Project Scope

Table des matières

[Analyse des besoins 2](#_Toc98248621)

[Les entités 2](#_Toc98248622)

[Le HUD 2](#_Toc98248623)

[Les projectiles 2](#_Toc98248624)

[La map 2](#_Toc98248625)

[Mode de jeu 2](#_Toc98248626)

[Non réalisable 2](#_Toc98248627)

[Tâches 3](#_Toc98248628)

[Création d’un niveau 3](#_Toc98248629)

[Création du gameplay 3](#_Toc98248630)

[Conception des graphiques 3](#_Toc98248631)

[Convention de nommage 6](#_Toc98248632)

[Nom des entités 6](#_Toc98248633)

[Variables 6](#_Toc98248634)

Pour notre projet nous devons créer un jeu de survie du type « shoot ‘em up » en vue de dessus nommé « Virus 306 », en référence au module du cours. Le joueur se servira d’une arme à distance pour éliminer les ennemis qui lui fonceront dessus pour le tuer. Une partie se termine quand le joueur n’a plus de points de vie et meurt. Une fois la partie terminé le nombre d’ennemis tués est comptabilisé et le temps de survie pour donner un score au joueur.

# Analyse

Dans un premier temps nous avons discuté des éléments nécessaires pour le jeu. Nous retrouvons parmi elles :

## Les entités

Une entité est un objet dans le jeu avec lequel il y aura une interaction. Parmi les entités nous trouverons :

L’avatar du joueur

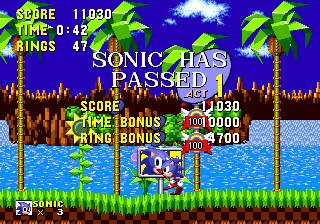
Ça position seras au centre de l’écran à moins que la bordure de la fenêtre touche la limite de la map, auquel cas l’avatar se décollera du centre pour toucher les bords.

Il sera muni d’une arme à feu à distance pour se défendre et tuer les ennemis. L’arme fera apparaitre des balles lorsque la souris sera pressée, les balles partiront du bout du canon dans la direction de la souris par rapport à la position de l’avatar.

Le joueur déplacera son avatar avec les touches classiques du jeux vidéo sur PC : W, A, S, D.

Sa vitesse de déplacement sera la référence de base des autres entités ou des projectiles. Exemple : la vitesse du joueur est 10px/sec (pixels par secondes).

Il aura une barre de vie qui indiquera au joueur s’il est proche du « Game over ». Une fois tombé à zéro, la partie s’arrête et le score est affiché en grand au joueur ainsi que son temps de survie.



Exemple de score : Sonic the Hedgehog

### Les ennemis

Les ennemis apparaitront au hasard sur la map et seront attiré par la position actuelle du joueur.

Les ennemis auront une vitesse de déplacement plus grande que celle du joueur, afin de forcer le joueur à s’en débarrasser au lieu de les esquiver. Cette différence sera de l’ordre de 10%. Exemple : si la vitesse de l’avatar est de 10px/sec alors les ennemis auront une vitesse de 11px/sec.

## Le HUD

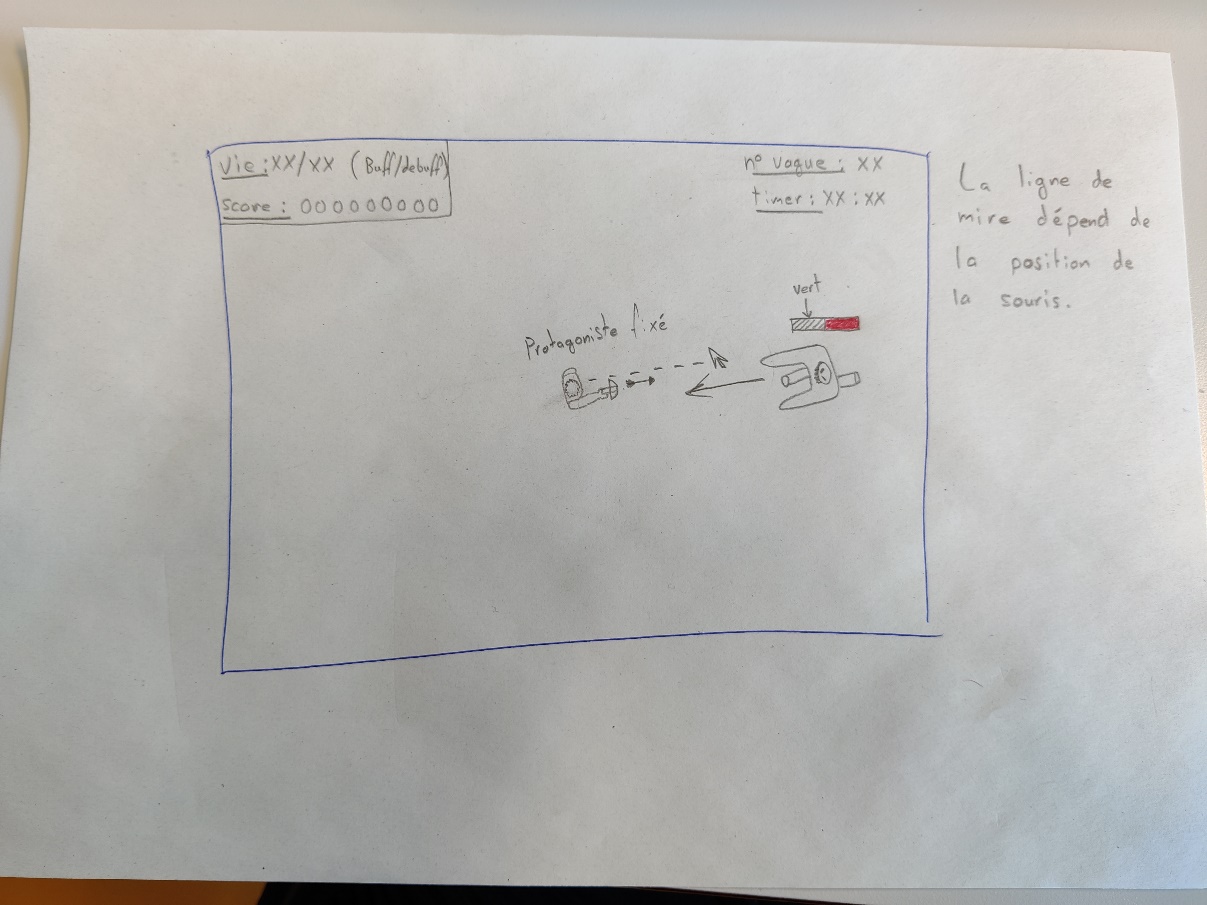
Le HUD (heads-up display), ou « l’affichage tête haute » en français, est un layer qui affichera les informations du jeu en cours au joueur pendant une partie.

Dans le schéma qui suit nous avons fait une ébauche de ce que le joueur verra et quelques idées de l’apparence finale du HUD.

Nous y retrouvons à gauche : la vie du joueur, une zone ou les buffs et debuffs s’affichent et en dessous le score qui s’affiche au fil du jeu.

À Droite : Le numéro de la vague en cours et le temps qui s’est écoulé depuis le début de partie.

Nous avons aussi pensé à afficher la vie des ennemis par une barre verte au-dessus de leur tête.



## Les projectiles

Les projectiles seront essentiellement le moyen avec lequel le joueur se débarrasseras des ennemis. Mais certains ennemis seront aussi capables d’en lancer au joueur pour le tuer.

Il y a les balles de l’arme à feu de base du joueur, mais nous avons aussi pensé à avoir différentes armes au fil du jeu comme par exemple :

* Lance grenade -> projectile : bombes
* Arbalète -> projectile : flèches
* Lance-flammes  -> projectile : feu

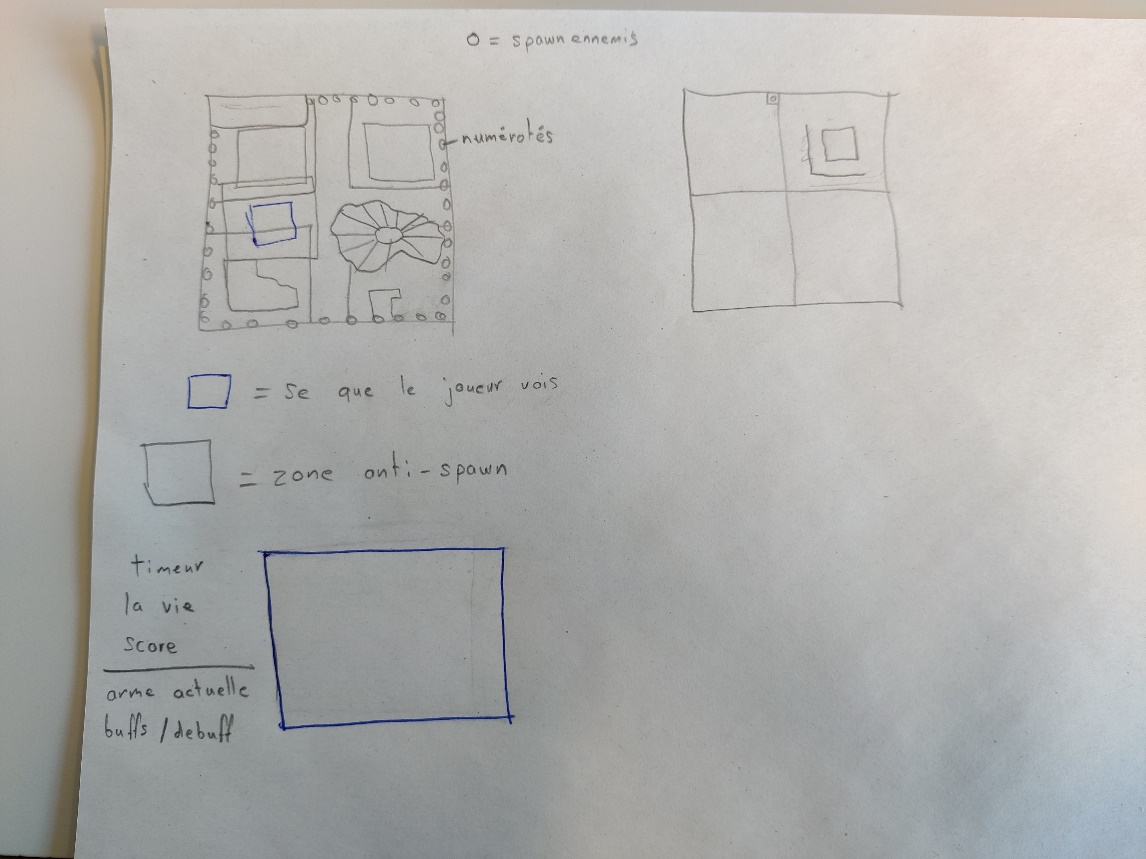
Les projectiles lancés par le joueur auront une vitesse supérieure au joueur de l’ordre de 100%. Exemple : si le joueur est à 10px/sec les balles se déplaceront à 20px/sec.

## La map

La map sera un carré de 1700 pixels de côté.

La bordure sera la limite de la zone de jeu.

Les ennemis apparaitront sur toute la map en dehors de la vue du joueur. Nous avons initialement pensé à diviser la map en quarts, pour gérer les spawns des ennemies. Mais l’idée a été jugée irréalisable car trop exploitable. Nous nous sommes donc penché sur une zone un peu en dehors de ce que le joueur peut voir pour empêcher les ennemis d’apparaitre trop prêt du joueur.



## Stats

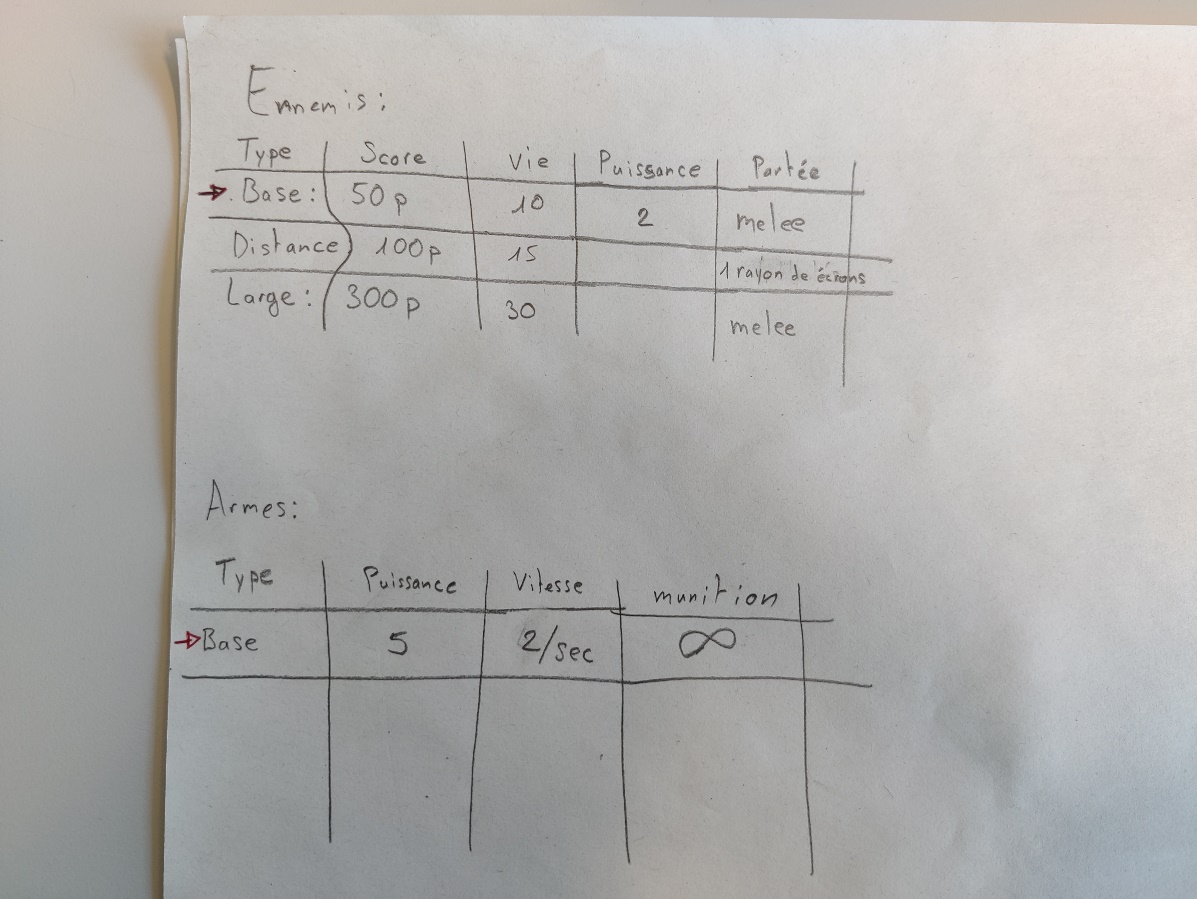
Nous avons discuté des propriétés que nos entités auraient besoin.

Notamment pour les ennemis :

* Le type (Type)
* Le score quand éliminé (Score)
* Ses points de vie (vie)
* La quantité de dégât engendré (Puissance)
* La portée des dégâts (Portée) : Au contact (melee) et à distance

C’est aussi sur cette feuille que nous avons survolé les propriétés de l’arme de base :

* Ses dégâts (Puissance)
* Sa vitesse de tir (Vitesse)
* Sa limite de munition (munition)



## Non réalisable

Parmi les idées discutées, certaine ont été jugée irréalisable dans le laps de temps donné :

* Variété d’armes différentes.
* Limite de munitions.
* Variété de projectiles différents.
* Variété d’ennemis différents.
* Des objets qui donneraient des bonus au joueur.
* Des maps différentes.
* Des obstacles sur la/les map/s.
* Effets sonores.
* Effets visuels.
* Animations diverses.

# Tâches

## Création d’un niveau

* Arrière-plan :
  + Obstacles.

## Création du gameplay

* Ennemis :
  + Vie.
  + Mouvements.
  + Variété.
  + Comportement.
* Protagoniste :
  + Vie.
  + Mouvements.
* Armes :
  + Variété.
  + Mode de tir (press).
  + Type de projectiles.
* Objets de buff.
* Score.
* Chronomètre.

## Conception des graphiques

* Effets :
  + Sang.
  + Coup de feu / Flash.
  + Animation de mort.

# Convention de nommage

## Nom des entités

* Avatar du joueur -> player
* Ennemis -> enemy
* Obstacles :
  + Trous -> hole
  + Murs -> wall
* Projectiles :
  + Balles -> bullet
  + Attaques à distance ennemi -> rangedShot
* Zone de d’apparition d’ennemis -> spawn
* Chronomètre -> timer
* Points obtenus -> score
* Barres de vie -> healthBar
* Souris -> mouse
* Clavier -> keyboard

## Variables

* Santé -> health
* Cadence de tir -> fireRate
* Vitesse de déplacement -> groundSpeed
* Vitesse de projectiles -> shotSpeed
* Dégâts provoqué -> damage
* Numéro de vague -> wave
* Nombres d’ennemis par vagues -> enemyNumber