StormVile Game Documentation

**PRGAP - Product Development Document**

*This documentation covers each phase from the requirements of the product through to the design and testing of the system.*

*StormVile Trench Run*

*814853*

|  |  |  |
| --- | --- | --- |
| Version | Date | Comments |
| *1.0* | *04/02/2019* | *Requirements were defined for the project.* |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

Contents

[**Introduction**](#_dnqylakuo5g8) **6**

[Features](#_n80jucnufuta) 6

[Gameplay](#_l8w5rn6psfj7) 6

[Purpose](#_ykq7b2y1xcyn) 6

[Scope](#_m7xa769afm60) 6

[Definitions, acronyms and abbreviations](#_cw0da9bxzoj) 6

[References](#_ks2h4kyg0n80) 7

[**General Description**](#_5rny9synv8x9) **8**

[Product perspective](#_usu1r7f63k19) 8

[Hardware](#_aedbli6wcbdg) 8

[Block Diagram](#_nghcgojilq0e) 8

[Product Functions](#_j2b1dx8um0is) 9

[User Characteristics](#_oc8l46j8v205) 9

[General Constraints](#_4qha7swhojwd) 9

[Assumptions and dependencies](#_wj3tnhdqwf4s) 9

[**Specific Requirements**](#_btrmzn5ipqg) **9**

[Textures Needed:](#_p5phj4xecev7) 9

[Models Needed](#_96ncw84l8ijx) 10

[Hardware Requirements](#_158i3nkfirfp) 10

[Software Requirements](#_kkq580mq48yj) 10

[Interface Requirements](#_jh8zesnv22kc) 10

[U1 - Mouse (100 Mandatory)](#_b32318j9pcx7) 10

[U2 - Keyboard (100 Mandatory)](#_24h7o3w81z6z) 10

[Functional Requirements](#_xq86csw1g8jq) 11

[F1 - Shooting (100 Mandatory)](#_fc71n0nd1zl1) 11

[F2 - Spawning Targets (100 mandatory)](#_fc71n0nd1zl1) 11

[F3 - Spawn Turrets(100 Mandatory)](#_fc71n0nd1zl1) 11

[F4 - Create Corridor (100 Mandatory)](#_fc71n0nd1zl1) 11

[F5 - Create Room (100 Mandatory)](#_fc71n0nd1zl1) 11

[F6 - UI (100 Mandatory)](#_fc71n0nd1zl1) 11

[F7 - Mesh Importer (100 Mandatory)](#_fc71n0nd1zl1) 11

[F8 - Texture Handler (100 Mandatory)](#_fc71n0nd1zl1) 11

[Performance Requirements](#_3cvaber9quy9) 12

[P1 - The system will perform on average no less than 40 Frames per second](#_ks4ppde3kixi) 12

[P2 - The system will have no sudden frame drops](#_ks4ppde3kixi) 12

[P3 - The system will support the input of any keyboard and mouse](#_ks4ppde3kixi) 12

[P4 - The system will initialise and run from hard-drive storage in under 10 seconds](#_ks4ppde3kixi) 12

[P5 - The system will be able to run from removable storage formats given that the right DirectX version is installed.](#_ks4ppde3kixi) 12

[P6 - The system will load levels in under 5 seconds.](#_ks4ppde3kixi) 12

[P7 - The system will not close unless instructed to](#_ks4ppde3kixi) 12

[Reliability requirements](#_3cvaber9quy9) 12

[R1 - Mean time between failures will be 24 hours of game-play.](#_ad3mpvudchyy) 12

[R2 - Failures can be caused by unknown player inputs/devices](#_ad3mpvudchyy) 12

[R3 - The user will not be able to change controls these are hard coded in](#_ad3mpvudchyy) 12

[Design Constraints](#_3cvaber9quy9) 12

[**Design Overview**](#_7051k1rx6lrq) **13**

[Software Structure Chart](#_o4b6osy2q7ns) 13

[Screen Design](#_28g87cchvocu) 13

[Main Menu](#_me2ttqj7mqgn) 13

[Game](#_r8u41is0xtlo) 14

[Level Selector](#_k9ahbwb2wee5) 15

[Level Design](#_2vysopacdlbi) 15

[Level 1 Level 2 Level 3](#_ll3traxqkq30) 15

[**Detailed functionality description**](#_fwzgwndz8stt) **16**

[D1 - Texture Importer](#_sgxke4u3vo37) 16

[Description:](#_uer2ah3va561) 16

[Sub-Components:](#_5eulugrzknwq) 16

[Dependencies:](#_ieo6fpm841si) 16

[Used by:](#_8w2657hl48rl) 17

[D2 - Model Importer](#_ixv7qwfzqo1e) 17

[Description:](#_nbo77guif5a0) 17

[Sub-Components:](#_xmm2ymrlmjgw) 17

[Dependencies:](#_uz1u6s7627dh) 17

[Used by:](#_x8zgn0skng1a) 17

[D3 - Transformation](#_d9ky8nqca8m4) 17

[Description:](#_t8h5na23nn2) 17

[Sub-Components:](#_h7d6h0xd3qyr) 18

[Dependencies:](#_2ebcc2pcdz4u) 18

[Used by:](#_cqn6dfle4zns) 18

[D4 - Collision Detection](#_s2wqbu1jlu1i) 18

[Description:](#_nlsg95mc1c6q) 18

[Sub-Components:](#_ae6y4i2919ea) 18

[Dependencies:](#_evnrny3qof4u) 18

[Used by:](#_a44cee7ml4t7) 19

[D5 - Targets](#_el9fzesduit4) 19

[Description](#_q6lpgb3cbnu7) 19

[Sub-Components:](#_wb7x85mhop27) 19

[Dependencies:](#_fgy4acbnf17f) 19

[Used by:](#_skioemvudzes) 19

[D6 - Shooting system](#_oaxzsxvaxjh5) 19

[Description](#_81a788ud4lne) 19

[Sub-Components:](#_jzbr2kue03hw) 20

[Dependencies:](#_ig9fbcgh0u6) 20

[Used by:](#_kb5dtjihny5t) 20

[D7 - Turret](#_pq83sxeianvb) 20

[Description](#_woocp101v5t) 20

[Sub-Components:](#_rx9bnxogbsk) 20

[Dependencies:](#_qpzjgepaleha) 20

[Used by:](#_1qe8hxhjvl8b) 20

[D8 - Player Input](#_o2dapq6gs4ml) 21

[Description:](#_4g4j9rfbls1) 21

[Sub-Components:](#_3z9c0smv7go0) 21

[Dependencies:](#_8zrm9mef01zx) 21

[Used by:](#_80ozxde4o69w) 21

[D9 - Ship Flight](#_rbkh3lewzm68) 21

[Description:](#_b28oa2f8l9bk) 21

[Sub-Components:](#_ek5ikixtrm9y) 21

[Dependencies:](#_hpmzh7boqxpj) 22

[Used by:](#_qjp2srnqzdtr) 22

[D10 - Corridors and Rooms](#_1dudhzla5wzi) 22

[Description](#_wjk39tkxpjgj) 22

[Sub-Components:](#_js5ehpc5vf76) 22

[Dependencies:](#_lvk0tarqdicz) 22

[Used by:](#_48tupnsvbmye) 22

[D11 - Text](#_t5aceby2vjs) 22

[Description:](#_kzt8e42axjlq) 22

[Sub-Components:](#_ml0x3iup7g6s) 22

[Dependencies:](#_hr9tz3l50dl1) 22

[Used by:](#_wmswdvrheukq) 23

[**Requirements Mapping**](#_fwzgwndz8stt) **23**

[**Full Function Plan**](#_4xqqqafwzy5z) **24**

[**Maths**](#_bkwfs95mfqtz) **24**

[**Text**](#_dhi3mtrphe29) **24**

[**Collision Detection**](#_p58zbu52m35g) **24**

[Update](#_aap6n1axc4g) 24

[Intersects](#_aap6n1axc4g) 24

[**Transformation**](#_lrle8l6ur7yt) **24**

[Draw](#_62aeawlbfybf) 24

[**Menu**](#_40yz69drydax) **24**

[Create Button](#_yqlx31emsaid) 24

[Update](#_g6tpxw30ino3) 25

[**Boundaries**](#_40yz69drydax) **25**

[Update](#_visnjngolxde) 25

[Spawn Target](#_visnjngolxde) 25

[Spawn Turret](#_visnjngolxde) 25

[**Bullet**](#_40yz69drydax) **25**

[Update](#_29b2fdrcbct7) 25

[**Target**](#_40yz69drydax) **26**

[**Player**](#_40yz69drydax) **26**

[**Rooms**](#_40yz69drydax) **27**

[Setup](#_ih82zxyjhfsm) 27

[Update](#_lcfb6hisonik) 27

[**Gun**](#_40yz69drydax) **27**

[**Corridors**](#_40yz69drydax) **27**

[**Main (Engine)**](#_40yz69drydax) **28**

# 

# Introduction

## Features

* Model Importer
* Random Targets (Health can be different)
* FPS
* Laser Bullets
* Flashlight
* Text

## Gameplay

Fly around in tight corridors

Shoot random targets or turrets

Go as quickly as possible

Do not waste bullets

## Purpose

* 1. This document details the design of an interactive 3-Dimensional product for submission along with the implemented product at the end of the DIRXENG course.

## Scope

* 1. The development will be of a product that allows a user to interact and do the following:
     1. Go through 5 pre-made levels
     2. Shoot at targets
     3. Timed on how quickly they go
     4. Counts the number of shots fired
     5. The object is to clear the trench and not waste shots

## Definitions, acronyms and abbreviations

* 1. *O-O – Object Oriented.*
  2. *DirectX – A Game SDK used to handle the input and output aspects of game play.*
  3. *Procedural - Generation of a level/assets using random elements*
  4. *Assets - Game objects/models e.g. Tree/book*
  5. *Level - Premade/user made layout of objects place in a position to create a world*

## References

* 1. *Walbourn, C. (2019). walbourn/directx-sdk-samples. [online] GitHub. Available at:* [*https://github.com/walbourn/directx-sdk-samples*](https://github.com/walbourn/directx-sdk-samples) *[Accessed 28 Jan. 2019].*
  2. [*https://www.pinterest.co.uk/pin/30188259980408133/*](https://www.pinterest.co.uk/pin/30188259980408133/)
  3. <https://github.com/Microsoft/DirectXMesh/wiki/Geometry-formats>
  4. <https://australbricks.com.au/sa/product/homestead/>
  5. <https://www.muralswallpaper.com/shop-murals/black-marble-wallpaper/>
  6. <https://www.pinterest.co.uk/pin/327707310363304027>
  7. <https://www.youtube.com/watch?v=SuMBYZrcQfU>

System Requirements Specification

# General Description

* 1. This Section describes the general factors that make the product and what requirements they have. IT does not state specific requirements, it just makes those requirements easier to understand

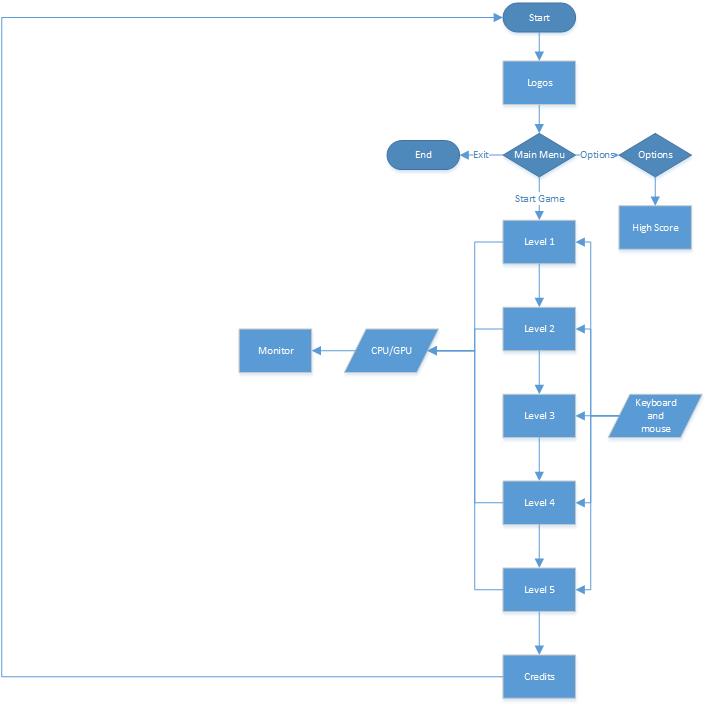
## Product perspective

* + 1. *The product will be developed to run on Windows computers capable of effectively running DirectX 11.0 or greater.*

### Hardware

* + - 1. PC Keyboard and Mouse. No controllers, joysticks
      2. Static Window resolution

### Block Diagram



## Product Functions

* + 1. The game will allows the user to take control of a star-ship going through tight and narrow corridors. The game will allow the user to move this ship in a 360 degrees with a 3RD person camera locked to the ship. The game will also allow the player to change this camera to be first person. There is possibilities for the game to have UI elements like crosshairs and additional information. When the user interacts with the ship they can accelerate it in the direction it is facing. The ship will be speed clamped to slow down. When the user hits a wall they take damage represented by the ships colour.

## User Characteristics

* + 1. *The user will be academics and students within the University of Portsmouth that are familiar with the use of computers and 3D graphical applications.*

## General Constraints

* + 1. *The development of the product will be designed and implemented for the lab environments within the University of Portsmouth.*
    2. *The game requires a 1920 \* 1080 resolution so the gameplay experience is more uniform.*

## Assumptions and dependencies

* + 1. The developers assume people playing this game have a “gaming” PC capable of handling basic gameplay, the game has multiple objects with differing textures and basic lighting, this can cause lag on older PCs
    2. Windows 10

# Specific Requirements

* *This section provides comprehensive detail on all requirements. It should include all of the detail, which the designer will need to create the design*

## Textures Needed:

Text

Modern Brick

Ancient Marble

Futuristic Carbon Wall

LED Wall

1 Ship texture

## Models Needed

Ship

Gun

Bullet

Turret

Wall

## Hardware Requirements

* + 1. *Computer:*
    2. *i5*
    3. *Hard Drive 4GB space*
    4. *GTX 970*
    5. *8GB Ram*
    6. *Monitor:*
    7. *Any 1920\*1080 monitor*

## Software Requirements

* + 1. *DX11 Installed on PC*
    2. *Windows 10*

## Interface Requirements

### U1 - Mouse (100 Mandatory)

* + - 1. Left Click: Fire Weapon
      2. Moue: Aims Gun Gimbals

### U2 - Keyboard (100 Mandatory)

* + - 1. W: Rotate Down
      2. S: Rotate Up
      3. A: Strafe Left
      4. D: Strafe Right
      5. E: Rotate Ship Right
      6. Q: Rotate Ship Left
      7. lShift: Thruster Increase
      8. Control: Thruster Decrease
      9. R: Center Guns
      10. X: Breaks
      11. Space: Fire Weapons
      12. Left: Rotate Camera Left Around Player
      13. Right: Rotate Camera Right Around Player
      14. Up: Rotate Camera Up Around Player
      15. Down: Rotate Camera Down around player

## Functional Requirements

### F1 - Shooting (100 Mandatory)

* + - 1. The program will shoot bullets from a gun placed on the player ship

### F2 - Spawning Targets (100 mandatory)

* + - 1. The program will spawn targets on walls or floors at random intervals and amounts
      2. Game logic should keep track of number of targets

### F3 - Spawn Turrets(100 Mandatory)

* + - 1. The program will spawn turrets on walls or floors at random intervals and amounts
      2. Game logic should keep track of number of turrets

### F4 - Create Corridor (100 Mandatory)

* + - 1. The program can create corridors at a set length ( max 50)
      2. The floors will be seperate so you can handle textures and lighting better

### F5 - Create Room (100 Mandatory)

* + - 1. The program can create 4 rooms
         1. 2 x 2
         2. 3 x 3
         3. 4 x 4
         4. 5 x 5

### F6 - UI (100 Mandatory)

* + - 1. The program displays stats like health and ammo used and time

### F7 - Mesh Importer (100 Mandatory)

* + - 1. The programm imports meshes from OBJ files

### F8 - Texture Handler (100 Mandatory)

* + - 1. The programm imports textures from DDS

## Performance Requirements

### P1 - The system will perform on average no less than 40 Frames per second

### P2 - The system will have no sudden frame drops

### P3 - The system will support the input of any keyboard and mouse

### P4 - The system will initialise and run from hard-drive storage in under 10 seconds

### P5 - The system will be able to run from removable storage formats given that the right DirectX version is installed.

### P6 - The system will load levels in under 5 seconds.

### P7 - The system will not close unless instructed to

## Reliability requirements

### R1 - Mean time between failures will be 24 hours of game-play.

### R2 - Failures can be caused by unknown player inputs/devices

### R3 - The user will not be able to change controls these are hard coded in

## Design Constraints

* + 1. *The target hardware system will be a I7*
    2. *The memory available will be 4GB of main memory.*
    3. *The size of the final product is restricted to the size of the CD-ROM format (under 700Mb).*
    4. *The graphics hardware required will be an NVidia GeForce 4.*
    5. *The computer system will have the DirectX 11 runtime installed.*
    6. *The user will not be able to use a controller or joystick*
    7. *Textures and models are basic to save performance.*

Software Design Description

# Design Overview

## Software Structure Chart

* + - (In Project File Go to: Stormville Trench Run\Tutorial07\Design\Software Structure Chart) Appendix 1: Software Structure Chart

## Screen Design

### Main Menu

### 

### 

### 

### 

### 

### 

### 

### 

### Game

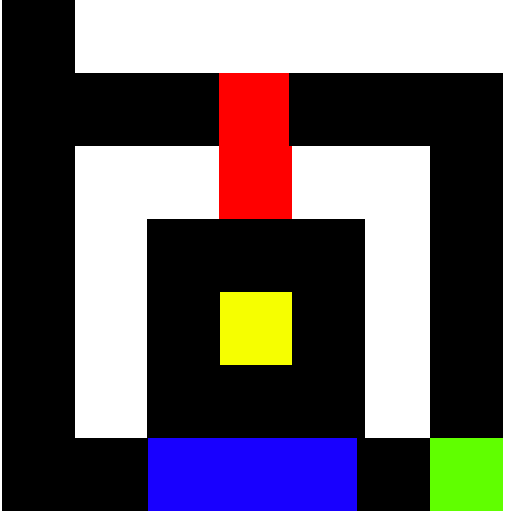
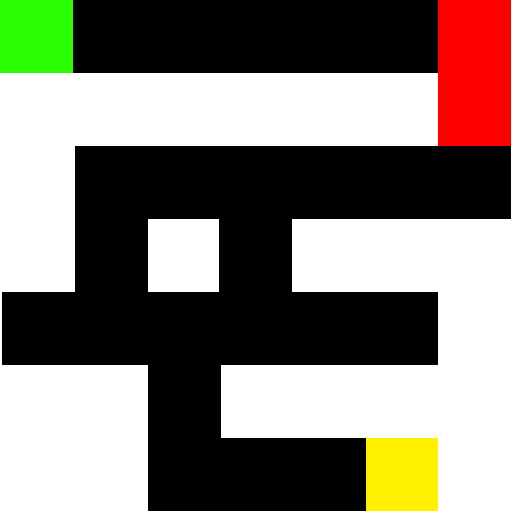
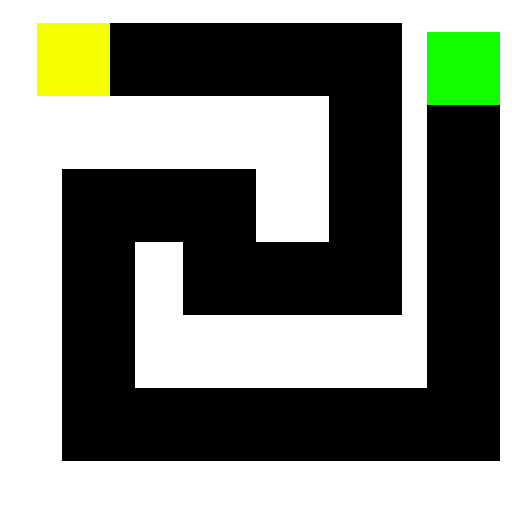
### 

### Level Selector

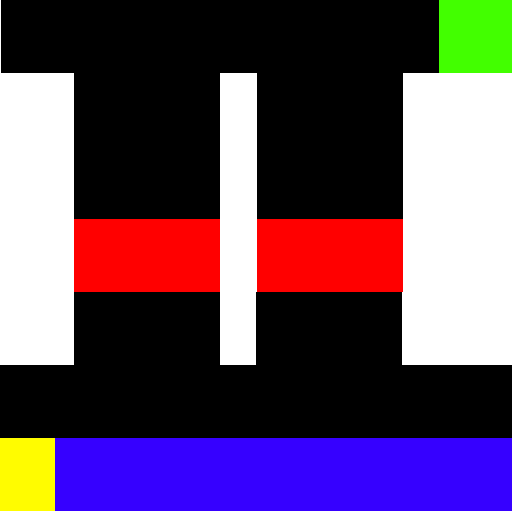
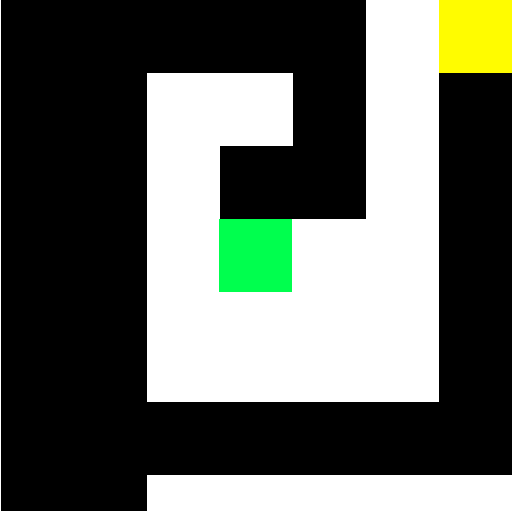
### 

## Level Design

### Level 1 Level 2 Level 3



Level 4 Level 5



Level 1 is a simple snake like level, black represents the corridors positions, Yellow is where the player starts and green is the finish. The level designs are not what the final level will look like but more of an idea of how to layout the level for the player and a general structure needed.

Level 2 brings in the concept of multiple pathways and shorter corridors, it also adds a new colour red, red is a vertical shaft up or down for the player to fly in. The player must again reach the green cube to go to the next level. Turrets and targets can spawn on either end of the connecting corridors and this can add challenge to the player

Level 3 starts in a room, the room size will be determined in the programming phase, to get out of the room the player must fly up/down the shaft to enter a corridor, this corridor will go around in a circle, with a dead end, the dead end is guaranteed to have a turret. The blue colour is used to represent that it is on a different level to the black room the Green cube is on a corner

Level 4 Goes back to a more basic concept but with a wider corridor/ room this will give the players more space to shoot and fly around in and the level is more build for fast speeds. The Corridors go tight quickly giving the player a chance to crash and die

Level 5 The most complex level to create in game this level will require a lot of manual setups with the prefabs.The player starts in the bottom left, the blue corridor is separate to the black corridor with an entrance at the end. The player then has a choice of two wide corridors, both of these corridors have a shaft going up to the next level the player can then make their way to the green end.

# Detailed functionality description

This section Details the functionality of classes, their purpose, sudo code of what they do and any dependencies of sub components

## D1 - Texture Importer

### Description:

* + - * The Texture importer, imports textures from DDS files, the purpose of this is to have external textures such as bricks. DDS files can be read from file and applied in main

### Sub-Components:

* + - * DX11

### Dependencies:

* + - * DX11
      * DDS

### Used by:

* + - * Main/Transformation
      * Objects and Meshes

## D2 - Model Importer

### Description:

* + - * The model importer reads obj from file and applies their model data to the mesh structure used by DX11, this can include UVS to apply textures. This is optional

Read file from directory

Vertex Buffer = Obj Model Vertices by reading the file line by line and getting the vertex point

UV = OBj Model UV

Store In memory

### Sub-Components:

* + - * DX11

### Dependencies:

* + - * DX11

### Used by:

* + - * Transformation
      * Main

## D3 - Transformation

### Description:

* + - * Main Transformation component which is used by all objects to define where they are in the world, how big they are, their rotation and their colour

Update Transformation:

Position = new Position (XMFLOAT3(0,0,0));

Scale= new Scale(XMFLOAT3(0,0,0));

Rotation= new Rotation(XMFLOAT3(0,0,0));

Red = 50

Blue = 10

Green = 1

### Sub-Components:

* + - * DX11
      * Mathf
      * Vector Maths
      * Collision Detection

### Dependencies:

* + - * DX11

### Used by:

* + - * All Game Objects
      * Collision Detection

## D4 - Collision Detection

### Description:

* + - * AABB Based Collision detection using basic collision checks to increase performance all objects have a AABB box that automatically updates every frame
      * Intersect tests are done in main but the functionality is in collision detection

Update

AABB MIN = Transformation - Scale

AABB Max = Transformation + scale

Interesects(AABB BOX1, AABB BOX2)

Check if Min is more then max is so swap them

If (BOX1 is in BOX2)

Return true

Else

Return false

### Sub-Components:

* + - * Transformation

### Dependencies:

* + - * Transformation

### Used by:

* + - * All Game Objects

## D5 - Targets

### Description

* + - * Class that handles and spawns targets in for the player to shoot, these targets are static but still need to check for collision detection

Update

Get Bullet list from players gun

Check bullet I box against target box

If collision is true

Set game logic score to +1

Set Shots fired to +1

Destroy target

### Sub-Components:

* + - * Transformation
      * Collision Detection

### Dependencies:

* + - * Collision Detection
      * Transformation

### Used by:

* + - * Level Creation

## D6 - Shooting system

### Description

* + - * Handles the shooting of bullets from guns, handles any physics involved and the timeout destructor

Fire

Get Gun Position

Get Gun Rotation

Shoot Bullet Direction from gun rotation

Apply Speed

Update for 10 seconds

Destroy

Update Bullet direction + speed

### Sub-Components:

* + - * Transformation

### Dependencies:

* + - * Transformation

### Used by:

* + - * Gun

## D7 - Turret

### Description

* + - * The turret module handles and creates the turret, it gets the player position and see if they are in distance to start shooting, it then shoots at the current player position at a set speed.

Update

Get players location

If Distance is less than 50

Shoot at player

Get players bounding box

Check Turrets Bullet list against players collision

If collision is true reduce health

### Sub-Components:

* + - * Transformation
      * Shooting system

### Dependencies:

* + - * Shooting system
      * Transformation

### Used by:

* + - * Level Creation

## D8 - Player Input

### Description:

* + - * Get keyboard and mouse input from windows and complete actions based on player input

Switch (Keyboard)

Case A

Strafe Left

Case D

Strafe Right

Case W

Pitch Down

Case S

Pitch Up

Space

Shoot

### Sub-Components:

* + - * Windows SDK

### Dependencies:

* + - * Windows SDK

### Used by:

* + - * Player Input

## D9 - Ship Flight

### Description:

* + - * Part of the player functionality accurately flies and moves the ship depending on player input

If Acceleration = true

Speed = 5;

Direction += speed

### Sub-Components:

* + - * Transformation
      * Player Input

### Dependencies:

* + - * Transformation
      * Player Input

### Used by:

* + - * Player

## D10 - Corridors and Rooms

### Description

* + - * Creates Corridors And rooms

Create a floor with a scale of 5

Creates walls with the same size and offset

Get Corridor Size

Create floors for how long the corridor is

Put the corridors in the direction intended

Make sure corridor collision detection is updated

Per floor randomise if it gets a turret / target

Choose a turret / target

Randomise wall/floor/roof

### Sub-Components:

* + - * Transformation
      * Turrets
      * Targets

### Dependencies:

* + - * Transformation
      * Turrets
      * Targets

### Used by:

* + - * Player

## D11 - Text

### Description:

* + - * Converts String to Displayable text

### Sub-Components:

* + - * DXUT

### Dependencies:

* + - * DXUT

### Used by:

* + - * Game logic

# Requirements Mapping

This section details a trace-ability matrix to map the requirements to the design functionality.

Requirement

|  |  |  |  |
| --- | --- | --- | --- |
| Requirement | Design | Mandatory | Importance (0-100) |
| *F1* | *D6* | *Y* | *100* |
| F2 | D7 | Y | 100 |
| F3 | D5 | Y | 100 |
| F4 | D10 | Y | 100 |
| F5 | D10 | Y | 100 |
| F6 | D11 | Y | 100 |
| F7 | D2 | N | 70 |
| F8 | D1 | Y | 100 |

# Full Function Plan

* + This sections details all the functions needed Per class, data types will be determined when the function is created. The plan does not include data types of structures as these can be changed function by function, it does not include optional

# Maths

* + - The maths class is a more generalised lengthy class of functions to handle vector maths related to the game mechanics and objects. This is from simple operations like addition and subtraction to more complicated functions like vector lerp and forward direction function used for ship movement

# Text

* + - DXUT function that handles the text display in game

# Collision Detection

## Update

* + - * + Get position
        + Get Scale
        + Min = Position - Scale
        + Max = Position + Scale

## Intersects

* + - * + Get Box1
        + Get Box2
        + Intersect test for two box
        + If intersecting return true

# Transformation

## Draw

* + - * Get Position, Rotation, Scale,Colour Red, Green blue
      * Transform Rotate Then Scale
      * Apply Colours

# Menu

## Create Button

* + - * Location
      * Colour
      * Text
      * Button Action

## Update

* + - * If mouse is on button highlight = true
      * If highlight is on button highlight = true
      * Highlight = keyboard input W up D down

# Boundaries

## Update

* + - * Check grid.
      * Check if player is in grid
      * If player is in grid check collision detection

## Spawn Target

* + - * If number is in the percentage chance of a target being spawned target = true
      * Randomise wall
      * Offset according to wall
      * Store In memory

## Spawn Turret

* + - * If number is in the percentage chance of a turret being spawned turret = true
      * Randomise wall
      * Offset according to wall
      * Store In memory

# Bullet

## Update

* + - * Get velocity
      * Timeout Destructor
      * Transformation + Velocity
      * Check Bullet Collision
      * Set Transformation

# Target

* + - Update

If Bullet is in Grid 10

If Target Is in Grid 10

Perform Collision Detection Test

If Colliding Destroy Target

Add One to target score

# Player

* + - Update

Forward Direction = Rotation

Right Direction = Cross Product Of Forward Direction

If Q == true

Rotation.y -= 0.05;

If e == true

Rotation.y +=0.05

IF w == true

Rotation.x -= 0.05

If s == true

Rotation.x +=0.05

If A== True

Velocity.X -= Right Direction;

If D == true

Velocity.X +=Right Direction

If Space == true

Player Gun.Fire

If (Shift == true)

Velocity += Forward Direction

If (LControl == true)

Velocity -=Forward Direction

Check Against Boundaries in Grid

If Colliding Player dead == true

If Colliding in Finish Grid stop time

Next level

# Rooms

## Setup

* + - * Get Room Size
      * If 2 BY 2
      * Create 4 Floors
      * Have Walls Open
      * Close Outside Walls
      * Create Percentage Chance Of Turrets and targets
      * Setup Collision

## Update

* + - * Check Room Grid Against Players and Bullets
      * If Collided Kill Player

# Gun

* + - Setup
      * Set Gun Position
      * Set Bullet Pool Size
      * Set Gun Rotation
    - Update
      * Update Bullet Array
      * Update Position and Rotation
    - Fire
      * If bullet fired = false then
      * Get Gun Position and Rotation
      * Velocity = 10
      * Bullet fired = true

# Corridors

* + - Setup
      * Get Corridor Length
      * Get Corridor Direction
      * If Corridor Direction = left
      * Create Floor Prefabs going to left direction
      * Length = floors used
      * Max is 50
      * Have Walls Open
      * Close Outside Walls
      * Create Percentage Chance Of Turrets and targets
      * Setup Collision
    - Update
      * Check Collision Against Player

# Main (Engine)

* + - Main/Render
      * This functions handles most of the DX11 functionality, in broad terms it reads a texture from a file and stores it, the device context is set and updated in the render code where the draw functions are called.
      * DX11 handles most of the graphics API however I still need to set the device context, P3D devices and the context buffer to get the drawing working. I also need to make sure each object has a position and colour in world space otherwise the vertex buffer will not draw.