Projet 320



Fleurdelys Brendan – FID 1

Vennes

40p

Chenaux P.

Table des matières

[1 Spécifications 3](#_Toc192760287)

[1.1 Titre 3](#_Toc192760288)

[1.2 Description 3](#_Toc192760289)

[1.3 Matériel et logiciels à disposition 3](#_Toc192760290)

[1.4 Prérequis 3](#_Toc192760291)

[1.5 Cahier des charges 3](#_Toc192760292)

[1.5.1 Objectifs et portée du projet 3](#_Toc192760293)

[1.5.2 Fonctionnalités requises (du point de vue de l’utilisateur) 3](#_Toc192760294)

[1.5.3 Travail à réaliser par l'apprenti 3](#_Toc192760295)

[1.5.4 Contraintes 4](#_Toc192760296)

[1.6 Livraison 4](#_Toc192760297)

[1.7 Validation et conditions de réussite 4](#_Toc192760298)

[2 Analyse 4](#_Toc192760299)

[2.1 Opportunités 4](#_Toc192760300)

[3 Réalisation 4](#_Toc192760301)

[3.1 Dossier de Réalisation 4](#_Toc192760302)

[3.2 Logique du jeu 5](#_Toc192760303)

[3.3 Utilisation de l'IA 6](#_Toc192760304)

[4 Tests 6](#_Toc192760305)

[4.1 Dossier des tests 6](#_Toc192760306)

[5 Conclusion 6](#_Toc192760307)

[5.1 Bilan des fonctionnalités demandées 6](#_Toc192760308)

[5.2 Bilan personnel 6](#_Toc192760309)

[6 Divers 7](#_Toc192760310)

[6.1 Webographie 7](#_Toc192760311)

[7 Annexes 7](#_Toc192760312)

# Spécifications

## Titre

Projet c# pour le module 320

## Description

Jeu de tir 1 contre 1 où les joueurs doivent tirer selon un angle et une vitesse pour toucher son adversaire. Des obstacles forcent à avoir un angle pour que le projectile fasse une trajectoire en cloche et ainsi avoir une gravité.

## Matériel et logiciels à disposition

Tout

## Prérequis

Module 319 Conception d’applications

## Cahier des charges

### Objectifs et portée du projet

Le projet vise à offrir une expérience de jeu compétitive en 1 contre 1 avec un système de tir basé sur un angle et une puissance ajustable. L'objectif est de proposer un jeu stratégique utilisant des mécaniques simples mais efficaces.

### Fonctionnalités requises (du point de vue de l’utilisateur)

Deux joueurs jouant chacun leur tour

Sélection de l'angle et de la puissance via une interface intuitive

Détection des collisions et gestion des dégâts

Système de score

### Travail à réaliser par l'apprenti

Joueurs : Deux joueurs prenant des tours alternés.

Tours : Chaque joueur possède une tour qui peut être détruite.

Mécanique de tir :

Une barre en arc de cercle détermine l'angle du projectile.

Une seconde interaction détermine la puissance du tir.

Collision : Détection des impacts pour déterminer si un joueur est touché.

Victoire : Le jeu s'arrête lorsqu’un joueur est touché 3 fois

### **Contraintes**

Contraintes techniques :  
- Programmation en C# avec gestion des entrées utilisateurs via la console  
- Affichage uniquement en ASCII, sans moteur graphique avancé  
- Gestion de la physique des projectiles avec une approximation simple de la gravité  
  
Contraintes fonctionnelles :  
- Tour par tour avec alternance des joueurs  
- Détection de collision précise avec les hitboxes définies  
- Gestion des points de vie et condition de victoire claire

## Livraison

* Le code
* Un diagramme de classe
* Un rapport
* Une documentation sur le code

## Validation et conditions de réussite

* Compréhension du travail
* Possibilité de transmettre le travail à une personne extérieure pour le terminer, le corriger ou le compléter
* Etat de fonctionnement du produit livré

# Analyse

## Opportunités

- Compétences développées : Programmation orientée objet en C#, gestion des entrées utilisateur, manipulation des mathématiques pour la trajectoire

- Matériel: PC, Visual Studio, console Windows

- Difficultés potentielles : Ajustement de la physique du projectile, gestion fluide du rendu en console

# Réalisation

## Dossier de Réalisation

Pour réaliser ce projet, j'ai décidé de le découper les classes comme ceci:

1.  **Joueur**  
   Gère les joueurs présents dans le jeu. Elle contrôle leur affichage graphique, leur positionnement dans l'interface, ainsi que leur nombre de points de vie. Elle s'occupe également de gérer les collisions avec les projectiles en mettant à jour la vie restante.
2.  **Tour**  
   Assure la gestion graphique des tours placées sur le terrain, en définissant leur forme, taille et position à l’écran. Elle enregistre également le nombre de points de vie des tours et gère leur dégradation progressive à chaque impact, en modifiant l'apparence des blocs touchés par les projectiles.
3.  **Score**  
   Représente visuellement les scores des joueurs en affichant de façon dynamique les points de vie sous forme graphique (des cœurs). Le score est actualisé automatiquement à chaque modification de vie, offrant ainsi une visibilité claire et intuitive pour les joueurs.
4.  **Game**  
   Constitue le moteur central du jeu. Elle initialise tous les objets nécessaires à une partie (joueurs, tours, interface de tir, etc.) et orchestre la boucle de jeu principale. Cette classe gère l'alternance des tours des joueurs, déclenche les interactions de tirs, contrôle les collisions et garantit la rejouabilité en relançant la partie une fois terminée ou en affichant les résultats finaux.
5.  **GestionnaireTir**  
   Permet de calculer l'angle et la puissance du tir. Cette gestion inclut une représentation visuelle interactive sur la console, animant un arc de cercle pour l’angle et une barre pour la puissance.
6.  **Projectile**  
   Gère la trajectoire, le déplacement et l'affichage des projectiles à l’écran. Elle calcule la trajectoire du projectile en utilisant une simulation physique (gravité, vitesse initiale, angle de lancement). Elle assure également l'animation visuelle fluide et continue du projectile.
7.  **Config**  
   Centralise tous les paramètres généraux et les constantes du jeu, tels que les dimensions de la fenêtre de console (hauteur et largeur). Elle permet une configuration rapide et centralisée du jeu, simplifiant les ajustements ultérieurs et assurant une cohérence entre les différentes parties du programme.
8.  **Position**  
   Définit les coordonnées spatiales utilisées par tous les objets graphiques du jeu (joueurs, tours, projectiles). Elle simplifie les calculs de déplacements, de positionnements, et les détections de collision en encapsulant les coordonnées X et Y dans un seul objet structuré.
9.  **Hitbox**  
   S’occupe exclusivement de la détection des collisions. Elle définit un rectangle virtuel lié à une position, permettant de vérifier si une coordonnée spécifique se trouve à l'intérieur de celui-ci. Utilisée principalement par les joueurs et les tours.

## Logique du jeu

Initialisation de tous les objets du jeu (joueurs, tours, etc...)

Ensuite le jeu est plutôt simple, le premier joueur obtient une interface avec un point qui s'affiche dans un arc de cercle autour de lui. Le joueur doit ensuite appuyer sur la barre espace pour confirmer l'angle de tir qu'il souhaite selon où le point est affiché. Quand le projectile a fini sa trajectoire c'est au tour du joueur 2.

Pour le joueur 2 l'affichage des points doit être en miroir.

Il faut aussi avoir une gestion de points si une tour est touchée.

Le jeu boucle jusqu'à qu'un joueur touche l'autre 3 fois.

## Utilisation de l'IA

Je me suis fait aider par l'IA pour commencer le projet en lui demandant de l'aide sur la structuration des classes et la logique imbriquée sans code.

Par la suite je l'ai utilisé pour certaines méthodes un peu plus complexes comme la mise à jour en temps réel du projectile et son implémentation dans une liste.

Je lui ai aussi demandé d'autres façon de faire certains de mes blocs de codes pour voir des solutions différentes qui utilise des fonctions que je n'ai jamais vu. Par exemple au lieu d'utiliser Convert.To on peut simplement écrire (int)\*(le bloc à convertir)

# Tests

## Dossier des tests

J'ai fait 2 tests pour voir si les hitboxes retournaient bien une valeur si elles détectaient un point à l'intérieur.

# Conclusion

## Bilan des fonctionnalités demandées

* Joueurs : Deux joueurs prenant des tours alternés. 
* Tours : Chaque joueur possède une tour qui peut être détruite. 
* Mécanique de tir :
  + Une barre en arc de cercle détermine l'angle du projectile. 
  + Une seconde interaction détermine la puissance du tir. 
* Collision : Détection des impacts pour déterminer si un joueur est touché
* Victoire : Le jeu s'arrête lorsqu’un joueur est touché 3 fois 

## Bilan personnel

Ce projet m'aura fait découvrir programmation orientée objet et son utilisation dans un projet. Cela m'a permis de me familiariser avec les classes et découvrir ses nombreuses applications et possibilités.

Malgré la difficulté à se lancer dans le projet à partir de rien, j'ai pris beaucoup de plaisir à le faire et à apprendre de nouvelles choses. J'aurai aimé aller un peu plus loin et avoir une meilleure finition pendant le temps du projet.

# Divers

## Webographie

* [Documentation .NET | Microsoft Learn](https://learn.microsoft.com/fr-fr/dotnet/?view=netframework-4.7.2)
* [Stack Overflow](https://stackoverflow.com/?tab=active)
* [ChatGPT](https://chatgpt.com/)

# Annexes

* Code source
* Code des tests
* Documentation Doxygen