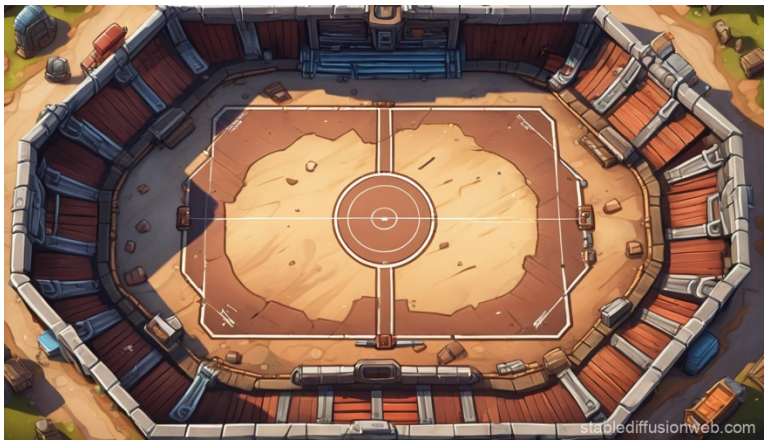
Battle Arena

Réinventons l'huis clos

Introduction du sujet

Le jeu est une version arcade simplifiée d'un "Battle Royale" en 2D. Le joueur contrôle un personnage qui doit survivre dans une arène où apparaissent des ennemis et des bonus aléatoires. Le but est de rester en vie le plus longtemps possible en évitant les ennemis et en collectant des améliorations. Plus le joueur survit longtemps, plus la difficulté augmente !



Objectifs

1. **Créer un jeu rapide et amusant** en utilisant C++ et une bibliothèque graphique légère comme **SFML** pour les visuels (et le son).
2. **Apprendre les bases de la programmation d'un jeu d'arcade** : gestion des entrées utilisateur, génération d'ennemis, collision, score, et génération de bonus.
3. **Renforcer les concepts fondamentaux en C++**, comme les pointeurs intelligents, les classes pour structurer les éléments du jeu, et la gestion de la mémoire.

Étape 1 : les mécaniques

ENNEMIS

Apparition

> Des ennemis apparaissent à des intervalles aléatoires aux quatre coins de l'écran (par exemple) et se dirigent vers le joueur.

Comportement

> fonctionnement normal : s'approche à vitesse constante et en ligne droite, d'abord lentement et de plus en plus rapidement au fur et à mesure du temps ;

> fonctionnement aléatoire : s'approche en utilisant des trajectoires aléatoires – par exemple, 80% du temps vers l'avant (droite, devant ou gauche) et 20% du temps vers l'arrière (droite, devant ou gauche) mais vous pouvez être créatifs sur le sujet ;

> fonctionnement vague : modification de trajectoire, de vitesse, remonte en arrière ?

Vague

> tous les X secondes, lancement d'une vague avec des caractéristiques supplémentaires : plus, plus vite, d'autres comportements

Défense

> ennemis qui peuvent avoir un tir de défense pour détruire les attaques du joueur
> possibilité d'avoir un bouclier

JOUEUR

Comportement

On gère le déplacement avec des touches directionnelles mais vous pouvez proposer des alternatives pour rendre le jeu plus difficile comme du point&click avec la souris mais attention à la gestion des armes.

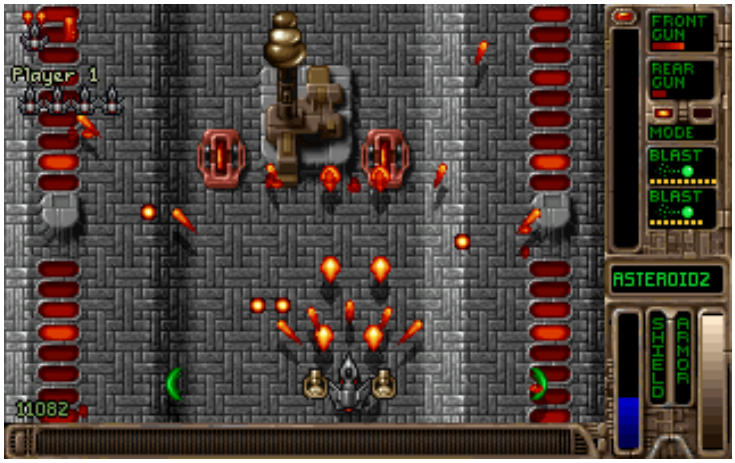
Les armes sont à déclenchement touche sauf dans le cas d'un P&C où le déclenchement serait automatique.

Choix des armes

Plusieurs mécaniques de choix des armes pourraient être proposées :

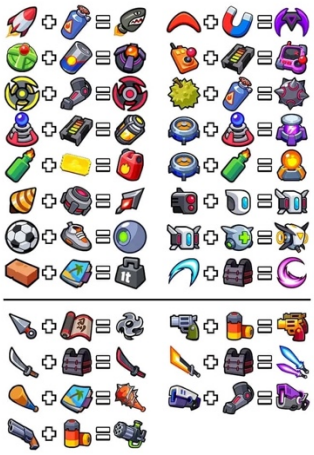
1. l'approche « tyrian » :

Les armes sont données par le pop d'un item. Derrière, plusieurs possibilités : un autre type remplace, le même renforce… etc



2. l'approche « Survivor » :

Choix entre 3 armes et stack de N armes en même temps. On peut avoir des bonus qui les renforcent.



3. l'approche « shop »

La destruction des ennemis fait pop de la monnaie qu'on collecte en repassant dessus (cela peut générer du challenge pour aller dessus).

A intervalle régulier ou sur demande du joueur, on propose une shop pour acheter des armes.

Avantage : ajoute du challenge de collecte

Inconvénient : coupe le rythme du jeu

4. rythme des tirs

Les tirs sont limités avec un rythme. Si le tir est automatique, cela se voit directement. Si le tir n'est pas automatique, cela implique de gérer la pression de la touche et de bloquer son action.

VIE ? bouclier ?

Possibilité de retrouver de la vie : à ramasser, toutes les X vagues, X ennemis…

Possibilité d'avoir un bouclier 1 touche, X touches…

ARENE

L'arène est un espace fermé mais on pourrait imaginer plusieurs choses :

- plusieurs arènes différentes : on sélectionne ou elle est aléatoirement choisie ;

- des arènes avec des thèmes et une liaison avec des types d'ennemis en particulier ;

- une évolution temporelle de la taille de l'arête (pensez à Survivor.io) ;

- des éléments destructibles dans l'arène qui sont présents au début et qui popent parfois, voir qui disparaissent : ils dérivent les trajectoires des ennemis et parfois renferment des bonus.

Étape 2 : les interfaces

Thème et style du jeu

Au choix

Gestion du score

Affichage du score (formule de calcul à déterminer et du nombre de vagues)

Gestion du temps

Affichage temps réel du temps écoulé dans la partie

Gestion de la vie, des bonus et des armes

Affichage des vies restantes, des vies totales ? des bonus en cours et des armes actives

Autre gestion

Soyez créatifs !

Ajouts supplémentaires

Apparitions : Mettez en place des effets pour animer l'apparition des mobs

Ajout de pouvoirs spéciaux : Créez des pouvoirs spéciaux activables, comme des attaques spéciales ou des téléportations.

Niveaux de difficulté : Permettez au joueur de choisir entre plusieurs niveaux de difficulté, avec des changements de vitesse, de nombre d'ennemis, etc.

Classement des scores : Ajoutez un classement pour stocker et afficher les meilleurs scores.

Déroulement du projet

Phase 1 : Conception et planification

Identifiez les classes principales : Player, Enemy, Bonus, Game.
Créez un design de base pour l'arène et définissez les différents bonus et leurs effets.

Phase 2 : Moteur de jeu de base

Installez SFML pour afficher une fenêtre de jeu.
Implémentez le mouvement du joueur et testez-le dans une arène simple.

Phase 3 : Ajout des ennemis et des bonus

Implémentez l'apparition des ennemis avec des mouvements simples pour se diriger vers le joueur.
Ajoutez les bonus et leurs effets sur le joueur (par exemple, en modifiant sa vitesse ou en activant un bouclier temporaire).

Phase 4 : Gestion de la difficulté et score

Augmentez la difficulté en accélérant les ennemis à chaque palier de score.
Ajoutez l'affichage du score en temps réel et le compte de points de vie du joueur.

Phase 5 : Effets visuels et finalisation

Ajoutez des effets visuels pour les collisions, les améliorations, et l'écran de fin.
Effectuez des tests pour équilibrer la difficulté et corriger les bugs.

Calendrier, organisation et évaluation

Organisation des séances

3 séances de TP de 3h soit 18 heures
Et 5 des 6 séances de TD de 1h30 soit 7h30
25h au total
Grille d'évaluation des compétences

1. Bien démarrer

Faire le diagramme de classes/objets

Créer une fenêtre et un renderer avec SFML

Importer une image de fond et l'afficher

Importer une image d'avatar et l'afficher

=> vous avez une fenêtre et la base du jeu

2. Gérer le joueur

Gérer les touches de déplacement, les touches de tir et/ou la souris

3. Gérer les ennemis

Ajouter un ennemi

Gérer l'apparition d'ennemis

Gérer le pathfinding des ennemis

4. Gérer les collisions

Un ennemi ne doit pas entrer dans un autre

Un ennemi ne doit pas pop dans un autre

5. Gérer les tirs

Le joueur peut tirer : apparition de projectiles

Déplacement du projectile

Collisions

6. Score et menus

Ajout l'affichage d'un score

Mise à jour du score

Calcul du score

Ajout d'un menu sur touche Echap

Connaissances / Compétences	Barème
Lisibilité du code - code-style présent et consistant - commentaires (quand c'est nécessaire)	/3
Structure du projet - découpage - architecture - tests d'intégration	/4
Gestion du joueur	/4
Gestion des ennemis	/3
Difficultés et score, temps	/3
Interface graphique	/3
Bonus: ajouts supplémentaires	/+1
Bonus: fun en jeu	/+1
TOTAL	**/20**

Compétences visées

- ➔ C++
- ➔ Librairie SFML
- ➔ Structurer et architecturer un projet
- ➔ Développement d'un jeu complet
- ➔ Gestion d'un projet d'envergure