BH-STG: Bullet Hell Shooting Game Parody

<< *One Piece* >>*[1]* Theme 2D non-Tile-Based Shooting Game

Team Wow
Junhao Zhang “Freddie”
Ran Tao
Alexander Lao
Chao Zheng

CptS 487 Software Design and Architecture

Fall 2017

Instructor: Bolong (Bob) Zeng

TABLE OF CONTENTS

I. Introduction

A 2D shooting game written in C# by using the MonoGame engine on WPF, that player controls “Luffy” (the main character in the Japanese Cartoon << *One Piece* >>*[1]*, also the character that player controls in this game) and shooting bullets to defeat rivals and dodging enemies’ attacks.

This game has one mid-boss with one stage and one final boss with multiple stages, and waves of minions that will be generated and live for specific amount of time based on the script the team planned ahead.

II.1 Background, Announcement and Related Work

The team started this project from ground. With the self-invented well organized structure and intelligent algorithm to track interactions among all in-game objects with others, every single in-game object individually exists and is independent to another. They know their life cycles in the game and what actions to take at the specific time. The team made this project easy to be extended, and convenient to be organized.

Special thanks to the following that provided part of or all beautifully designed materials that the team used in this project:

- Baidu Cloud (https://pan.baidu.com/share/init?surl=eXAHK)
- Pinterest (https://www.pinterest.com/1818rose/grille-croquis-point-de-croix/)
- Pinterest (https://www.pinterest.com/pin/493214596670575234)

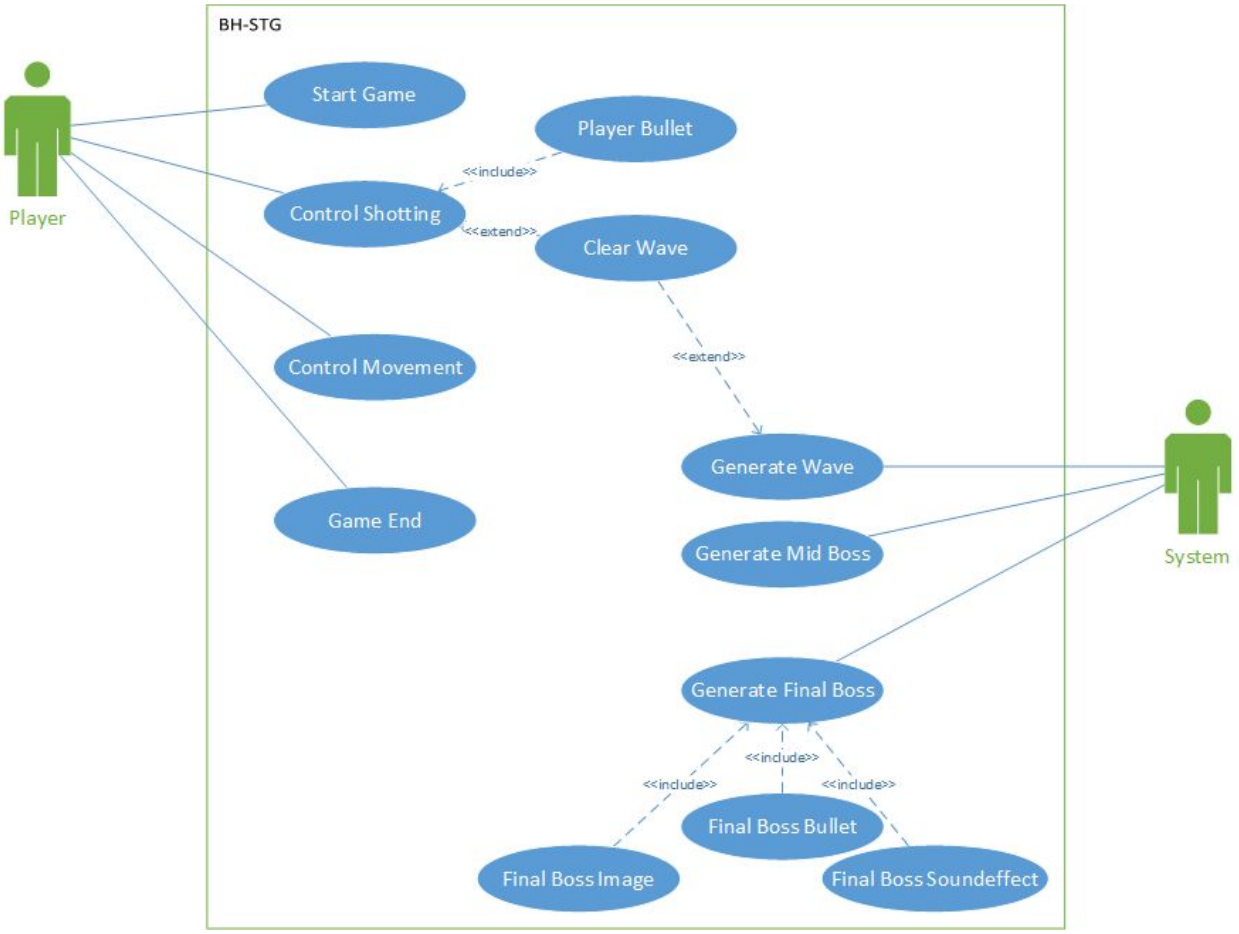
II. System Requirements Specification

II.1. Project Stakeholders

Stakeholder	***Major Value***	***Attitudes***	***Major Interests***	***Constraints***
Clients	*Use of Product*	*N/A*	*Polished, feature laden product, Easy and safe to use*	*N/A*
Development Team	*Job Security, Experience*	*N/A*	*Extensible, maintainable application*	*N/A*
Project Supervisors	*Competency Assessment Purposes*	*N/A*	*Standards compliant, extensible software*	*N/A*

II.2. Use Cases

User: The user is any human being that controls the player character of the game.
System: The System is defined as the Front-End components and Back-End logic.



II.3. Functional Requirements

II.3.1 Generate Enemies (waves of minions, bosses, etc.) based on the time elapsed.
Description: Generate specific object(s) based on the absolute game time elapsed.
Priority : High

II.3.2 Logic to decide interactions among the currently existing objects in the game
Description: At each time the game updates, need to decide the interactions between currently existing objects in the game such as: hitted by bullets, cross-over enemies, etc.
Priority : High

II.3.3 Interpreter Design
Description: Be able to composite and produce the required objects based on the pre-defined requirements.
Priority : Medium

II.3.4 Game Interface Design
Description: Practical game interface which will be able to display the status (points, lives, time elapsed, etc.).
Priority : Low

II.4. Non-Functional Requirements

II.4.1. Operation

II.4.1.1. Operation Speed of All Game Components

Description: The overall response time of the game should be reasonably quick under the most environments. The operations of all game components should be straightforward rather than causing endless loop. Maximum time-complexity of each game update should be O(N) which N is the total objects currently exist in the game.

II.4.2. Software

II.4.2.1. Windows

Description: The operating system of DirectX will run on.

II.4.2.2. MonoGame

Description: The update frequency should be more than one hundred time per second. Should not Synchronize With Vertical Retrace.

II.4.3. Hardware

II.4.3.1. Monitor

Description: The minimum resolution should be 800 * 600.

II.4.3.2. Physical Input Device

Description: Input devices (physical keyboard, controller, etc.) should generate at least seven different stdin signals (for up/down/left/right/slow/shoot/exit).

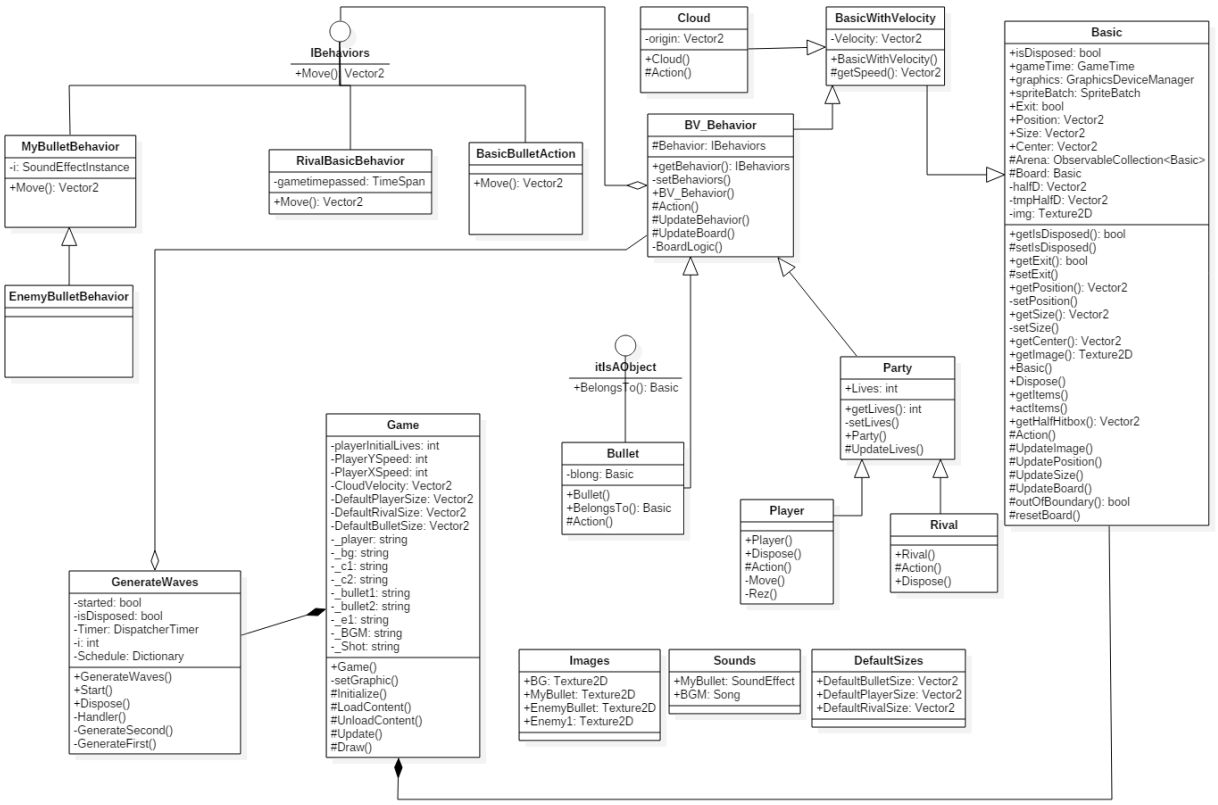
III. Software Design

III.1. Architecture Design

III.1.1. Overview

When an object was constructed, it will be automatically added to the game. If the object's position is out of the boundaries of the game (camera view), or its life cycle had been terminated because certain conditions had been reached (such as, running out of lives, being hit) then it will be disposed from the game. Therefore, every object in the game is independent individual, and they know their life cycles in game and the actions they should take (update statuses, construct new objects, dispose themselves from the game, etc.) at each update time.

Since every object in the game was independent individual, therefore there are strict rules to define their hierarchies and relationships. Shown as following:



III.1.2. Subsystem Decomposition

III.1.2.1 Front-End

III.1.2.1.1 “IBehavior”

Description: Make replaceable behaviors of Player/Minions/Bosses/Bullets to move, shoot bullets, generate other objects or special effects and background music.

III.1.2.1.2 Hitbox

Description: Box(es) belongs to Player/Minions/Bosses that if hitted then the owner of the box(es) decrement HP/Lives. Player has one hitbox that needs to be displayed always but non-players have hitboxes over their regions. The size of a single hitbox should be the size of the smallest object ever existed in game.

III.1.2.1.3 Player

Description: Basic behaviors are up/down/left/right/slow/shoot. Has one hitbox that needs to be displayed always. If it was hitted, then its number of lives will be decremented and its position will be reseted to the original location where it was born (middle-bottom of the game camera) and immune damage for period of time. If it’s number of lives is equal to zero, game over.

Contains property “IBehavior” to extend its behavior.

III.1.2.1.4 Rival (Minion, (Mid/Final) Boss)

Description: Contains property “IBehavior” to extend its behavior. Has hitboxes over its region. If it was hitted, then its number of lives will be decremented, and if it’s number of lives is equal to zero or it moves out of the boundaries of game (camera view), it will be removed from game.

III.1.2.1.5 Bullets

Description: Contains property “IBehavior” to extend its behavior. Doesn’t have hitbox or lives. Needs to know its owner in order to decide what to do when it cross-over any others. If it moves out of the boundaries of game (camera view), it will be removed from game.

III.1.2.1.6 Belonging

Description: Needs to know its owner, and the only behavior is updating its position if it’s owner moves. If it’s owner had been disposed from the game, it also needs to be disposed.

III.1.2.2 Back-End

III.1.2.2.1 Logic Engine

Description: At each time the game updates, need to decide the what action to take if any of those in-game objects cross-over another. The logic is:

- If Player’s hitbox cross-over the any of the hitboxes of Enemy or Enemy’s bullet, Player is hitted.
- If any of the hitboxes of Minions or Bosses cross-over the hitboxes of Player’s bullets, that Minion or Boss was hitted.
- If any of the hitboxes of one bullet cross-over another bullet’s hitbox, then replace.
- Otherwise, do nothing..

Concepts and Algorithms Generated: The algorithm is, first we have a 2D array to track all in-game bullets, player, enemies and bosses. The dimensions of such array should be (the width of game camera divided by width of a single hitbox)* (the height of game camera divided by height of a single hitbox). Then, at each time the game needs to update all its in-game objects, compress the in-game bullets, player, enemies and bosses based on the hitbox size to one point if is Player or several points if is not Player. If multiple points of different objects locate on one same slot of the array, decide what to do based on the logic in Description. This way, we keep the time complexity less than O(M*N), which M = the width of game camera divided by width of a single hitbox, and N = the height of game camera divided by height of a single hitbox.

III.1.2.2.2 Scheduler

Description: Generate specific characters based on the total game time elapsed.

Concepts and Algorithms Generated: Generate waves or specific characters (Minions, Mid/Final Bosses) based on the script the team planned ahead. Start to time at the moment when the game was first launched and start to generate required character(s) if the it is the scheduled time.

IV. Glossary

MonoGame - An open-source software used to make Windows and Windows Phone games.It implements the Microsoft XNA Four.
WPF - Stands for Windows Presentation Foundation. Details at: "Windows Presentation Foundation." Wikipedia. September 02, 2017. Accessed September 22, 2017. https://en.wikipedia.org/wiki/Windows_Presentation_Foundation.

V. References

[1] "One Piece." Wikipedia. September 21, 2017. Accessed September 22, 2017. https://en.wikipedia.org/wiki/One_Piece.