College of Civil and Environmental Engineering

12-780 Finial Project Report

Name: lin lyu

Andrew ID: linlyu

Date:12/15/2019

Contact info: linlyu@andrew.cmu.edu

Running Man

1. Introduction

1.1 Problem Description

The game market is booming, and hundreds of millions of people around the world are putting big chunks of time into the game world. Besides, nowadays people are under a lot of pressure. They urgently need a mini-game that can help them decompress without taking up too much time. So, it is necessary that we should design a fancy and attractive game for people.

The inspiration for the cool running game actually comes from the fashion extreme sports of Parkour. Since it is extreme sports, parkour is very challenging and not everyone can flexibly control their own body. The biggest feature of parkour games is that people always feel nervous and frightened, just like they are on a roller coaster.

For the overview of this project: there is a blue guy called running man and this running man would always advance on the set track, and he would be forced to jump the non-track area in the set track or the blue running man would die.

And as the distance along the track grows longer, the speed of the running man will gradually increase.

In the top left corner, there is a button and the scores the player obtained would be shown there. The scores is counted by the running distance on the setting track.

1.2 Objectives for Project

This kind of game can make people feel relaxed meanwhile feel excited. The rhythm of these Parkour games is not controlled by the players themselves. Players can only be forced to do things like dodge and jump. When the player wholeheartedly puts himself into the game, the body and mind is flowing through the game. The way these games control the rhythm gives players an exciting game experience.

This kind of game will have a score record. Players need to surpass themselves through constant challenges and greet their peaks one after another. This is the most important point for players to be fascinated. Every effort of the player will have traces, and new obstacles will arouse the blood of the players.

1.3 Overview of Approaches Taken

We would use algorithm of html5 and javascript to design this game. At first, we need to add a canvas in html and write the html framework. Secondly, we need to create some game scenarios by using ‘Phaser’ for this game. There are four main scenarios: boot, load, title, play.
Boot scene is used to prepare for some games. Load scene is about implementing the loading progress bar and loading some resources, such as images and audios, and then going to the next scene after loading. Tile scene is the menu of the game. And the play scene is the main game part.

2. Detailed features developed

2.1 Potential users and customers

The entry threshold for this type of game is low. Players only need to control a few buttons, it is very easy to get started. Users can use the fragment time to play the game to relax, not always addicted to the game, but will play it for a while from time to time.

So the customers of this game can be anyone, from students to office workers who can play this game whenever they want to release pressure.

2.2 Features of the project

Parkour games have various scenes. The scenes can be subways, express trains, and mountain climbing, but they are essentially parkour games.

And parkour games have different types of settings but the essence of all these different types are all parkour games.

From the emotion sight, it is a achievement game because users can obtain the corresponding scores through overcoimg the difficulties.

2.3 Plans for the project

I have three ideas for this project, the second one and the third one would be my future plan for this project.

The first mode: the player must avoid each non-track area. And players would get scores according to the distance they run. If the user does not fulfill this requirement, the player would fail the game. If users can finish all the jump every missing area, the game continues.

The second mode: the difference of this mode from the first mode is that users can obtain gold coins during their advancing process. They only need to obtain as many as possible during this process. And when the points obtained by collecting gold coins reach a certain value, the player will win the game.

The third mode is exactly the same when the scene is in the running set track as the second mode. The difference is that we add a second level: the player accidentally enters the virgin forest after obtaining enough points and would begin its second trip, and the user is chased by the hominid. So, the user would be forced to flee on the set track way. On the way in the forest, the rules are the same as the rules in the running track.

3. Assessment of the Projct

3.1 Algorithm Design

At first, this is achieved by adding a canvas in html5 and instantiating a game object. However, the phaser dynamically adds this canvas by the ID of the parent element of the DOM.

For example, ‘Phaser.Game(width, height, renderer, parent, state, transparent, antialias, physicsConfig)’. In this function, width is the width of the canvas, height is the height of the canvas, Phaser.CANVAS uses the html5 canvas, Phaser.WEBGL uses the better performing WebGL for rendering, Phaser.AUTO is automatically detected, if the browser supports WebGL, then it would use WebGL, otherwise using Canvas. And parent specifies that the parent element of this canvas, it can be an id or the dom element itself and phaser will automatically create a canvas in this parent element. State is the scene. Transparent means whether to use a transparent

canvas background and antialias means whether to enable antialiasing. PhysicsConfig means game physics system configuration parameters. Above all the parameters can be selected.

The following screenshot shows the initial creating state.

```
(function() {
    //var width = window.innerWidth;
    var width = 800;
    var height = window.innerHeight > 480 ? 480 : window.innerHeight;
    var gameScore = 0;
    var SantaGame = {
        init: function() {
            this.game = new Phaser.Game(width, height, Phaser.CANVAS, 'game');
            //Creating state for doing some preparation for the game
            this.game.state.add('boot', this.boot);
            //Creating state for showing the progress of loading game
            this.game.state.add('load', this.load);
            //Creating the main scene play for the game
            this.game.state.add('play', this.play);
            //Creating the menu for the game
            this.game.state.add('title', this.title);
            this.game.state.add('gameOver', this.gameOver);
            this.game.state.start('load');
        },
```

The pictures and audios I used is shown as following:

The following screenshot shows the part about boot state.

```
boot:{
    preload: function () {
        this.game.load.image('loading','assets/xiaoshixiaoguo.png');
    },
    create: function () {
        //After the finishing load, call the load scene
        this.game.state.start('load');
    }

},
```

The next step is to create the scene, which can be a custom object or a constructor. The ‘load’ scene is about creating a sprite to display the progress of loading meanwhile loading some other resources, such as images and audios. And after this part has loaded, the player would enter the next scene ‘title’.

```
load: {
    preload: function() {
        //creating a sprite to dispaly the progress of loading
        var preloadSprite = this.game.add.sprite(this.game.world.width/2,
            this.game.world.height / 2 ,'loading');
        this.game.load.setPreloadSprite(preloadSprite);
        this.game.load.audio('drivin-home', 'assets/world.wav');
        this.game.load.audio('ho-ho-ho', 'assets/bonbon.wav');
        this.game.load.audio('hop', 'assets/bomb.wav');
        this.game.load.image('platform', 'assets/1.png');
        this.game.load.spritesheet('santa-running', 'assets/runman.png',
        493/5, 174,5);
        this.game.load.image('snow-bg', 'assets/beijing1.png');
        this.game.load.image('snow-bg-2', 'assets/yuanjing1.jpg');
        this.game.load.image('snowflake', 'assets/xiaoshixiaoguo.png');
        this.game.load.image('logo', 'assets/name.png');
        this.game.load.image('startbtn', 'assets/bangzhujiantou.png');
    },
    create: function() {
        this.game.state.start('title');
    }
},
```

Thirdly, we would create the title scene. ‘Autoscroll’ means constantly moving, which makes the background and ground move at different speeds and then we need to create the tween animation for the title. ‘TitleGroup’ is a group concept that puts a small group together for operation.

For example, to(properties, duration, ease, autoStart, delay, repeat, yoyo). Properties contains a js property, which is what the object should become.

Duration means interval time with Milliseconds as units. Ease means whether need to beease and the default is uniform speed animation. Autostart means whether the game needs to start automatically. Delay means the delay time with milliseconds as units at which the animation starts. Repeat means the number of times the animation is repeated. If the animation needs to loop forever, set this value to Number.MAX_VALUE. Yoyo means if the value is true, the animation will automatically reverse. Finally added a btn plus click time and set anchor point.

```
title: {
    create: function() {
        //Make the background and ground move at different speeds
        this.bg_heaven = this.game.add.tileSprite(0, 0, width, height,
            'snow-bg-2').autoScroll(-50,0);
        this.bg = this.game.add.tileSprite(0, 0, width, height,
            'snow-bg').autoScroll(-100,0);
        this.logo = this.game.add.sprite(this.game.world.width/2-158,
            20, 'logo');
        this.logo.alpha = 0;
        //Crete title tween animation
        this.game.add.tween(this.logo).to({
            alpha: 1
        }, 1000, Phaser.Easing.Linear.None, true, 0);
        this.startBtn = this.game.add.button(this.game.world.width/2-89,
            this.game.world.height - 120, 'startbtn', this.startClicked);
        this.startBtn.alpha = 0;
        this.game.add.tween(this.startBtn).to({
            alpha: 1
        }, 1000, Phaser.Easing.Linear.None, true, 1000);
    },
    startClicked: function() {
        this.game.state.start('play');
    }
},
```

Finally, the most important part in the game is the ‘play’ scene. It mainly contains two important parts. One is the create function. In the create function, players can jump the missing area by using the space bar in the

keyboard. The other is update function. In this update function, it would calculate the scores the player obtained. And the last function is about ‘game over’. And the code is in the appendix.

3.2 Interface Display

You can see as following is the initial interface of my project. After the load is done, the button would show in the interface. And players can just click on the button to start the game.

Figure 1: The initial interface

This figure following shows the main interface of the whole game. The blue guy called running man. He would continually run from the game start. To obtain the scores and avoid to die, players should use the space bar on the keyboard to help the running man jump the missing area. And as the distance the man has run increase, the speed of the running man would increase too. So it is clear that the difficulty of the game is increasing.

Figure 2: The play interface

The following figure displays game over interface. When the running man can not jump the missing area, he would die and the game is over, the scores obtained would be cleared.

Figure 3: The game over interface

4. Appendix: Any details about the GUI, codes and other information about the product.

```
play: {
    create: function() {
        gameScore = 0;
        this.currentFrame = 0;
        this.particleInterval = 2 * 60;
        this.gameSpeed = 580;
        this.isGameOver = false;
        this.game.physics.startSystem(Phaser.Physics.ARCADE);
        this.music = this.game.add.audio('drivin-home');
        this.music.loop = true;
        this.music.play();
        this.bg_heaven = this.game.add.tileSprite(0, 0, width, height,
            'snow-bg-2').autoScroll(-50,0);
        this.bg = this.game.add.tileSprite(0, 0, width, height, 'snow-bg');
        this.bg.fixedToCamera = true;
        this.bg.autoScroll(-this.gameSpeed / 6, 0);
        this.emitter = this.game.add.emitter(this.game.world.centerX,-32,50);
        this.platforms = this.game.add.group();
        this.platforms.enableBody = true;
        this.platforms.createMultiple(5, 'platform', 0, false);
        this.platforms.setAll('anchor.x', 0.5);
        this.platforms.setAll('anchor.y', 0.5);
```

```
        for (var i = 0; i < 5; i++) {
            plat = this.platforms.getFirstExists(false);
            plat.reset(i * 192, this.game.world.height - 44);
            plat.width = 300*0.6;
            plat.height = 88*0.6;
            this.game.physics.arcade.enable(plat);
            plat.body.immovable = true;
            plat.body.bounce.set(0);
        }
```

```
        this.lastPlatform = plat;
        this.santa = this.game.add.sprite(100, this.game.world.height-200,
            'santa-running');
        this.santa.animations.add('run');
        this.santa.animations.play('run', 15, true);
        this.santa.width = (493/5)*0.5;
        this.santa.height = 174*0.5;
        this.game.physics.arcade.enable(this.santa);
        this.santa.body.gravity.y = 1500;
        this.santa.body.collideWorldBounds = true;
        this.emitter.makeParticles('snowflake');
        this.emitter.maxParticleScale = 0.02;
        this.emitter.minParticleScale = 0.001;
        this.emitter.setYSpeed(100, 200);
        this.emitter.gravity = 0;
        this.emitter.width = this.game.world.width * 1.5;
        this.emitter.minRotation = 0;
        this.emitter.maxRotation = 40;
        this.game.camera.follow(this.santa);
        this.cursors = this.game.input.keyboard.createCursorKeys();
        this.spacebar = this.game.input.keyboard.addKey(Phaser.Keyboard.SPACEBAR);
        this.emitter.start(false, 0, 0);
        this.score = this.game.add.text(20, 20, '', {
            font: '24px Arial',
            fill: 'white'
        });
    },
```

```
    update: function() {
        var that = this;
        if (!this.isGameOver) {
            gameScore += 0.5;
            this.gameSpeed += 0.03;
            this.score.text = 'Score: ' + Math.floor(gameScore);
            this.currentFrame++;
            var moveAmount = this.gameSpeed / 100;
            this.game.physics.arcade.collide(this.santa, this.platforms);
            if (this.santa.body.bottom >= this.game.world.bounds.bottom) {
                this.isGameOver = true;
                this.endGame();
            }
            if (this.cursors.up.isDown && this.santa.body.touching.down ||
                this.spacebar.isDown && this.santa.body.touching.down ||
                this.game.input.mousePointer.isDown && this.santa.body.touching.down ||
                this.game.input.pointer1.isDown && this.santa.body.touching.down) {
                this.jumpSound = this.game.add.audio('hop');
                this.jumpSound.play();
                this.santa.body.velocity.y = -500;
            }
```

```
            if (this.particleInterval === this.currentFrame) {
                this.emitter.makeParticles('snowflake');
                this.currentFrame = 0;
            }
            this.platforms.children.forEach(function(platform) {
                platform.body.position.x -= moveAmount;
                if (platform.body.right <= 0) {
                    platform.kill();
                    var plat = that.platforms.getFirstExists(false);
                    plat.reset(that.lastPlatform.body.right + 192,
                        that.game.world.height - Math.floor(Math.random() * 50) - 24);
                    plat.body.immovable = true;
                    that.lastPlatform = plat;
                }
            });
        }
    },
```

```
        endGame: function() {
            this.music.stop();
            this.music = this.game.add.audio('ho-ho-ho');
            this.music.play();
            this.game.state.start('gameOver');
        }
    },
```

```
    gameOver: {
        create: function() {
            this.bg_heaven = this.game.add.tileSprite(0, 0, width, height,
                'snow-bg-2').autoScroll(-50,0);
            this.bg = this.game.add.tileSprite(0, 0, width, height, 'snow-bg');
            this.bg.autoScroll(-this.gameSpeed / 6, 0);
            this.score = this.game.add.text(this.game.world.width / 2 - 100,
                200, 'Score: ' + Math.floor(gameScore), {
                font: '42px Arial',
                fill: 'white'
            });
            this.score.alpha = 0;
            this.game.add.tween(this.score).to({
                alpha: 1
            }, 600, Phaser.Easing.Linear.None, true, 600);
            this.restartBtn = this.game.add.button(this.game.world.width / 2
                - 103.5, 280, 'startbtn', this.restartClicked);
            this.restartBtn.alpha = 0;
            this.game.add.tween(this.restartBtn).to({
                alpha: 1
            }, 600, Phaser.Easing.Linear.None, true, 1000);
        },
        restartClicked: function() {
            this.game.state.start('play');
        }
    }
};
SantaGame.init();
}());
```