# **Shoot Me UP**

Une image contenant Dessin d’enfant, dessin humoristique, capture d’écran, illustration

Le contenu généré par l’IA peut être incorrect.Damien Rochat

Vennes

CID2B

Table des matières

[**Shoot Me UP** 1](#_Toc212643703)

[Introduction 3](#_Toc212643704)

[Jeu 3](#_Toc212643705)

[Pédagogiques 3](#_Toc212643706)

[Conception 5](#_Toc212643707)

[IA 5](#_Toc212643708)

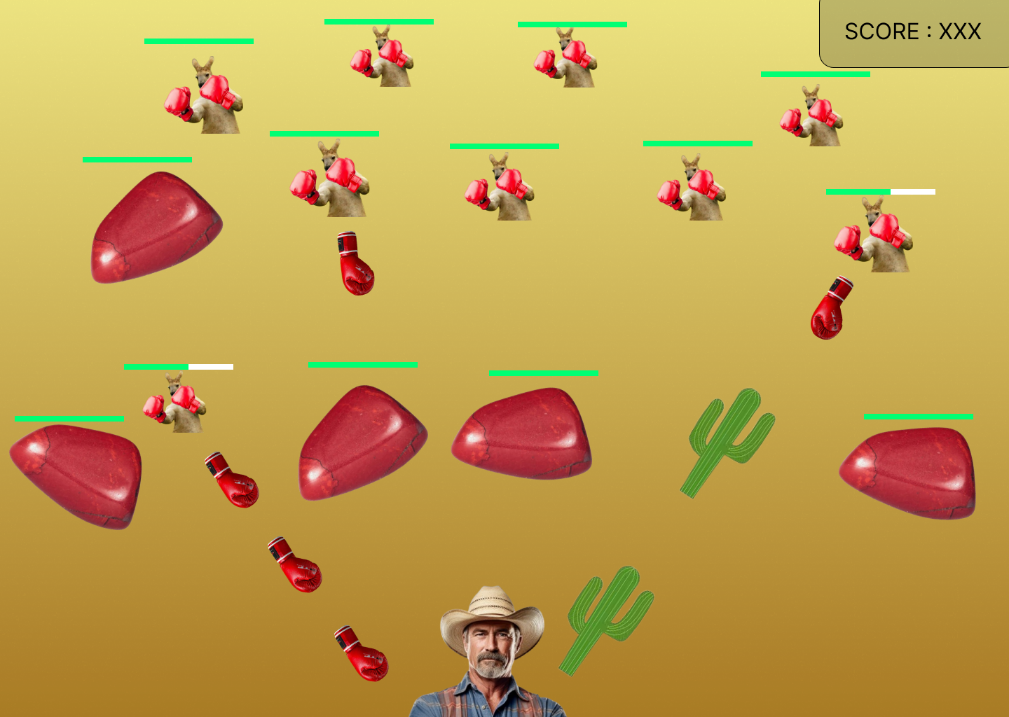
## Introduction

### Jeu

Le jeu "CowBoy" est un Shoot’em Up centré sur la survie et l'esquive. Le joueur incarne un cowboy devant faire face à des vagues de kangourous ennemis qui le chargent et lancent des gants de boxe comme projectiles. L'objectif principal est de survivre le plus longtemps possible en gérant les déplacements (esquive) et la riposte (tir de cactus). La dynamique de jeu est enrichie par la présence de quinze rochers-obstacles permanents qui réapparaissent immédiatement à un emplacement aléatoire lorsqu'ils sont détruits. Ce système maintient un environnement de jeu dynamique et imprévisible, forçant le joueur à intégrer ces éléments de couverture et d'entrave dans sa stratégie pour maximiser son score.

### Pédagogiques

Ce projet sert de plateforme d'application concrète des concepts de la Programmation Orientée Objet. La conception du jeu repose sur la création et l'interconnexion de classes bien définies (Player, ennemi, Projectile, Obstacle, Sand et score), illustrant les principes de la modélisation logicielle. Il permet de mettre en pratique l'encapsulation (gestion des attributs privés via des propriétés), l'héritage (implicite avec la classe Forme pour Sand), l'abstraction et le polymorphisme (gestion commune des collisions entre différents types d'entités, comme l'impact d'un Projectile sur un Obstacle ou un ennemi). La séparation des responsabilités entre les classes garantit un code modulaire et extensible, préparant le projet à de futures évolutions, comme l'ajout de nouveaux ennemis ou de types de projectiles.

Analyse fonctionnelle

Ce si est le premier visuel global du jeu avec les différant élément que je détailles en dessous.

En ce qui concerne les fonctionnalités, elles sont multiples. Le joueur peut se déplacer en bas de l'écran avec les touches « A » et « D ». Le joueur peut également tirer ; cette action se fait avec le clic gauche de la souris, et le projectile, qui est représenté sous forme de cactus, partira en direction de votre curseur de souris. Le joueur gagne également de la vie, en partant de 50 PV et pouvant monter jusqu’à 100 PV, de manière automatique (1 PV par seconde).

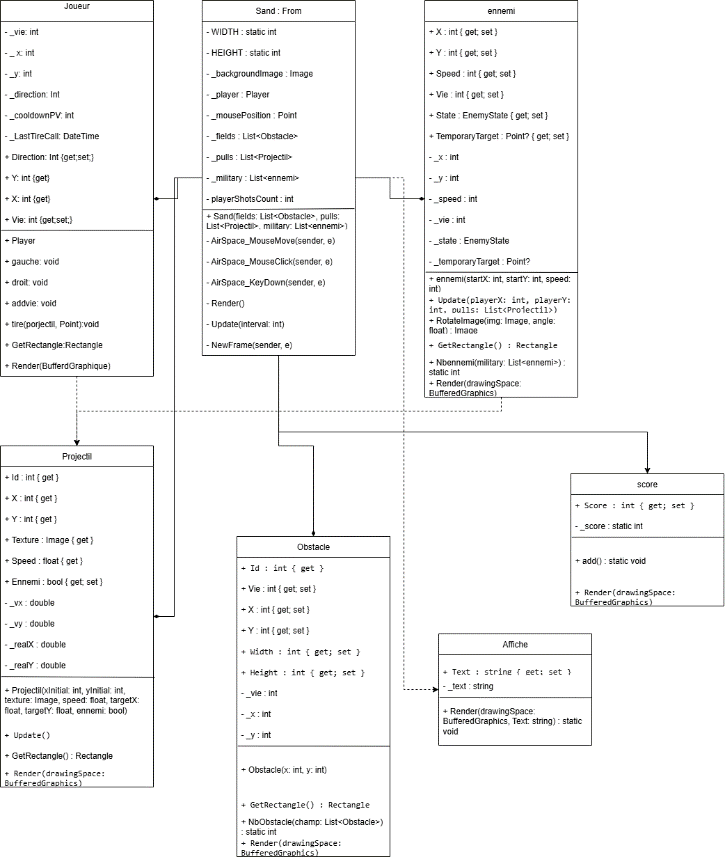
Côtés ennemis, ils ont 3 vies, donc 3 projectiles doivent les toucher avant qu’ils ne disparaissent. Ils sont attirés par le joueur et avancent directement dans sa direction. Les ennemis tirent des gants sur le joueur à intervalle irrégulier ; les projectiles ennemis infligent 5 PV de dégâts au joueur. Les ennemis apparaissent en haut de l'écran et sont toujours 10 sur le plateau de jeu.

Finalement, il y a les obstacles, qui sont dans notre cas des rochers. Ils apparaissent dans la tierce centrale de l'écran (voir la maquette en dessous). Ils sont au nombre de 15 constamment, ce qui veut dire que si un est détruit, il sera automatiquement régénéré dans le centre de l'écran à un endroit aléatoire. Les obstacles sont aussi soumis à un système de vie : ils ont 2 vies, ce qui implique qu’il faut que deux projectiles touchent l’obstacle pour que celui-ci soit détruit.



Cette maquette donne une idée de ou les éléments se trouve.

## Conception



Toutes les méthodes qui s’appellent GetRectangle c’est pour générer le hidbox pour pouvoir faire les collisions

Pour se qui est des projectiles les valeurs \_vx, \_vy se sont des variables utiliser pour calculer la distance entre le ou il est (\_realX, \_realY) et là où il doit aller donc le target. Ce sont des valeur double car avec les multiplications que je fais on retombe régulièrement sur des nombres a visuels, et si je force le INT le projectile n’avance pas fluidement.

La class affiche est probablement la moins clair, la variable « \_text » permet de définir le texte qui sera afficher quand on appelle cette méthode.

Dans ennemis la méthode RotateImage est un peu barbare, elle prend une image de base et en en fonction de l’angle entre le la diagonale qui target le joueur et l’e bord, l’image se fait une rotation de temps de degré bref les calculs sont complexes mais elle prend une image et en retourne une nouvelle

## IA

Dans ce projet, j’ai utilisé l’IA à des fins d’apprentissage, comme pour la détection des touches du clavier, des clics de la souris ou encore pour cibler la position du curseur. Toutes ces utilisations que j’ai citées sont des bouts de code plutôt complexes à connaître, que j’ai découverts grâce à l’IA. Un autre cas d’utilisation a été lors de la compréhension du get et du set. C'était pour moi une nouveauté complexe à comprendre, j’ai donc demandé des exemples concrets à Gemini pour comprendre le fonctionnement. Pour finir L’intelligence artificiel m’a été utiles pour corriger mon français chaotique.