+

Do It Tomorrow

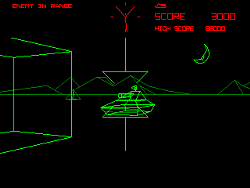
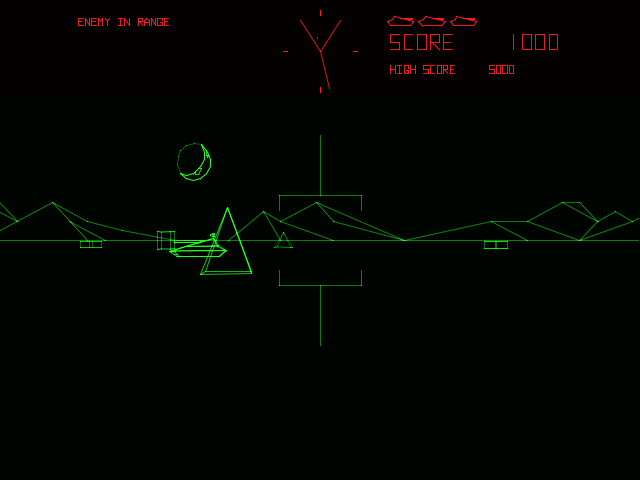
Design Document – Tank Game

Team M: Daniel Hogan, Colm Mulhall, Conor Sargent

Summary

For this project we are going to be creating a Tank game. The game will be 3D based and will be programmed in XNA. It will be based on the 1980 arcade game “[BattleZone](http://en.wikipedia.org/wiki/Battlezone_(1980_video_game))”. The game will make use of 3D vector graphics. We will be using blender for 3D renders as it is open source.

Below are screenshots of the original BattleZone game:

The team has experience with the XNA framework which is a framework used to make games. This makes it the ideal tool for us. We will be working to get a playable prototype up and running in the coming weeks and we plan on bringing the game to this year’s Games Fleadh. The theme to Games Fleadh this year is Robot Tank by Activision. That game was basically a clone of the original 1980 BattleZone game.

We have decided to make our game as similar as possible to the original BattleZone game. We want to make us of wireframe graphics and use a similar control system. As well as this we want to add a few features that weren’t in the 1980 version. We would ideally like to get radar set up so that the player can easily locate the enemy once they go out of sight.

Version History

The design document has been a work in progress throughout the project. Because of this, there have been several versions of it. This version is our final design that we are submitting.

# Version 0.2

This is the first version of the design document. It includes all of our UML diagrams. This version was completed on 21st February 2013.

# Version 0.4

I have included more in this version according to our progress in the project so far. This version was completed on 7th March 2013.

# Version 0.8

This includes most of the information for our project. I will be beginning work on the final document in the coming days. This version was completed on 28 March 2013.

# Version 1.0

This is the final version of the design document. All of the design has been accounted for. It was completed on 22 April 2013.

Game Overview

The game that we are creating is going to be a clone of the 1980 game produced by Atari called “BattleZone”. We will be creating our game using newer technologies than those that were available when the game was originally created so we hope to be able to add more modern features such as radar to the game.

BattleZone was significant in the gaming industry as it effectively was the first first-person shooter game and created a genre that would go on to dominate gaming in the years to come. It used 3D graphics and vector animation to create a more immersive gaming experience.

It’s little wonder so that when the game was released in November 1980 it became a huge hit. Gamers were treated to a 3D gaming experience like they had never seen before. The game represented the birth of 3D gaming.



- The game was originally an arcade game. It has since been ported to many different platforms.

# BattleZone Gameplay

The game itself is relatively simple. It is played in a first-person perspective. It includes the following features:

* Obstacles which are indestructible. They act as good places to hide during gameplay.
* The player will only be battling one tank at a time.
* The game consists of multiple levels, each time getting more difficult.
* Current score will be displayed on the screen at all times.

Assumptions & Dependencies

# Related Software/Hardware

We are creating the game using Visual Studio with XNA. The game will be playable on any system that has Visual Studio 2010 with the XNA game studio.

# Operating System

We will be doing all of the programming for the game on Windows using Visual Studio. The game will be Windows based.

# End-User Characteristics

* The user will be able to control the game using the keyboard.
* They will be able to move the tanks camera to aim in different directions using the crosshair.
* The game will also be controllable with a gamepad.
* It will not be difficult to learn the controls of the game, but it will be difficult to master them.

# Possible Changes in Functionality

As the development progresses we may or may not encounter problems. We might have to add certain features or we may have to remove some. Right now we have a basic goal to create the game and keep as true as possible to the original. That is what will take priority in the project. Once that is complete we will look into adding extra features that were not in the 1980 original.

Development Methods

When we had a meeting to decide on what method we would take to develop our game we soon came to the realisation that we could not use any of the usual development methodologies that we had used for different modules in our course so far.

# Why are sequential models not suitable for game development?

Methodologies such as the waterfall model and star model would make no sense for us. These models are sequential. They follow a strict flow of development going through different cycles. There is a high emphasis on the planning stage, time schedule and target dates. It is all about planning ahead, something that game development isn’t all about.

It is difficult to plan ahead while creating a game. The main reason for this is because it is so unpredictable. You cannot put a specific timeframe on when a feature is going to be completed. Every aspect of the game does need to be planned, but that does not mean that we need to put a specific timeframe on each feature. Therefore is would be inefficient to use a sequential development method.

# The answer: SCRUM

SCRUM is an iterative and incremental agile software development framework. As opposed to the traditional, sequential approach to development SCRUM allows for much more flexibility and is not as tied down to meeting deadlines. Instead, SCRUM is focused towards all members in the group working towards the same goal. No particular path is followed.

Although technically there should be a “Daily SCRUM” where all team members meet up each day to get the agenda, we found that this was not possible because of our other modules that we needed to work on. Instead we had meetings each week in which we had clear goals set out. Following the SCRUM methodology, meetings were short and to the point. We discussed what work needed to be done and how we would go about doing it. Once the work was split between us we got down to it from there. This turned out to be much less restrictive than previous methodologies that we were familiar with.

For a more comprehensive description of SCRUM here are a few links:

* [SCRUM for small teams](http://members.cox.net/risingl1/Articles/IEEEScrum.pdf)
* [What is SCRUM?](http://www.selectbs.com/process-maturity/what-is-scrum-development)
* [Benefits of SCRUM](http://www.scrumalliance.org/pages/benefits_of_scrum)

High Level System Architecture

Our game will be developed while focusing on different areas. Each of these requires their own focus.

# Load models

All content needs to be loaded to the screen when the game begins. This includes the player, the enemy and the obstacles. There will also be sprite fonts to load. These include the player score, radar and background mountains.

# Movement

The movement of the player, enemies and bullets will all be handled by an update function. The player’s movement will be defined by what the player presses on the keyboard or the gamepad depending on which they are using. The enemy will have its own AI logic.

# Enemy AI

The enemy will be moving around the general direction of the player. The player will be able to hide from the enemy using the various obstacles around the battle map. The enemy will move through series of a pre set-up waypoints. As long as the enemy has not been shot down it will continue to follow this path, making it difficult for the player to get a shot at it. This also makes it difficult to use the obstacles to hide from the enemy tank.

# Collision Detection

We will be using bounding sphere collision detection in the game. The basic idea is that for each model you are using, or for each mesh in a model, you will construct a sphere from it. Once you determine the centre of the model, you then figure out the farthest vertex in the model or mesh from this centre -point, and the distance to it is the radius of the bounding sphere. When two spheres intersect each other then there is seen to be a collision and you add logic there. We need to detect when the player collides with an obstacle or with the enemy. There also has to be some way of handling when the enemy collides with an obstacle.

# Testing

Between each of these steps we will be testing for possible bugs in the code and other problems. This will be done on a regular basis and will involve all team members. After the coding is complete we will perform final tests. The aim of these tests is to find any last minute bugs before submission.

# Sounds

An important part of game development is to create an immersive game experience. While there was a lot of work done on the visuals and gameplay mechanics we also seen it as very important that we included appropriate sounds in the game. Our first step in this process was to do some research on other tank related games. We found that most had the usual explosion sounds and shooting sounds. After these sounds were included, we looked into adding a soundtrack to the game. We found an army theme soundtrack that suited the game well. The final sounds that were added were for player collisions with obstacles.

# Game World

As with most elements of the game, we wanted to create a game world that stayed true to the original 1980 game ‘BattleZone’. We looked up videos of the gameplay and discovered that the map was simply a large area populated with obstacles. We added obstacles into the map that spawn in random locations each time that the game runs. The player and enemy then can interact with the world based on where these obstacles have spawns.

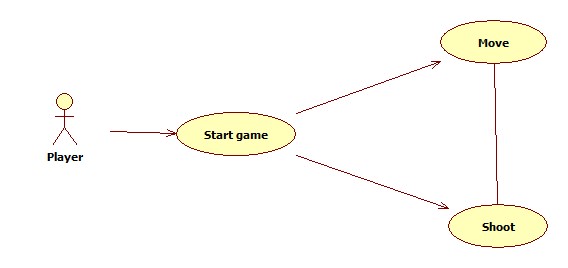
# Scoring

The scoring works by taking into account the players current health. When the game finishes, either when the player gets killed or when the player clears the three levels, the players’ current score is multiplied by a pre set-up value and that determines the score. That way, the fewer shots you take from the enemy the higher the end score will be.

# Radar

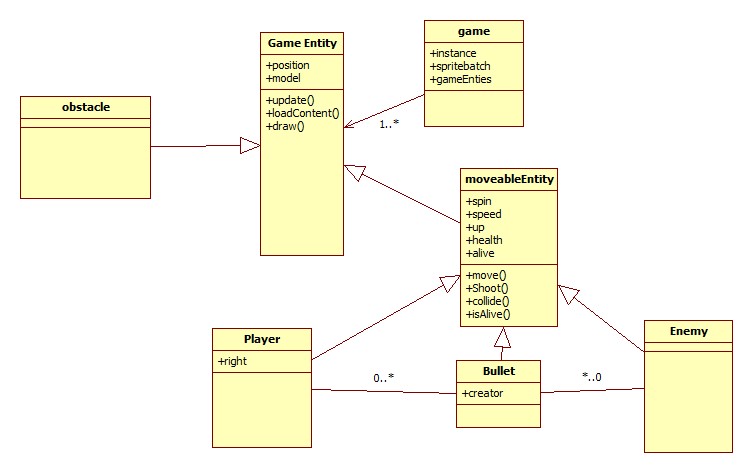
One feature that was not in the original game is radar. We found that it could be difficult at times to find the enemy once he goes out of your sight. Often the enemy would be hidden by obstacles. The radar makes the game more enjoyable to play. It allows the player to hide from the enemy while always keeping track of where it is. The radar works by transforming 3D coordinates into 2D coordinates. The image for the players location rotates based on what direction the player is facing.

Use Case Diagram



Because we are making a game for our project we have a relatively simple use case diagram. Our game is single player, so there is only so much that the player can do. The player can move around the gameplay area and can shoot enemies.

Class Diagram

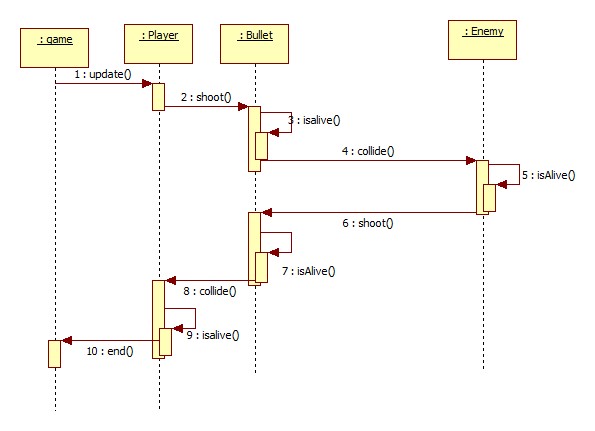


The structure of the game will involve three main objects. The player, the enemy and the bullets. Each of these includes their own model and textures. Bullets will be created and destroyed when they are used by the player and the enemy.

The game entity is what keeps all of the objects in the game in synchronisation. It will keep track of the status of the players and enemies. The objects are drew to the screen and then they use various functions to interact with the environment such as shoot() and move().

The three movable entities in the game are the player, the enemy and the bullet. Each of these share similar attributes and operations such isAlive, speed, move and collide.

Sequence Diagram



The sequence diagram above maps out what the player does when they are in the game. The player can shoot a bullet. The bullet travels until it either hits an enemy or hits some other obstacle. At this point the bullet exits the isAlive() state. If the bullet hits an enemy their health is checked and depending on whether their health hits 0 or not, they either continue on or get removed from the game.

This same interaction happens between the enemy and the player. The enemy is also capable of shooting and killing the player. This again comes down to the amount of health the player has.

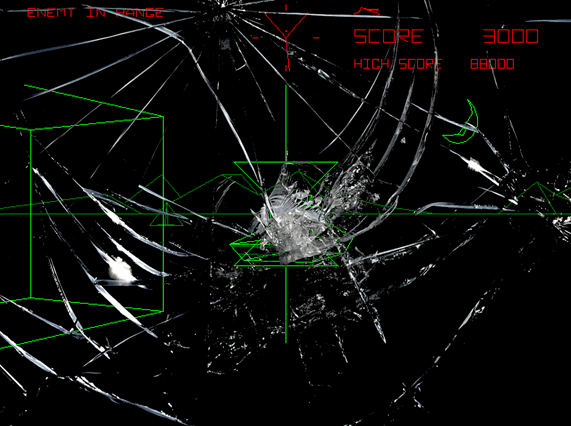
Once the player is killed this ends the game. The players score is then displayed.

GUI Prototype



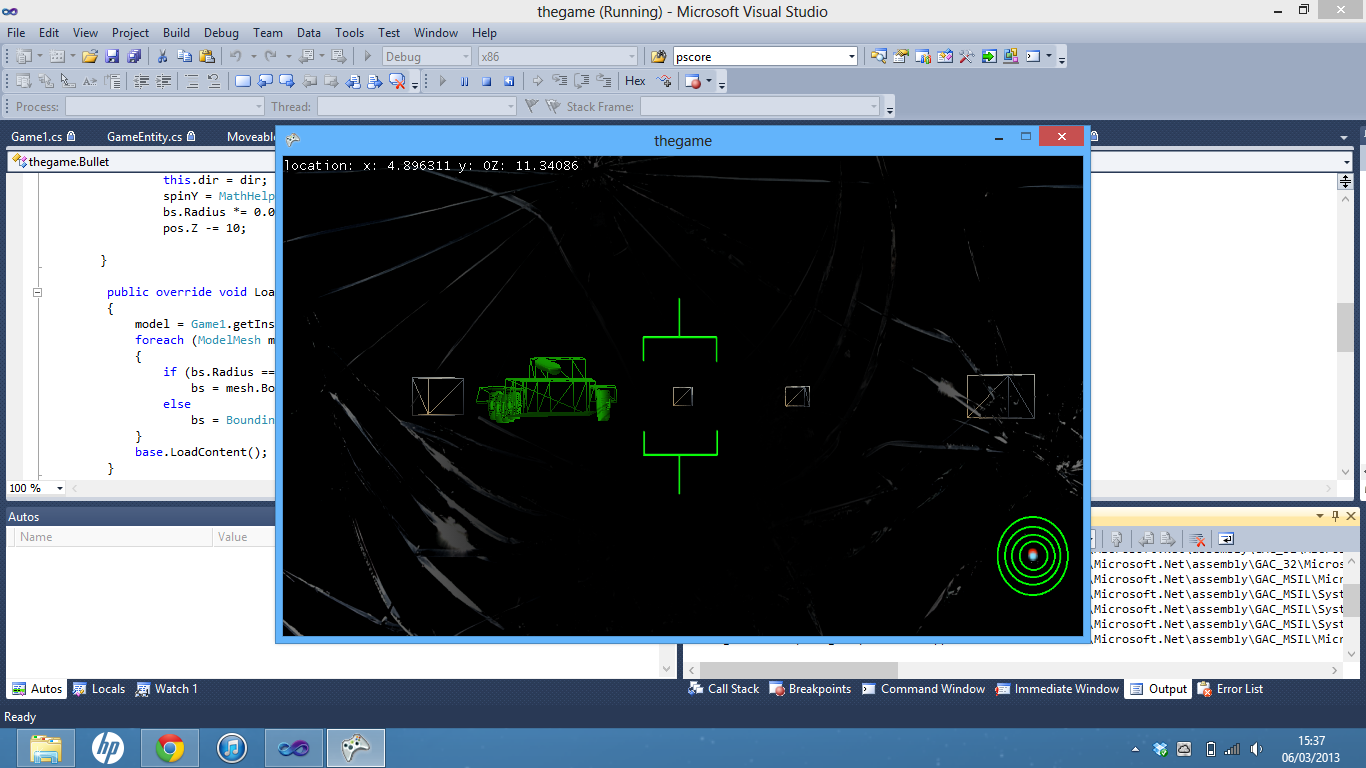
The above picture is a mock-up of when we would like our “start game” screen to look like.

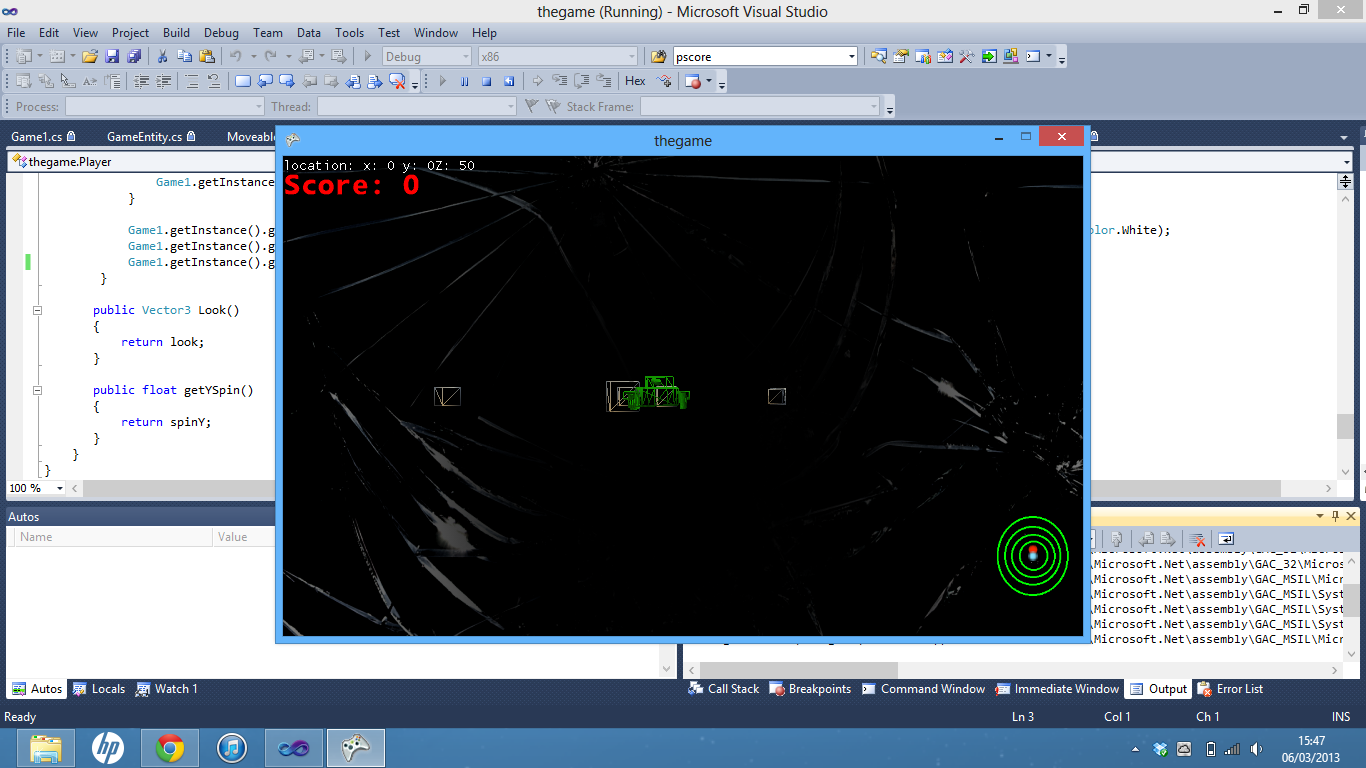
The below picture is another sample mock-up which show how we could represent the player being shot. This looks much more modern than on the original game.



Game Progression

As the game has progressed it has taken several different forms. Below are some screenshots of the development.





Gameplay and Mechanics

# Game Progression

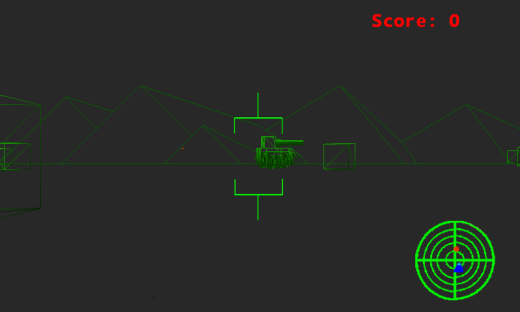
The game consists of three levels. Each time the player destroys the enemy tank they will be prompted to start the next level. Each level has a different difficulty. It takes more shots to destroy the enemy tank as the levels progress. Health will regenerate by a small amount each time the player passes a level. The game does not end until either you destroy the enemy tank, or it destroys you.

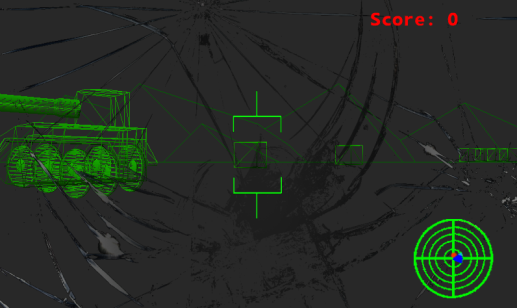
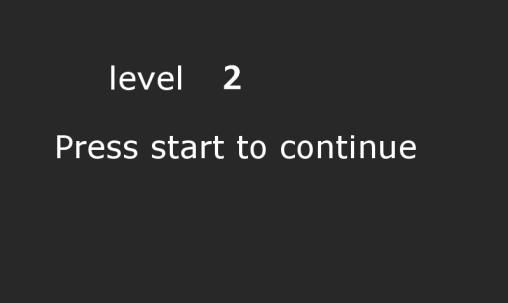
# Controls

Player movement involves moving forward, backwards and turning left and right. The reverse speed is not as quick as forward speed. The player can shoot an infinite amount of bullets at the enemy tank. The enemy can also fire bullets and is controlled by its own AI.

# Screens of the game

Below are some sample action screens of the game.



Related Documents

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | | **Document Title** | **Author** | **Description** | | *Design Document* | Colm Mulhall | A description of our overall design of the game from the early stages until the final deliverable. | | *Test Plan* | Conor Sargent | An overview of the testing process which was on-going throughout the project. | | *User Manual* | Daniel Hogan | A document which describes to a new user how to play the game by showing the controls and scoring system. | | *Team Meetings* | Colm Mulhall | Each time we had a meeting we documented them. This gives a good overview of our progress. | | *Team Member Diaries* | Each Member | Each member of the team has been keeping a personal journal of their progress throughout the project. | | *Presentation* | Each Member | A final presentation of our project. | |