Améliorations possibles

# Conclusion