Tongji - Development

**Technical Design Document**

(Version 2)

Table of Contents

[1. Overview 3](#_Toc260387709)

[2. Requirements 3](#_Toc260387710)

[2.1. Brief 3](#_Toc260387711)

[2.2. Reference 3](#_Toc260387712)

[3. Dependencies 3](#_Toc260387713)

[4. Existing Technology 4](#_Toc260387714)

[4.1. Features 4](#_Toc260387715)

[4.2. Reference 4](#_Toc260387716)

[5. Implementation details 4](#_Toc260387717)

[6. Proof 5](#_Toc260387718)

[7. Issues 5](#_Toc260387719)

[8. Risks 6](#_Toc260387720)

[9. Estimates 6](#_Toc260387721)

# Overview

To validate the “Virtuos Game Development Course” we will have to develop a video game running on PC under Windows. The game is a 3D version of the classical “Bomberman” video game. This game is developed by our team, consist of 4 students. We design the game just to show the harvest of the whole semester.

# Requirements

## Brief

* We design the game as a PVE mode. The player can cause damage to his opponent by setting bombs. At the same time, for different bomb, it has different attack range, different damage value and different functions.
* The behavior the player operates must be in accordance with a specially rhythm. Only turning to the beat can the behavior take place. We use the publish-subscribe pattern to make the interfaces of the behavior wait in a queue, and complete the behavior when the beat comes.
* To make our project develop in the future, we use interface mode. We do our best to use more interfaces
* When the bomb explode, we make some special efficiency. At the same time, we make lots of buff to increase the visual effect.

## Reference

* Game Development Course from Virtuos – Project
* <software architecture>
* A PC game: *Crypt of the NecroDancer*

# Dependencies

* Game Development Course from Virtuos – Project
* PPT of the courses of the Game Development Course
* Teachers of Virtuos
* The tutorial of the unity on the official website
* Reference books , such as <software architecture>

# Existing Technology

## Features

Publish-subscribe pattern

* Because our game needs the player to control the game characters on the beat. The time of the bomb exploding also allows to the specially rhythm. So as to response the beat properly, we use the publish-subscribe pattern. The player controls the character to do some behavior and the system delivers the interface to the queue. If the behavior is in the range of the rhythm, the system will call the interface of this behavior. In this way, when the player do some behavior, the system doesn’t need to wait for the beat with doing nothing. Just waiting in the queue, when the beat comes, calling the interface is okay.

Interface mode

* <Feature bullet points, what it is in this feature we’ll be using>
* Because we do our project in 3 months, we don’t have enough time to complete all the functions. As to make the project develop in the future, we deliver the data in the form of interface.

Particle system

* As we use unity to complete our game, we will use some special efficiency , particle system is a good example. When the bomb explode, we use particle system to make the fire look more real.

UGUI

* We use UGUI to make the UI better. With the help of UGUI, the design of UI becomes easier and nicer.

## Reference

* The tutorial of the unity on the official website
* Reference books , such as <software architecture>

# Implementation details

At first, we use MapDataHelper to create the map, NPC and character. To record the position of the character, NPC and bomb, we use the interface locatable to save the position in GameDataProcessor. In this way, it is easy to deal with the data we might use in the future.

For controlling AI, we use the algorithm of A star. In this way, we calculate the shortest way and save the final location in locatable and send the data to the system.

At the same time, we build an interest map and according to the things around NPC to set different weight. In this way, the NPC can walk reasonably.

For players, we use publish-subscribe pattern to make sure the player/AI can move only on the beat. What’s more, the player can get skills, for example, walking through the wall. In normal situation, the player’s property of throughWall is false; if he get the skill, that value is true.

For bomb, it has different levels. For different levels, we must save the attackValue in the function of destroy, the person setting this bomb in the class of atackby, and other values in different classes. When the bomb explodes, we use particle system to set fie on the square according the range of the bomb. If there is a wall in that range, there shouldn’t be any other fire in the later squares in that direction.

For rhythm, we use the interface of RhythmObservers to save the queue of the action of AI. When the beat comes, the AI complete the behaviors. We use the interface of RhythmFlag to save the queue of the behavior of the player. We allow that the operation of the player can delay or advance for specially time. Only in this range can the interface of these behavior realize.

Finally, we use GameManager to control game state and change between them, the Game State is include: GameStart, LevelEdit, LevelEnd, Playing, GameEnd.

# 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 – map testing>

We will use a character to walk in the map. We will test whether the character can walk out o the map, or walk through the wall.

<Example2 – bomb testing>

We will test the bombs with different levels. We must make sure the range of attacking, the attack value, and the fire special efficiency are all the right.

<Example3 – HP testing>

Use different level of the bombs to attack the character and NPC. Make sure the HP is right.

<Example4 – buff testing>

Let the character get different kind of buff and test whether he get specially skill.

# Issues

* In April: the realize of the map
* In May: the realize of the basic functions, such as the bomb exploding, the HP of the player, the control of AI, the use of rhythm.
* In June: increasing of specially efficiency, such as bomb fire, realize of buff, and specially skills.

# Risks

|  |  |  |  |
| --- | --- | --- | --- |
| Risk | Chance | Solve | Time |
| Map management, for example, because of the data in memory going wrong, the player walk out of the map | low | Destroy the player and rebuild the map and start again | 100ms |
| The normal bomb explode through the wall or attack the wrong value | middle | If the mistake makes the game over, restart the game; if not, make up the mistake | 200ms |
| When the beat comes, the system doesn’t complete the behavior of the interface. | low | Make where the problem happen, and solve them | Little long |

# Estimates

|  |  |
| --- | --- |
| Tasks | Estimate in days/person |
| The build of the map, the set of AI | 12 days/马文韬 |
| The build, use of the rhythm | 10 days/刘道畅 |
| The realize of the bomb and attack | 13 days/赵世雨 |
| The realize of thee player, such as HP, and skill | 9 days/王浩旭 |
| Total | 44 days |