The Design Document for CSEE 4840 Embedded System Design:

Bullet Hell Game: Bad Bird!

Xinye Jiang (xj2253)
Po-Cheng Liu (pl2812)
Spring 2022

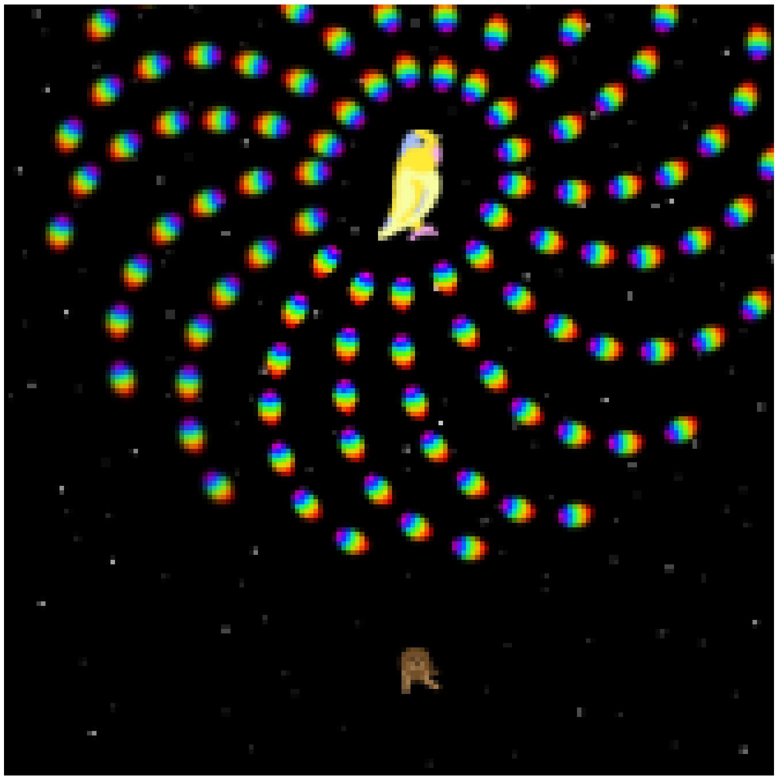
Contents

1 Introduction

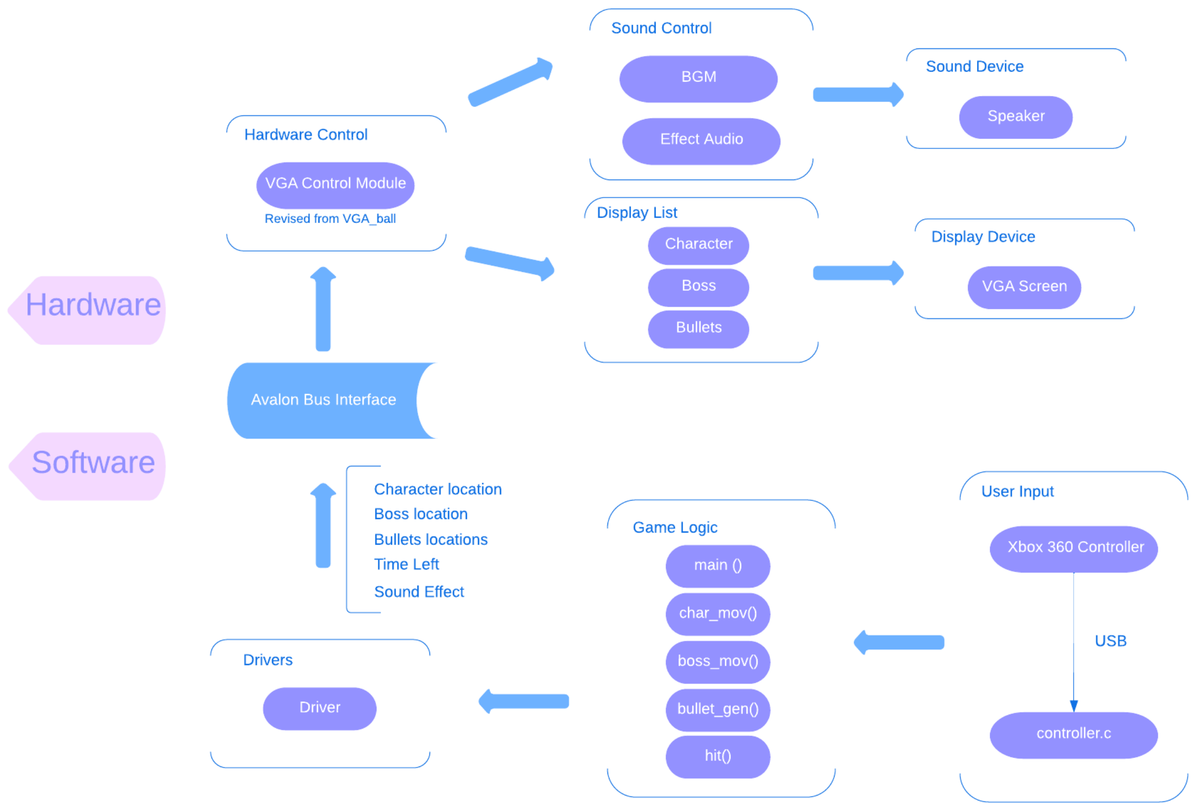
Our project is to create a pixelated bullet-hell-shoot' em-up game. It provides a gameplay of dodging the bullets from the 2D enemies. Just like the famous danmaku game *Touhou Project*, our game would have enemies that shoot 2D bullets at the player while the player would try their best to avoid getting hit by the rains of bullets.

Our game allows users to control the playable character with a Xbox 360 controller for its movements, and give feedback if the character is hit with any of the bullets. The goal of the game is to survive. So long as the player lasts through the time limit set without getting hit at all, the player wins. The player can also choose to fight back with their own bullets. If the player eliminates the enemies before the time limit set for each enemy, the player also wins.

The game has colorful pixelated graphs of enemies, player, and the bullets, and has different sound effects as well as a main background music.


Sample Game Graphics Drawn with Photoshop

2 System Block Diagram



3 Algorithms

- Boss action:

Attack patterns are predesigned and may be stochastic (generate bullets at random direction).

```
If pattern is not selected:
    Select an attack pattern by random.
Else:
    Play the predesigned action from the pattern.
    Ex:  time  action
         0     generate bullet vertically
         1     move up
         2     move left
         3     generate bullets at 45 degrees
         6     generate bullets with outward spirals
         9     generate 5 hollow circles of bullets falling down
         12    generate a ball of bullets bouncing around
         13    end of the pattern, select a new one
```

- Player action:

Move in up, down, left, right.
Attack: generate bullet vertically up.

Bullets movement:

```
For every bullet:
    Properties:
        Damage, position, velocity, current_direction, A, B, C, D
    Movement every timestep:
        Position change is defined by the velocity and current_direction.
        In order to create curved path, update current_direction by the equation:
            Current_direction = A + B*time + C*sin(D*time)
        Use the current direction to select the correct bullet figure to show.
```

- Collision detection:

```
Loop for all boss' bullets:
    If it hits the player, decrease the player's health for 0.5 or 1 depending on bullets' types (3 from start).

Loop for all player's bullets:
    If it hits the boss, decrease boss' health by 1 (100 from start).
```

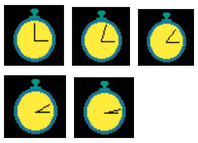
Bullets from player and boss will not cancel each other. They overlay on the display with the boss' bullets on top.

- Result:

A player wins if the boss' health decreases to 0 or time out.
A player loses if the player's health decreases to 0 before a win.

4 Resource Budgets

(1) Display Memory Budget

Objects	Graphics	Size (pixel)	Number	Total Size (bit)
Boss		40 * 25	3	72000
Player		30 * 30	3	64800
Bullet 1		25 * 25	4	60000
Bullet 2		25 * 25	4	60000
Bullet 3		14 * 13	1	4368
Player's Health		15 * 15	3	16200
Timer		20 * 20	5	48000
Result	YOU WIN YOU LOSE	60 * 50	2	144000
Total				**469368**

(2) Audio Memory Budget

Objects	Time (sec)	Frequency (kHz)	Size (bit)
Background Music	38	8	12050000
Bullet Generate	0.2	8	28800
Bullet Hit	0.45	8	56000
Total			**12135400**

For both display and audio, we need a total of 12,604,768 bits (1.6 MB) of memory.

5 Hardware/Software Interface

	Size (bits)	description
Time	8	Time in seconds
Player health	8	
Player position	20	10 for X, 10 for Y
Player sprite number	3	Select which player’s figure to show
Boss health	8	
Boss position	20	10 for X, 10 for Y
Boss sprite number	3	Select which boss’ figure to show
Number of bullets	8	Current number of bullets existed
Bullet[N] position	20	N=0:255
Bullet[N] sprite number	4	N=0:255