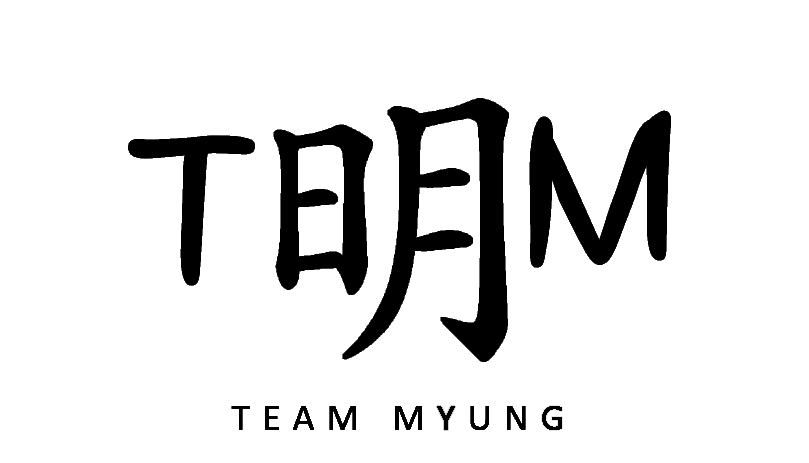
GAM150S16KR

Spring 2016

# 

<Technical Design Document>



Team Myung

Junhyung Kim (Producer)

Guesik Kim (Lead Designer)

Heejae Kwon (Technical Director)

# 

# 

# **Table of Content**

* [Cover Page](#h.6hadb4mvsf5m)
* [Table of Content](#h.mozqrdn0fc05)
* [Technical Specification Structure](#h.md051vvq0yoe)
  + [Architecture Overview](#h.kat9c3pfv4yn)
    - [User Interface](#h.vp39jkcwga28)
    - [Map](#h.i8ae8aoqk4rm)
    - [Object Manager](#h.mvsxxme7x5h1)
    - [Warp Engine Tools](#h.p2moywdaqe0o)
  + [Graphics Implementation](#h.6erz18r49zzl)
  + [Physics Implementation](#h.d77u58v5bar8)
    - [Ball’s movement](#h.vbwwq8hbiqkr)
    - [Collision & Reflection](#h.j50ifiz08s8q)
  + [Player Controls Implementation](#h.pwajlxhlutf7)
  + [Behavior Implementation](#h.rmwy3ki1v3xe)
  + [Coding Methods](#h.zd16fzxi3hfg)
  + [Debugging](#h.j1fuowpvjr8k)
  + [Tools](#h.419ir22dqmvj)
  + [Scripting Languages](#h.awguswlp5db8)
  + [Technical Risks](#h.uy9tsx4y9ecc)
* [Appendices](#h.y1ly1fgavd26)
  + [Appendix A: Interface Flow](#h.rsanp8j7lfg5)
    - [Flowchart](#h.8onq7cgmjkbz)
    - [Mockups](#h.sqtm5hffieh5)
  + [Appendix B: Art Requirements](#h.mlg1g36h4r12)
  + [Appendix C: Audio Requirements](#h.3059ra560o4w)

# 

# **Technical Specification Structure**

## ***Architecture Overview***

### User Interface

User Interface helps game play without text explanation. Simple images will be used to let the player understand and have an idea what it will be when the player gives input.

### Map

There are obstacles placed on every levels beside ball and player created boxes. Each obstacles are placed by codes and locations based on screen width and height.

### Object Manager

Object manager, manages all objects in game. It includes balls, buttons and blocks. Each object has properties like scale, position and colors. At loading states, object manager sends object’s information to draw objects.

### 

### Warp Engine Tools

Particles, graphics, audio and physics are made based on the Warp Engine. Physics engine is the most significant component in game. Using dot product and vector calculation, physics engine enables reflection of the ball which is the main game play.

## ***Graphics Implementation***

In the Warp Engine, there are functions that help drawing objects, sprites and background. To express sparkling effects, particle systems will be made with group of small sprites and random movements around the ball.

## ***Physics Implementation***

### **Ball’s movement**

* Ball has a velocity and acceleration.
* Ball’s position is updated every frame as much as velocity.
* To shoot the ball, get mouse cursor position to choose the shooting direction.

### **Collision & Reflection**

In the Warp Engine, there is a function that notifies the intersection between circle and line. In every frame, intersection between ball and each rectangle’s edge uses that function. In case of reflection, calculate the vector of the ball and the lines to calculate the reflection vector and implement the ball’s reaction when collide.

## ***Player Controls Implementation***

The Warp Engine supplies functions which check basic inputs. Warp Engine has detailed mouse click functions which enables getting the input click, pressed and double click. There is no way to check whether the mouse is unplugged or not.

## ***Behavior Implementation***

All the objects’ behaviors are chosen by the location of the mouse cursor. The degree of the shooting is chosen by the location of the mouse cursor and the player can move the boxes by dragging.

## 

## ***Coding Methods***

* + File Name: NeonSign\_Number
  + File Location :
* Source files in the ‘Source’ folder
* Texture files in the ‘Textures’ folder
  + Source Code Managing
* SVN: Digipen provided tool for version control.
  + All header files include #ifndef, #define, #endif

## ***Debugging***

* + Debug console which is supported by Warp Engine.
  + Breakpoint which is supported by Microsoft Visual Studio 2013.

Above 2 tools are main debugging tools for our project.

## ***Tools***

* + Microsoft Visual Studio 2013
  + Paint.net
  + Audacity
  + SVN
  + Google Docs

## ***Scripting Languages***

* Not applicable in our project.

## ***Technical Risks***

Risk 01: Reflection on corners

Solution 01: Develop more precise mathematical codes.

Risk 02: Rotating boxes

Solution 02: Study how to use radians in the code.

Risk 03: Tracking path of the ball

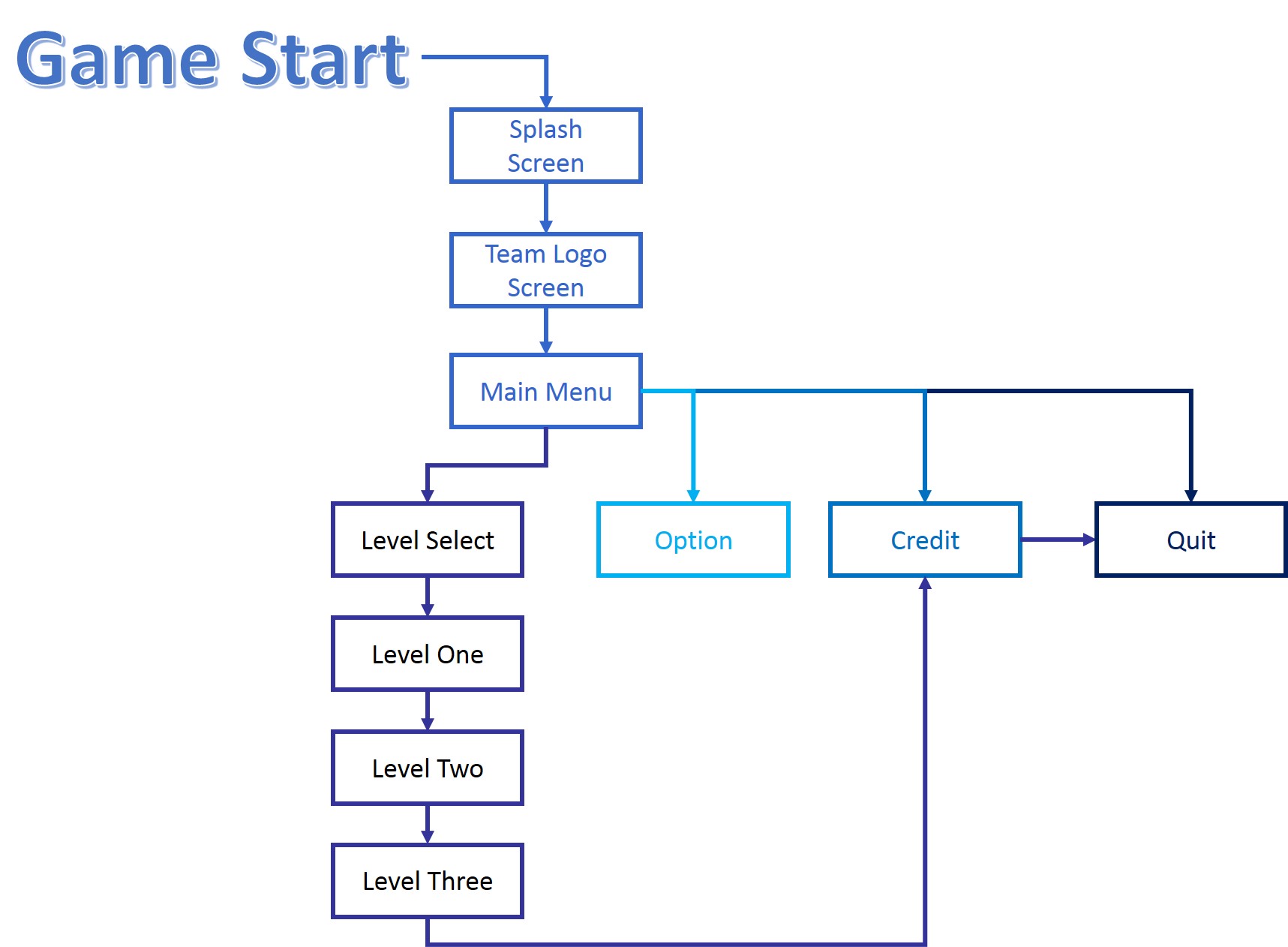
Solution 03: Using graphics, draw the path of the shot ball.

# 

# **Appendices**

## ***Appendix A: Interface Flow***

### **Flowchart**



### 

### 

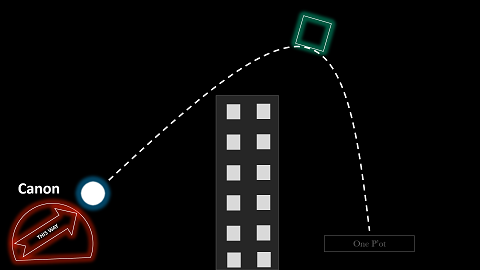
### **Mockups**



<Team Logo>



<Main Menu>



<Playable Level>



<Credit>



<Digipen splash screen>

## ***Appendix B: Art Requirements***

All the sprites are made by team Myung members, using shape tools in the paint.net with tga format. All sources are located inside the Textures folder of the project file. File names start with capital letters.

## ***Appendix C: Audio Requirements***

All the audio assets are from Digipen libraries. Background music files have name starting with‘BGM\_’ and sound effect files with ‘SFX\_’. All the sources are located in the Audio folder of the project file with wav format.