

COMP7604 Game design and development

Group 11

Doge's Adventure

Chen Yuanfei, Du Tianyi, Lai Shangqi, Xia Fan

1 BRIEF DESCRIPTION

1.1 STORY

It's said that there is an evil and sequester laboratory in HKU. Over years, they did horrible experiments on animals to make war machines. One day, an animal named Doge escaped from the lab. And this game tells the story of Doge's revenge.

1.2 GAME LOGIC

This game is a typical FPS (First-person Shooter) game. In the game, player will play the role of Doge, controlling her to explore the campus, and defeat all the enemies.

1.3 GAME CONTROL

W/A/S/D : move (forward / left / back / right)

Space : jump

Left mouse : shoot

Mousemove : change view direction

Shift : hold to run

Dogecoin : collect to get score

Green rune : stay to restore HP

Yellow rune : pass to add attack

2 GAME TECHNIQUES/TECHNIQUE DETAIL

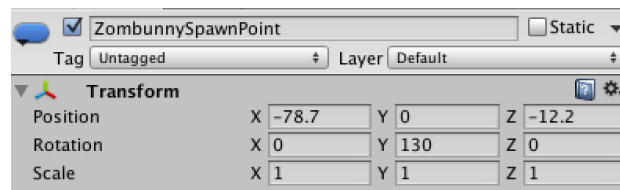
2.1 SPAWNS MECHANISM

2.1.1 Basic method

All monster's models, stats in the game stores in prefab, Unity provides a method to instantiate these prefab:

```
Instantiate(GameObject, position, rotation);
```

In the game, we set up three special game objects with Transform attribute as spawn points for our three kinds of monsters (Zombie Bear, Zombie Benny and Hell Elephant), figure above is an example spawn point. Therefore, Instantiate method creates a prefab instance in the spawn point.



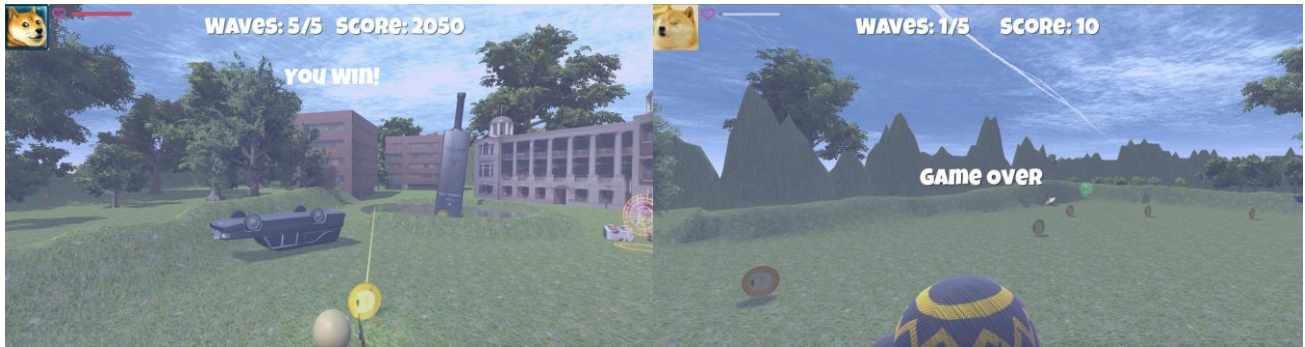
2.1.2 Wave-Based spawn

In our plan, the game consists of several levels, so we try to change the endless spawn to a wave-based spawn. The wave-based spawn has three main logics in our game:

- In the n-th wave, the enemy manager spawns $3 \times n$ monsters;
- Next wave of enemies will come after you kill all monsters in this wave;
- There are five waves in our game, if you defeat all monsters, then win text will be displayed; otherwise, you will see game over text.

To implement these three logics, we use a counter to log the enemy number, the enemy manager calls instantiate until the counter achieves the number of $3 \times n$. After that, when you kill an enemy, the counter counts down 1, if the counter goes back to 0, the wave manager will be notified to generate new wave of enemies.

In addition, if player kills all enemies in these five wave attacks, Score manager displays the *Win Text* (see figure below) on the screen and it tells players their rank in leaderboard. What matter if player dead? A Game over Text will display for them, but they will always receive their rank information.



2.2 CAMERA MECHANISM

2.2.1 Camera selection

The view of the camera is an important part of the game experience. An appropriate camera may make player have a clear perception and a convenient control to the game. There are some typical but different types of camera, like the Look-At camera, the Dungeon-Crawler camera, the Follow camera and the Mouse-Aim camera. Because the Doges Adventure is a shooting game, we consider to adopt the Mouse-Aim camera to enhance player's visual enjoyment.

2.2.2 Camera detail

The Mouse-Aim camera sits behind and above the player and rotates according to the movement of the mouse. First it would set the player as the target.

- Rotate the player

We can access the horizontal axis of the mouse and use this input to rotate the player.

- Keep behind the player

We calculate an offset according to the initial distance between the camera and the player then use this offset and the player's angles to get a new position for the camera. Make the camera look at the player after that.

For details, there is commit history:

<https://github.com/Demonsu/DogesAdventure/blob/master/Doge's%20Adventure/Assets/Camera/MouseAimCamera.cs>

2.3 PHYSICAL EFFECT

As a shooting and also action game, some fancy effect when player does something right would greatly enhance the player's enjoyment. There are already existing scripts with our enemy assets, but we think some physical feedback may provide a more real and exciting effect.

2.3.1 Shooting mechanism

There are some typical methods to make the player shoot in the game. We choose a straight way which is drawing a line along the direction that the player is facing, until it touches any parts of an enemy or reaches its end.

2.3.2 Hit-back effect

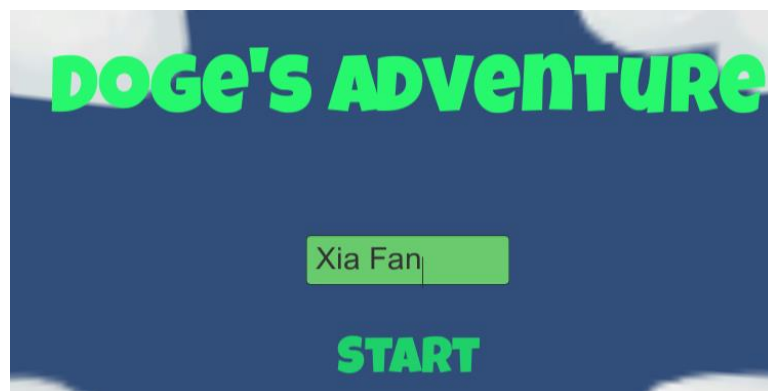
When the enemy is hit by the player, besides deduction of enemy's health value, we would add a force to the enemy along the direction from the gun to the enemy. The unity engine would do the rest to push the enemy back.

For details, there is commit history:

<https://github.com/Demonsu/DogesAdventure/blob/master/Doge's%20Adventure/Assets/Scripts/Player/PlayerShooting.cs>

2.4 SCORE RANK SERVER

When the game starts, player's name is required for recording his final score.



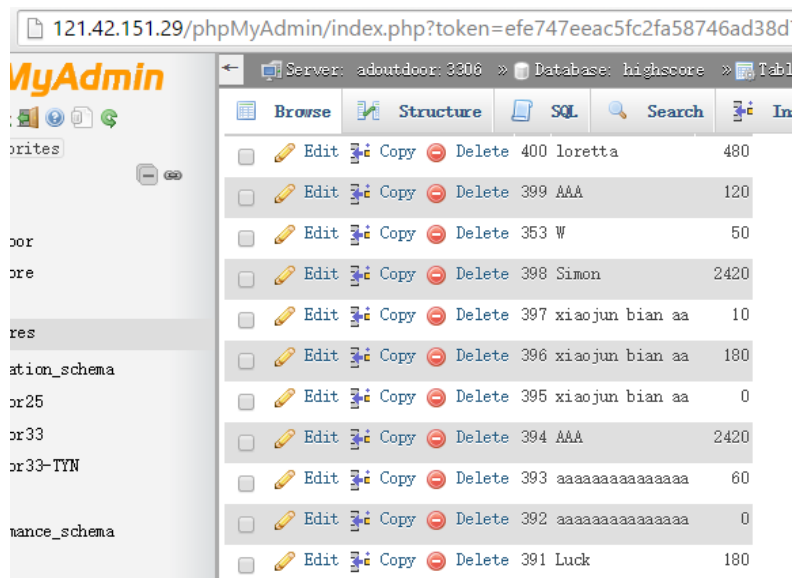
And when game ends in no matter the player wins or loses, his score together with his name will be uploaded into our server. Then the server will give him a rank in the leaderboard.



In this module, we need:

- A remote server:
 - Store names and corresponding scores in database.
 - Handle new score post.
 - Handle leaderboard request.
- Unity communicate with this server:
 - Post name and score to the server in security ways.
 - Fetch leaderboard from the server.

To implement the above game logic, we built a server based on apache, PHP and MySQL in <http://121.42.151.29>. All the names and scores are stored in database:



The screenshot shows the phpMyAdmin interface for a database named 'highscore'. A table named 'loretta' is selected, showing a list of scores. The table has two columns: 'id' and 'score'.

id	score
400	loretta
399	AAA
353	W
398	Simon
397	xiaojun bian aa
396	xiaojun bian aa
395	xiaojun bian aa
394	AAA
393	aaaaaaaaaaaaaa
392	aaaaaaaaaaaaaa
391	Luck

And we set up two interfaces:

- For posting score: <http://121.42.151.29/addscore.php>
- For fetching rank list: <http://121.42.151.29/display.php>

Then in Unity, we added a manager called HighScoreManager. All the logic corresponding to leaderboard is implemented in it. And we use the 'WWW' package to communicate with our server. For example, to post player's score:

```
public string addScoreURL = "http://121.42.151.29/addscore.php?";
IEnumerator PostScores(string name, int score) {
    //for security concern
    string hash = Md5Sum(name + score + secretKey);
    string post_url = addScoreURL
        + "name=" + WWW.EscapeURL(name)
        + "&score=" + score
        + "&hash=" + hash;
    WWW hs_post = new WWW(post_url);
    yield return hs_post;
    if (hs_post.error != null) {
        print("There was an error posting the high score: "
            + hs_post.error);
    }
    //Then we use StartCoroutine to invoke
}
```

For details:

- Server side code:

<https://github.com/Demonsu/DogesAdventure/tree/master/Server>

- Unity code:

<https://github.com/Demonsu/DogesAdventure/blob/master/Doge's%20Adventure/Assets/Scripts/Managers/HighScoreManager.cs>

3 DIFFICULTIES ENCOUNTERED

- The main character and scene of our game are highly customized, however we are not familiar with building models, especially building models with animation.
- We wrote our own camera controller script, instead of using any existing camera like FPS Controller. Inconsistence between player movement and camera direction took us a lot of time.
- We build a high score server, which means you can compete with anyone who has played this game in anyplace at any time. Server cross domain request problem and network communication in unity also took us a lot of time to solve.
- Communication between multiple processes. Although unity does a lot for us, there are still many problems, it caused many bugs in the game. Such as, sometimes, it will add two waves of enemies at the same time, sometimes, it won't add the score of the last dead enemy. It's difficult to fix such bugs.

4 OBSERVATIONS/FEEDBACKS FROM PLAYTESTING THAT USED TO IMPROVE THE GAME

After the game playtesting, we received some worthy feedbacks. We did modifications according to these feedbacks one by one.

- Can the players use the controls? Do they understand them?

Testers thought they need more instructions on the controls. We did add detailed control instructions in the start menu to help the new players

- Can the levels be completed? With how much effort?

Most testers thought the game is a bit easy to complete. We adjust some parameter of the enemies to make one kind of them move quite fast to make the players have to avoid them all the time. Besides that, coin collection challenge is added, players have another quest to handle. The holistic difficulty of the game has definitely been enhanced.

- Was anything confusing? Please take me through what you found to be confusing.

There were actually some problems during the playtesting. But we solved all of them. Coins can be collected, all the objects in the scenario could be collided, player life is indicated more conspicuously and even the movement of the player could be increased with a secret key. Now there won't be anything confused, all things are going well in this world.

5 TASK DONE BY EACH MEMBER

Our project is hosted in GitHub: <https://github.com/Demonsu/DogesAdventure>

Each member contributes a lot to the game. In general:

Chen Yuanfei(WTIFS): Built all the game models (Doge, the campus buildings, trees, terrain, coins) and wrote the corresponding scripts, such as collecting DogeCoin, rune effects(add HP/ add Attack). Added the function of holding Shift to run. Adjusted camera angle. Adjust NavMesh parameters.

Du Tianyi(obiwandu): Camera controller; Pause menu; Player controller (jump logic); Physical effect (enemy hit).

Lai Shangqi(sznyxslsq): Wave manager; Enemy manager; Score manager.

Xia Fan(Demonsu): Basic game logic (move, attack, die, generate enemies, etc.); Game menu; High score server.

And for more details, there is commit history:

<https://github.com/Demonsu/DogesAdventure/graphs/contributors>

6 STATE EXISTING RESOURCES AND ORIGINAL WORK

6.1 EXISTING RESOURCES

6.1.1 Models:

- Gun, Enemies (Zombie Bears, Zombie Rabbits and Hellephant), Dollhouse, Sword and Car: imported from the tutorial assets.
- Trees, water and grass: imported from the standard asset.
- Dogecoin, runes and skybox: online resources.

6.1.2 Sound Effect:

- Background music: <Split Sine Waves> (by Ergo Phizmiz)
- Coin Sound: from <Super Mario>
- Other sound effects: imported from the tutorial assets.

6.2 ORIGINAL WORK:

- Models in the environment: Doge; Main Building; Main Library; Hui Oi Chow Building; the terrain; light; etc.
- Game menu: Start menu; Pause menu; High score rank and restart.
- Game logic: Player Manager (shooting, moving); Camera; Score Manager; Game Manager; Wave Manager; Enemy Manager (moving, damaging).
- High score server.
- Model adaptations:
 - Dollhouse, car: change texture color
 - Sword: add project and group member name to the texture
 - Runes: adjust rune size, and add script
 - Dogecoin: add script

7 APPENDIX

7.1 ONLINE RESOURCES LINKS

Dogecoin : <http://www.turbosquid.com/3d-models/free-max-mode-doge-coin/787687>

Runes : <http://www.u3dchina.com/t-748-1-2.html>

Skybox : <https://www.assetstore.unity3d.com/en/#!/content/6332>

BGM : http://freemusicarchive.org/music/Ergo_Phizmiz/Instrumental_music_from_Billiards/05_Split-Sine-Waves