- ReadMe -
Top-Down View Shoot 'em up Game

Context and Tools : an army of balls wants to kill you : destroy a maximum of them with your fire weapon ! This game has mainly been coded in C++ within Visual Studio 2017, except for animations where UE4 blueprints have been used.

1 Time management

I planned to work on this test during 2 weeks, my daily working time being around 2 hours. Different tasks have been scheduled to accomplish this project :

— Implement a Z/Q/S/D movement system by adapting *Top Down UE4 template* (3 days) ;

— Manage animations for sprinting, jumping and crouching (3 days) ;

— Implement a shooting projectile / scoring system (2 days) ;

— Implement an AI able to inflict damage on the mannequin (3 days) ;

— Implement a random spawning system bounded to the map (3 days).

Please bear in mind that my schedule has been busy, and my work scattered among my available moments. A considerable amount of time was devoted to learning.

2 Features required according to specifications

2.1 Player's gameplay

The aim of the game is to score the highest number of points by killing ball enemies that are chasing the player. The default UE4 mannequin has been used to model the player, and the corresponding C++ class within the project is **ShootemUpCharacter**. The player could fire bullets, with a behaviour stipulated in **ShootemUpProjectile** class.

— Go up, left, down, right : (Z, Q, S, D buttons).

— Fire bullets : (LEFT MOUSE button).

— Jump : (SPACE button).

— Suicide : (SUPPR button).

The player could use walls and pillars to bounce bullets that will collide with enemies. To do some pauses during the game, player can rest on pillars (balls cannot climb obstacles), but the longer he stays, the more difficult the mission will be. The mannequin can bounce on balls when they are numerous as well.

The player has 100 points of health. When the player reach 0 points, game restarts. Same thing happen when player decides to commit suicide (if he is blocked by too much balls, or map is too infested of balls).

2.2 AI

AI is randomly generated within a spawning box that is approximately the size of the map. So a ball can appear in all places, implemented in **ShootemUpEnemySpawn** class. The first step is to detect the player, and the second step is to follow / chase him until his death. UE4 BTService has been used to implement these features within **BTService_FindPlayer** and **BTService_MoveToPlayer** classes.

When the player and a ball mesh overlap, damage is inflicted by enemy (10 points of wealth per second on the 100). Sometimes, balls can not detect player's position by its radar, because player is behind an obstacle hiding him. The code is accessible in **ShootemUpEnemyAI** and **ShootemUpEnemyCharacter** C++ classes.

2.3 Animations

An automaton and an event graph has been created from scratch within the mannequin animation blueprint. Transitions between states depends on Speed, Jump and Crouch variables. This blueprint has then been linked to the mannequin blueprint.

A weapon has been attached to the player's right hand of its skeleton, in order to move with player's movements, whose animations were adapted from *Animation Starter Pack*.

3 Problems encountered and outlook

3.1 Main hardship and remarks

The major problem of this task has been to code almost the entire project (in C++), because most of the tutorials were exclusively using Blueprint, to be accessible by designers and artists. The main purpose of this decision was to demonstrate my skills in C++, applied to video games, in particular for gameplay and AI which were my focus points (due to the fact that it was stated graphics - my speciality - will not be evaluated for this work).

My end-of-studies internship helps me a lot, thus the mastery of UE4 game engine has been quite fast.

3.2 Outlook

— Take time to correct dysfunctional code / blueprints concerning jump animation, crouching ability, scoring system and target position when the mannequin is moving.

— Implement an AI moving more elegantly, with more evolved and intelligent behaviour (eventually fire in direction of the player for instance,....).

— Introduce items, like batteries to collect in order to increase player health or traps.

— Coding an UI to have a more design interface rather that using debug screen (indicating when points are increasing and health decreasing,...).

4 Acknowledgements

I would like to warmly thank you in particular for giving me such an exercise. Indeed, it combines both to implement specifications set beforehand, but also to manage one's time properly, to understand the algorithmic architecture of a Shoot'em up video game and to study UE4 game engine in more details.