Tongji - Development

**Technical Design Document**

(Version 3)

2015/5/15

Table of Contents

1. Overview 3

2. Requirements 3

2.1. Brief 3

2.2. Reference 3

3. Dependencies 3

4. Existing Technology 4

4.1. Features 4

4.2. Reference 4

5. Implementation details 4

5.1. Map Editor 4

5.2. Level loading 4

5.3. Element Design 4

5.4. AI Design 4

5.5. Sound Develop 4

5.6. Animation Develop 4

6. Proof 5

7. Issues 5

8. Risks 6

9. Estimates 6

# Overview

This system is a game based on the Unity3d engine. The game’s concept is according to Bomber Man and we’re trying to make it more interesting and funnier. Our goals are to finish the whole game including 3 levels which should be harder by each level. The purpose of this game is of course to make people love playing it and to finish our course project.

Differ from other bomber man game, this game will much more like a fps game, such as PROTOL and COD, which means, player will not only player this game, but also enjoy the plot.

# Requirements

## Brief

* The game is based on Bomber man but we will recreate the background and make the game in 3D
* The game will be developed using Unity3D
* The game will developed on script which is will make the game like a story.
* The story script will be as an enclose with TDD.
* Whole the game also should have a map editor which will make the map design more easier.
* Map information should contain the element position player life, tool rate, map visibility.
* The scene should be reused, each level will be loaded on the same “room”.
* Game element will contain “Canon floor”, “Breakable cube”, “Unbreakable cube”, “Robot”, “Player”, “Tools”. All the elements will define in Implementation Detail.
* Depended on different level we should design different AI.
* Sound and scene change animation also should be design.

## Reference

* We want to recreate the playing method to make the game played not in an original way
* As we are not good in constructing 3D models, we are planning to search for some free 3d models and music on the Internet and use them
* All the technical reference will find in Virtuos 2015 game course PPT.

# Dependencies

* As our group don’t have much developing experence in 3D game area, we will try to find some examples or instructions to help us learn better and faster about the whole process
* One of our group members has been learning about Unity3D but we are all beginning learners. We are trying to find a faster way to write the codes correctly, such as some open-sourced codes on which we can just modify.
* We’re not sure whether 3D models are hard to find. We can build the gaming scenes and bombs by ourselves

# Existing Technology

## Features

qqtang

Gaming basics

* Our game style will be generated based on the game mode of qqtang and added with some of our original ideas to make it different.

Characters&Bomb appearences

* There are lots of characters and different bomb appearences in qqtang, we’ll consider to make several models so that players can choose.

Counter Strike

First-person

* It is fun to think if we play bomberman in first-person sight. So we are thinking how to roll the camera.

## Reference

* Some tutorial materials about Maya, Unity3D on the Internet
* Open-sourced game projects built basing on Unity3D

# Implementation details

<Feature>

<This is the meat of the TDD. It includes whatever you need to describe this feature. It can be as long as it needs to be. Feel free to include, suda-code, real code, UML, whatever you need along with paragraphs of text describing how the feature will be implemented. Obviously on smaller features and in smaller TDDs this section might be really small as well.>

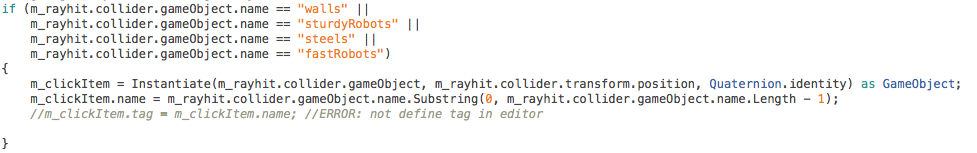
## Map Editor

Description:

This part is focus on give programmer or designer a tool to build a map quickly and obviously.

Function:

Drag: User could drag the cube to the map, and editor will mark the position and the cube type of it. The code like this:

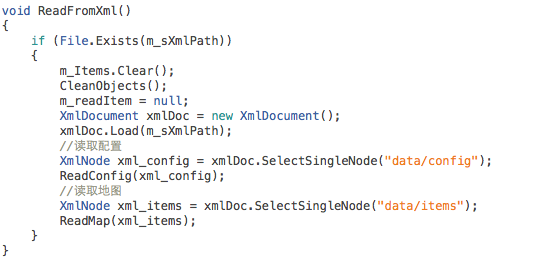


Save: Map should be save as normal file such as XML or some thing else.

the code like this:



Load: When user wants to redesign the map it could load the map. the code like this:



## Level Loading

There are 3 level of map, so we should load level or map in same scene, we build a script like this , it just like load map in map editor. But we change the element in it , because map editor is a 2D while bomber man is 3D. All the elements are different.



And the frame work of map is just like this:



## Element Design

We have such elements in the game:

* Canon floor: Instead of bomber, canon could shot to four direction and could rise and fall. Hence there are at least 4 part in class Canon

***ChangeMateiral() could change material when mouse point to it.***

***Rise(int time) rise function***

***Fall(int time) fall function***

***Shot(int time, int distance) shot function***

* Breakable cube: those cube could be destroyed and produce tools. Hence the main function of it is :

***Break(float toolRate) paly the destroy animation and build new tool in current position.***

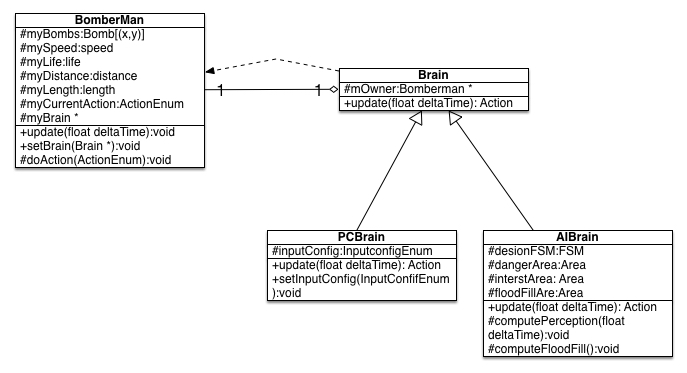
* Unbreakable cube: this is unbreakable cube with no function or class.
* Tools: we have Faster, Life+1,Longer,Far those 4 tools, every tools will change player’s attribute directly.
* Elevator: when player finish one level he/she will ride elevator to go down to next level. So the function of it are:

***Show() elevator show up***

***Run() go down to next level***

## AI Design

AI is a most important part in the game. We use UML to design it:



**Memory Implications:**

<How will this effect Memory, it may not>

**Performance Implications:**

<How will this effect Performance, it may not>

**Networking Implications:**

<How will this effect Networking, it may not>

|  |
| --- |
| **Hint –** <These things may be needed to think. >  Thread usage (single/multi-threaded)  Supported platforms (PC, 360, PS3, PSP, WII…) |

# Proof

* <Prove the system is working correctly. You can design test case, unit test here. You can also prove it by description, values, or other ways.>

<Example1 – math optimization>

< Unit Test – write unit test for sqrt function… >

<Example2 – Shadows>

< Test Case - We can see smooth shadow on ground that casted by cars and dynamic objects... >

<Example3 – Bullet optimization>

< Verify - After bullet optimized, the function World::stepSimulation() calling time will be in 14ms. >

# Issues

* <Bullet points on any issues for this task.>

# Risks

* <Bullet points on risk, mitigations and contingency plans> <Chance of Risk> <General Time Estimate if risk happens>

# Estimates

Task time table is in folder.

|  |
| --- |
| Note – Try to make document simple, clean and focus on important parts. |