Tank Battalion Project Plan

Sean Lin (seanlin)

**Project Description**

Tank Battalion is based off of the video game *Battle City*, a 2d tank shooter game. The player controls a tank and must shoot and destroy 20 enemy tanks every stage. The enemy tanks spawn one by one along the top row of the game grid and proceed to travel down. The stage is complete if the player destroys all 20 of the enemy tanks, and the game moves on to further stages. The game ends if the player loses all lives or the enemy tanks get to the bottom of the grid and destroy the player’s base. The player can shoot to destroy brick walls. After the first stage, the walls are randomly generated, getting more and more sparse as the stages progress, diminishing mobility for the player.

**Structural Plan**

The game will be organized into multiple files and will utilize Object-oriented programming. The game mechanics will be stored in one file, the player data will be stored as classes in another file, and the wall generation will be stored in its own file. Image files will be used as game sprites.

**Algorithmic Plan**

The most algorithmically complex part of the game is the wall generation after the first stage. A 2d list is generated depicting the walls, which will follow standards regulated by iterative functions making sure that the difficulty is appropriate and that it does not block off entire sections of the grid. Another algorithmic part is how the enemy tanks slowly come down.

**Version Control Plan**

Github will be used to back up code. Changes are pushed whenever a class or function is completed as well as at the end of a working day.

Graphical user interface, text, application, email

Description automatically generated

**TP2 Update**

The walls are no longer formed with images, as they seemed to slow down the whole game. Instead, shapes are drawn in their place, which has made the game run with much more smoothly. The first level generates the map based on a template. In later stages there will be random wall generation which increasingly limits mobility.

**TP3 Update**

Explosion animations have been added in the event that the player or an enemy tank is shot. Enemy tanks and bullets are no longer represented by images to reduce lag. DFS is used to test whether the map is connected and used to avoid the generation of isolated unreachable space, and it is reinforced with limits on indestructible block chain lengths. The wall generation remains random and does not become more sparse as the game progresses, as that would make the game too difficult with increasing amounts of enemy firepower. Sound effects have been added for the start of a stage (stageStart.mp3), game over (gameOver.mp3), shooting from the player tank (shoot.mp3), player/enemy being shot (explosion.mp3), brick wall being hit (brick.mp3), and steel wall being hit (steel.mp3). Difficulty increases as enemy speed and shooting frequency scales with the stage. The shortcut command key L advances the game to the next stage, demonstrating the wall generation and enemy difficulty progression.

Pyinstaller was used to pack the whole program into an .exe file, allowing Windows users to run the game without having to install the pygame and PIL modules. (Not included in Autolab submission due to file size limit)