Interim Report

JAVA PROJECT – PHASE 1 – DANK TANKS

JAKOB ETTLES & KEN MALAVISURIYA

1. Executive Summary

The overall goal of this project is to design a peer-to-peer (P2P) online multiplayer 2D top down shooter game. It involves each player controlling a tank which fires bullets in order to destroy the opponent's tank and rewarding players with points to determine the winner. At this stage the core game is developed offline where the user is able to play the game in 3 different game modes. These game modes consist of Arcade, Training and Local Multiplayer mode. The Game has a multilayered menu system where the user can easily navigate between Single Player, Multiplayer, Options and Leaderboard menus. Each game is 2 minutes long and run in a 1024 x 768 window. After the 2 minutes has elapsed, the user is brought to the End Game menu where he can look at player stats and high scores. In local multiplayer, the player with the highest score is congratulated while in arcade mode, the player with a new high score is able to enter it into leaderboards.

In the backend of the game, majority of the game is run on a single thread. This thread is used to run a basic game loop which restricted to game to update and draw at 30 times per second. All the controls in the game are handled by the keyboard, where the tank can be controlled using two different control schemes in single player. The backward and forward movement is handled by using a set speed for the tanks and the rotations of the tank is handled in 22.5 degree increments. Bullets can be fired by every tank at a set fire rate and the bullets can bounce of walls and last for a set time. Every 30 seconds, up to 3 power ups are dropped into the map which can be picked up by any user controlled tanks. There are up to 5 different power ups with 2 of them being penalties which adds variety to the game.

The game is developed in java using the eclipse environment. This programming language was used as it is suitable for 2d games of this scale. It also contains a lot of in built function which helps with the development of the game and has very good support online. Due to time constraints, the game was kept very simple. However, as time progresses, more features will be added to the game to make it a more complete experience.

Contents

2. Game Design

2.1. Game Mechanics

Majority of the game is run on a single thread, with the exception of some timers which carries out tasks that require precise timing. For the scope of this game, single thread worked without any noticeable performance issues. However, if the game were to expand into being more complex, for example advanced AI, then more threads will be required to offload some tasks to make sure the game doesn't suffer from lag issues.

2.2. Game Loop

The main game loop utilises a simple thread to make sure the game updates 30 times per second. This was done by making the thread sleep for the target wait time every loop, so it is delayed and restricted to run 30 times per second. This works relatively well for this game as both game mechanics and graphics are locked to 30 FPS. However, a better implementation would be to avoid a fixed time step and use a delta time system. This is because a fixed time step could cause performance issues if the game is very complex and running on limited hardware. With a game of this scope, a 30 FPS lock provided adequate performance on most test computers.

2.3. Keyboard Inputs

Keyboard inputs are handled by using keyboard listener which is a built java class. With each key press, a sequence of if statements are used to see which function that key event will trigger. Also different game states are taken into consideration as common keys such as the arrow keys are used for multiple functions depending on the game state. For example, the arrow keys can be used to navigate the game menus, be used as player 2 controls, and also be used as player 1 controls in single player game modes.

2.4. Objects

All the objects have the same parameters such as x and y co-ordinates, speed, and a rectangle hitbox. However, some of the objects such as stationery walls do not use all these parameters but it allows to keep the code clean and allows for easy expandability when it comes to introducing new game objects.

For this game, to keep things simple and effective, all the object interactions are handled through rectangle hitboxes. For some of the objects such as tanks, the rectangles are rotated with respect to the tank's angle. This is done by the use of AffineTransforms, which proved effective. The rotations are necessary to keep a realistic feel to the game as tanks are controlled via tank controls by the user. When the tank collides with a wall or window boundary, the forward/backward velocity vectors are set to 0, depending on the direction of the tank. This allows for a desirable function, however it feels clunky with a fast phase game mode such as the arcade mode. Bullet interactions are handled in a similar fashion and has expected effects. For bullet hitting the same tank as it was fired from, the bullet is tracked to see if it bounces, as that is the only way of the bullet realistically colliding with that tank. With the bullet bounce, the interactions are handled in a very simple way. The side of the collision is first evaluated, then depending on the side, x or y velocity vectors are inverted. This works very effectively as the walls are fixed to the axis of the game. There are some unique functions that handles some of the interaction with enemy AI as they behave differently to the user controlled tanks. The biggest difference is the fact that multiple enemy AI are in play instead of just one user controlled enemy. Therefore, a special function is created to handle the bullet vs multiple tank objects. This simple way of handling object interaction is very effective for a game of this scope as it uses less complicated calculation which allows it to run on a single thread.

However most functions need to be revaluated and redesigned to add more depth for a more complicated game.

2.5. AI

The enemy AI uses the same class as user control tanks but expands on the functionality. For the movement, AI follows a simple tracking method where it tries to get to a target location. If it runs into any obstacles, it sets a new temporary target to move away from the obstacle and sets it focus back on the primary target. The AI still uses the same rotation controls as the user control and the rotation angle is calculated by using trigonometry calculations between target coordinates and AI's coordinates. The primary target is the user controlled tank, so the AI will constantly be trying to get closer and shoot the tank down. To select an appropriate temporary target, a certain range within the enemy AI is searched a positions without any other objects. The map design is kept simple so finding a new target position will be relatively fast. If the AI tank manages to get close enough to the user controlled tank, it fires without any delay. This range at which the enemy AI fires was chosen to be challenging but manageable to introduce skill to the game and keep it enjoyable.

2.6. Maps

All the maps are based off a grid based system of the game window. Each grid is the size of a wall object so it is simple to insert wall objects to create a map. This is a very effective way of building maps as the game boundaries are fixed and all the objects are restricted within these boundaries. Each grid is tagged with a variable which shows whether that grid should contain a wall object or not. For map creation, for loops are used to place big walls which contains many smaller wall objects. Some of the for loops are nested for both axis to create symmetry in maps which is important to keep both sides balanced, especially in local multiplayer mode.

2.7. Timers

Two types of timers are used within the game. For some tasks such as keeping track of game time, power up effect time, difficulty time, etc. tick based timers are used. This is because these tasks are timed with regards to game runtime and needs to be reset. For example, with power up effects, the timer need to be reset back if a power up is picked while another power is active. For tasks such as fire rate, object lifetimes, spawn timers, thread based java.util timers are used. This is because the timing of these tasks are fixed and multiple timers for the same tasks have to be run simultaneously. For example, with fire rate, both user controlled tanks and enemy AI tanks have to be timed individually and simultaneously as they can fire at any time and not affect the other tank's fire rate.

2.8. Log File

The log file is kept very basic at this stage of the project as it is not required for the game. The log file is created at the beginning of every game and given a randomly named based on the current date and time to make sure the names don't overlap and the log files don't get overwritten. Then the log file updates every game tick where it writes the current time and the keys in action at that tick. More stats such as tanks coordinates and speeds can also be written at each tick which will be useful to connect other games via a P2P system in the future.

3. System States Design

The Game Starts by entering the user into the Title Screen State. The user can then select the sub menus; Single Player, Multiplayer and Options. Single Player will allow you to select two game modes; Training and Arcade, and Arcade mode Leader Boards. Multiplayer will allow the user to select the game mode Local Multiplayer and the multiplayer options. Option Menu allows the user to pick either Single Player or Multiplayer Options. Within the different game modes, you can access an in game Menu where you can either return to the main menu, access the appropriate options menu or escape back into the same Game. Also within the game modes you will enter an End Game State when the 2 minute timer reaches zero. At the End Game State you can view game stats and leader boards (only for Arcade mode). Exiting the End Game State will then return you to Title Menu.

State Features;

Game Modes (2 minutes Long)

- Training Mode – Single Player mode where the player practices getting use to tank movement and firing a fixed tank object.
- Arcade Mode – Single Player mode where the player moves around a random map, firing bullets at as many A.I. Tanks as possible while try to avoid the A.I.'s bullets. The aim is to gain as many points as possible through killstreak point multipliers.
- Multiplayer Mode – 2 player mode where both players share the keyboard and try to kill each other's tanks as many times as possible. The player with the most points wins.

Leader Board – State displays the 5 best scores in order or ranking achieved in Arcade Mode.

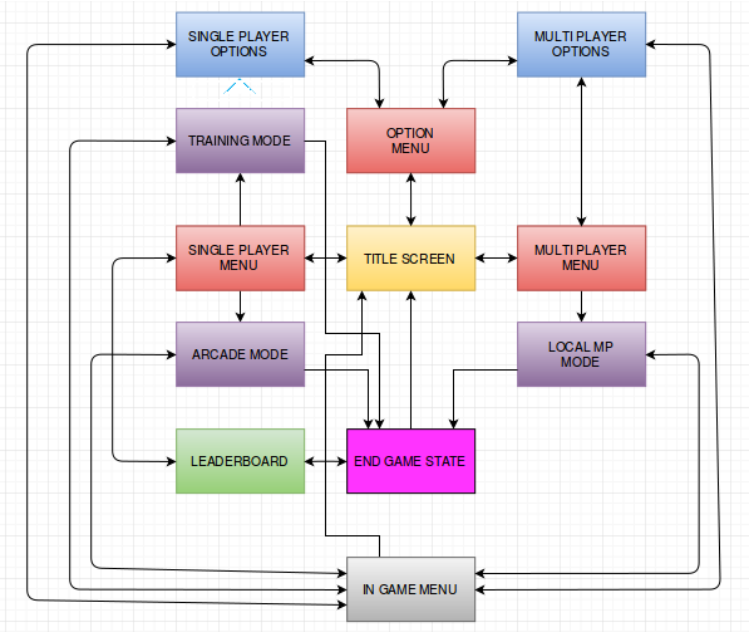
Options States

- Single Player – Player gets to choose between two different control layouts.
- Multi Player – Players gets to choose between 4 different tank colours.

End Game State - Player/s is presented with game stats such as Kills, Deaths, Kill Death Ratio and player score. Leader Board Option if game mode was arcade mode.

The System is set out with having;

- 1 x Title Screen State
- 3 x Sub Menu States (Single Player, Multi Player, Option)
- 3 x Game States(Training, Arcade Mode and Local Multiplayer)
- 2 x Option States(Single Player and Multiplayer)
- 1 x In Game Menu
- 1 x End Game State
- 1 x Leader Board State



4. Significant Issues

4.1. Combining code

Combining code was always a worrying thing, especially when both of us were working on parts that used the same classes. Often we would get contradictions that could take a while to fix, this was especially the case during the mid-period of working on the game. At that time we used bit bucket to merge code but often we would get errors, the addition of wrong files, missing files and so it could take a few hours to just merge the code. We tried to prevent as many contradictions as possible by working on different parts of the code. Ken worked on a lot of the game logic while Jakob worked on the menu's and states. Later we decided to combine code by copy and pasting certain parts of one person's code to the others. Again this wasn't that effective and still produced errors part of the time.

5. Additional Features

5.1. Option Menus

Too enhance the user interface, we added both a single player and multiplayer option menu. These option states can both be accessed through the menus or in game. The single player option allows the user to switch control layouts. Control layout 1 has W, A, S, D (move) and SPACE (shoot). Control layout 2 has UP, DOWN, LEFT, RIGHT (move) and ENTER (shoot). Players may have a preference because of the other games they play may use similar control setups. The multiplayer option allows both players to pick the colour of their tank. The colours include, Red, Yellow, Green and Violet. Again players may have a colour preference and thus giving them the option for some customizability adds more to the experience of the player.

5.2. Leaderboards

Leader Boards was a feature that we had to add to this game. Arcade mode is a game mode that is dedicated to trying to get the highest score. So keeping track of high scores and storing them in a text file was very necessary. The Leaderboards stores the top 5 scores ranked in order with both the name and the score. When a high score is attained by the user, the user is given the option to enter their high score and inputting their name. The Leaderboard is updated in game and written to the file shortly after the name is inputted.

6. Java Development and Game Improvements

The game could be improved by improving the graphics. This could include by adding animations to object interactions, adding a larger variety of objects with different sprites such as walls, trees, bushes, etc. The game could have better map design such as having random generated maps. The game could have a more intelligent A.I. with the addition of possible path finder functions and other behaviour methods. Add a larger variety of sounds for different object interactions. Java is a good platform for developing 2d games of this scale because of the inbuilt functions such as the swing GUI and java timers. It's a popular language with extensive support with lots of tutorials online. It has a wide selection of external libraries for more complex application.