

**Technical Design Document**

(v1.0)

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

This system, UniBOOM, is a bomberman game. It shall let players, as Unity-chan in the game, experience a classic bomberman gameplay, with some innovative features.

The system shall include all typical bomberman gameplay, such as moving in the stages, setting up bombs, break walls, killing enemies/opponents, and getting bonus items.

Besides, the system shall provide players some action game features and role playing game features, such as playing this game in third-person perspective, moving at different speed, minimaps, detecting enemies/components and changing equipment/skill points.

The system shall also provide a stage editor, which can make custom stages to play. Stage data shall be saved as files in hard disks.

# Requirements

## Brief

* Pre-game Features
  + Setting up games, including graphics, audio, etc.
  + Choosing levels which have been played before
* Game Features
  + Moving in the stages
  + Deploying bombs
  + Bombs explode and generate damage zones
  + Destroying blocks (using bombs)
  + Damage enemies/self
  + Item drop from blocks
  + Picking up items and being granted abilities
  + Recording scores for players
  + Enemy AI
  + HP bars for player
  + Moving at different speed (upon holding different buttons)
  + Pausing, exiting or restarting game

# Dependencies

* Unity documentation (<http://unity3d.com/learn/tutorials/modules>)

# Implementation details

Pre-game Features

This part describes an overview of the implementation of pre-game features.

Most features here are menu matters. Unity UI System will be used to implement those features.

Main Menu:

* New Game
* Load Game
* Option
* Quit

Upon entering “Load Game”:

* Stage 1/2/3/4/5/6
* Back

Upon entering “Option”:

* BGM (Scroll Bar)
* SE (Scroll Bar)
* Back

Game Features

This part describes the implementation of all game features.

Character model

Use “sd\_unitychan\_generic” in “SDUnityChan” package as base.

Remove **all script components** **expect**:

* SpringManager
* AutoBlinkforSD
* RandomWind

Add following components:

* Rigidbody (Collision check)
* Audio Source
* Unitychan
* SE Controller

Create a new controller “Unitychan” for Animator.

The Animator contains following states:

* Stand
* Run
* Salute
* Damaged
* GoDown (Death animation 1)
* KneelDown (Death animation 1)

Controlling

Player use keyboard to control character, mouse to control perspective.

Available controls:

* Looking (Mouse X/Mouse Y)
* Moving (Horizontal/Vertical)
* Dashing (Dash)
* Deploying bombs (Fire)
* Pause (“Escape” button)

Controlling is splitted to 3 layers:

* Reading raw input data (UnityEngine.Input, Axis data/Hard buttons)
* Convert raw input to control state (Enums or Booleans)
* Process control state

Bomb related

When player deploys a bomb, Instantiate a bomb where the character stands. Bombs are always generated at “Integer points” (In this game, this means an integer + 0.5)

Bombs will explode in about 4 seconds after deployment. The explosion Instantiates “Blast” in 4 directions.

Blast will continue to spread toward its direction if there is no block/brick in its way. It will be stopped if encountering a block/brick (brick will be destroyed).

Blast check a list in “Room” to find what is on its way.

Blast disappears after about 2 seconds.

Blast is attached with a box collider, which would have collision with character/enemy.

**Performance Implications:**

The blast effect may be very consuming. Make sure it does not affect the performance too much.

Camera

The camera is in third-person perspective, which means it orbits character when moving mouse.

The “Main\_Camera” is attached with script ThirdPersonCamera.

Suppose “radius” is the distance between camera and Unitychan, “theta” is the included angle between the line of sight and horizontal plane, “phi” is the rotation angle.

Use the following formula to calculate position:

position = (radius \* sin(theta) \* sin(phi), radius \* cos(theta), radius \* sin(theta) \* cos(phi)) + unitychan.position

Radius and theta is confined in sections.

Map Generation

The map is generated upon entering the stage.

Generation unit is called “Branch”. On each step, each branch will spread one room to one direction except the backward one. Step counts are defined by the “depth” of each branch.

One branch is tagged as “Boss”, which will generate a boss room at the end.

Each map consists of 4 branches on each direction.

Room Generation

Rooms in stages are generated upon entering the stage.

Upon generation, one pattern is randomly picked from a pattern-pool, which is pre-defined in a CSV file and read beforehand.

All breakable bricks are generated upon a probability.

The Num.0 pattern is specifically used by boss room.

**Performance Implications:**

File reading procedure exists. Make sure all CSV are read beforehand.

Room Data Record & Process

The following room data should be recorded during game:

* Map grid (What is on every position, Transform/null)
* Is Room Clear
* Door Transform List
* Enemy Transform List

Rooms have following functions:

* Open/Close doors
* Check enemy status

Brick

Brick will call Destroy() and Instantiate Wreck at the same time upon destroyed by Blast.

Enemy Model

Use raw model as base. Make sure the parent object does not contain any meshes.

Add following components to the parent object:

* Animator
* Rigidbody
* Box Collider
* Enemy Body
* Enemy AI

Enemy AI & Body

Enemy AI has 2 basic parameters:

* Speed
* Turn Tendency (Probability of taking different action)

AI has 3 types:

* Dumb, whose action is completely random
* Wandering, which will find available route
* Auto Tracing, which will chase enemy.

Use Flooding algorithm to implement Auto Tracing AI.

The AI **cannot** control the actual movement. It can send messages to Enemy Body, which can modify the parameter of Rigidbody.

Directors

There are 4 Directors:

* Stage Director
* Map Director
* UI Director
* Audio Director

Stage Director is the overall director of one stage. It has the following functions:

* Modify the state of the game
* Pause/Resume the game
* Manage available item drop
* Change active room
* Change Stage

Map Director is responsible for generating the map and rooms. See “Map Generation” and “Room Generation” for details.

UI Director has access to all UI Objects.

Audio Director can control the volume of BGM and SE.

UI

Use Unity’s UI System to make UI.

Use Unity’s Animation System to make UI Animations.

Inter-Object communication

Most active object may have communication with Stage Director. Make sure they have a reference of Stage Director.

Stage Director has a delegate called EventHandler, and has 2 event EventHandler called OnPauseGameEvent, OnResumeGameEvent. They send message to all objects that related to pausing and resuming.

# Issues

UI System & Animation System

The group is not very familiar with the UI System and Animation System. May take some time to learn.

C# Delegate

The programmer is not very familiar with C# Delegate. May take some time to learn.

Uniformity of model

Make sure the size, format of models and textures are uniform.

# Risks

Project data corruption

Data may be corrupted if wrong operations were done.

Make sure copies on every phase are backed-up.

If corruption happens, recover the project to the last phase as soon as possible.

Model data corruption

Model data corruption should be specifically listed. The risk is similar to the overall data corruption, but model data is harder to recover. On the other hand, this can be prevented much more ahead.

Check the model output of Maya when each model is complete. Do not close the current scene in Maya before the output is checked.

If corruption happens, remake the model as soon as possible.

Group member absence

Group members may be absent due to various reasons.

If the risk happens, report status as soon as possible. Assign job to other people. If necessary, cut down some of the sub functions.