|  |
| --- |
| Wits University |
| Tank Battle Game Design |
| ELEN3009 |
|  |
| **Danielle Joy Winter 563795**  **Jacques Daniel Visser 457817** |
| **07/10/2014** |

|  |
| --- |
| Abstract: Type the abstract of the document here. The abstract is typically a short summary of the contents of the document. Type the abstract of the document here. The abstract is typically a short summary of the contents of the document. |

# Introduction

Atari Inc. released the Atari 2600 video game system in 1977. With the system, they released the game cartridge, *Combat*, which was comprised of 27 games including two-player tank and fighter plane games on various maps [1].

The *Tanks!* game designed is based on the Atari tank game, where players can move on a two-dimensional (2-D) maze which contains obstacles.

# Requirements

The developers are required to implement basic functionality [2] as follows:

* The game is two-player with keyboard input controls.
* Tanks can move vertically and horizontally.
* Tanks can fire missiles in straight lines.
* The game is constrained to the map, where tanks cannot move off-screen or pass through barriers.
* Tanks can plant mines which destroy either player’s tank when collided with.
* The game ends when a tank is destroyed by a mine or missile.

# Constraints

The developers were bound to the following project constraints [2]:

* The game must be coded in ANSI/ISO C++ using the SFML 2.1 library.
* The game must run on Windows.
* The maximum screen resolution is 1600x900 pixels.
* OpenGL may not be used.

# Criteria for Success

The criteria for success as per the project brief is implementing all requirements and adhering to the constraints in sections 2 and 3.

The developers wish to implement additional functionality to add to the basic requirements. The extra minor feature enhancements chosen for implementation are:

* There is a scoreboard which keeps track of the number of times one destroys one’s opponent.
* Destructible barriers.
* Good graphics

The additional major feature enhancements chosen are:

* Tanks are able to move and fire missiles in any direction. Missiles can bounce off maze walls.
* Destructible gun turrets rotate and fire missiles at tanks when they move into range.

Furthermore, the developers consider good layer separation, use of inheritance and the DRY (Don’t Repeat Yourself) principle as well as thorough testing with Google Unit Tests, to be of great importance.

# Design Modelling

The screen is modelled as a 2-D plane of x and y coordinates with the origin located at the top left of the screen. Positive y increases downwards and positive x increases to the right.

The structure of the game is based on inheritance.

## Game Objects and their Construction

The game objects are defined as follows:

* Tanks
* Crates and Concrete barriers
* Turrets
* Missiles
* Mines
* Collision objects

Tanks and missiles are moveable objects. Barriers, turrets and mines are stationary. However, the guns of the turrets rotate. Tank objects are able to move and fire missiles in all directions.

Objects are constructed as vectors of triangles. These triangles are made up of lines, which are made up of coordinates. Triangles were chosen as they reduce complexity for collision detection. Triangles, lines and coordinates are themselves objects in the game. Thus the game objects inherit coordinates which denote their position on the screen, and lines and triangles which bound the region in which objects are contained on the screen.

Game objects, excluding concrete barriers, are destructible. Thus game objects have health, which decreases when the object collides with a rocket or mine.

## Object Physics

The developers decided to design and implement a physics engine to control motion and collision handling for objects. This allows game objects to have a centre of mass, determined by the density of coordinates within the object. Game objects can then have forces applied to their masses which create impulses resulting in movement in a particular direction. Thus forward movement is the application of a forward resultant force on the tank. Furthermore, by implementing the concept of forces within the game, objects can have velocity which can be dampened. Movement is not completely halted when the user stops pressing keys; but rather, tanks will slow down more gradually before coming to a halt. The game physics also allow the tank velocity and rocket velocity to be set easily.

By implementing a physics engine, one can handle object collisions in a way which approximates real life collisions. Objects have velocity and mass, and thus momentum. This means that when one object collides with another, one can implement Conservation of Momentum, meaning that objects bounce off one another.

Furthermore, using the law of reflection, where the angle of incidence equals the angle of reflection, one can model rockets bouncing off surfaces.

## Movement

Tanks can move in any direction on the x, y plane. This was implemented by having forward and reverse controls with clockwise and anticlockwise rotation. Forward and reverse controls were implemented by adding or subtracting to/from the current x and y position of the tank. The amount of forward or reverse movement is dictated by the “engine power” of the tank. The forward and reverse direction of the tank is also dictated by which direction the tank is facing.

Rotation is achieved by using a rotation matrix, which rotates each coordinate in the object’s triangles vector about the origin through an angle θ, and then moves the object back to its (x,y) coordinate. Rotation is realised as a member function of Coordinate.

## Game Graphics

All game object graphics were designed on paper and implemented in Adobe Photoshop 2014.

## Collisions

Collisions are defined by overlapping between objects. Thus a collision occurs when a coordinate or coordinates belonging to the triangle/s of one object move into the triangle/s belonging to another object.

# Structure Overview

The game was designed in two layers, namely the presentation layer and the logic layer. The presentation layer deals with the SFML graphics interface using the SFML 2.1 library. The logic layer implements all other classes and their behaviour. The graphics layer is separated from the logic layer for ease of understanding and as a good coding practice.

## Presentation Layer

The presentation layer is made up of the SfmlInterface and SfmlGameSprite classes. This layer has the lowest level of abstraction and is responsible with presentation of graphics to the user through the rendered SFML window. It is also responsible for user interaction with the game and thus communication with the logic layer.

### SfmlInterface Class

### SfmlGameSprite Class

## Logic Layer

The logic layer comprises all other classes within the project. This layer has the highest level of abstraction, being responsible for all game object physics, calculations and behaviours which can be seen in the presentation layer. The class descriptions below start with the lowest level classes and build up to more complex classes.

### Coordinate Class

The Coordinate class deals with the setting and obtaining of the x and y position of a coordinate. It also incorporates all mathematical operations that can be performed on a coordinate or between two coordinates. This class also has implementation for rotating a coordinate.

### Line Class

The line class takes a vector of shared pointers to coordinates. One can also find the length and slope of a line and calculate whether a coordinate is above, below or on the line as well as finding whether two lines intersect. This is useful in collision detection. The normal to the line can also be found using this class.

### Triangle Class

### GameObject Class

### Collision Class

### CollisionManager Class

### Boundary Class

### Tank Class

### Mine Class

### Rocket Class

### Turret Class

### Crate Class

### Concrete Class

# Object Interactions

## Layer interaction

## Movement

## Collisions

# Critical Analysis

## Layer separation

## Poor Design components

## Design Analysis

# Future Improvements

## Design improvements

The screen resolution was limited to 1600 by 900 pixels. One could enhance the game graphics if the screen were to have a larger resolution. In addition to this, all designed graphics were created with resolution equalling that of the screen. Object graphics were shrunk down for use in the game, which decreased their resolution and thus diminished detail. By increasing the screen resolution, the individual object graphics would appear more detailed.

## Additional Features

SFML allows for the addition of sound clips to the game. The developers could implement a background track and short clips to be played in the event of collisions causing explosions as well as rocket fire.

The design brief suggested a minor feature enhancement of allowing tanks to become invisible for short periods. This feature was written off by the developers, despite being easy to implement as the player controlling the tank would also be unable to see their tank, making it difficult to control. An alternative measure would to be to make tanks intangible for short periods of time, where missiles would pass through them or alternatively not cause any damage.

The original Atari game set had many levels with different maps. A future enhancement would add different maps which could be chosen from a drop down menu. One would read map details from a text file.

The original design incorporated different coloured tanks. Users would be able to choose their tank colour from a list upon start-up of the game. This feature was not implemented due to time constraints.

Another major feature enhancement would be to add computer controlled tanks which target both players.

The developers also implemented moveable crates, which tanks could push around and be protected from turrets. However, the collisions for the moveable crates did not behave as expected and this feature was removed due to time constraints.

# Conclusions

# References

|  |  |
| --- | --- |
| [1] | T. R. G. Geek, “Retro Review: Combat - Atari 2600 - 1977,” 18 February 2013. [Online]. Available: http://theretrogaminggeek.blogspot.com/2013/02/retro-review-combat-atari-2600-1977.html. [Accessed 3 October 2014]. |
| [2] | L. S.P., *ELEN 3009: Software Development II. Project 2014 - Tank Battle,* 1.0 ed., School of Electrical and Information Engineering, University of the Witwatersrand, 2014. |