Human Computer Interaction Course Project

Forest Runner—A leap motion based interactive web game

1552681 陈淇格
1552616 张晓琰

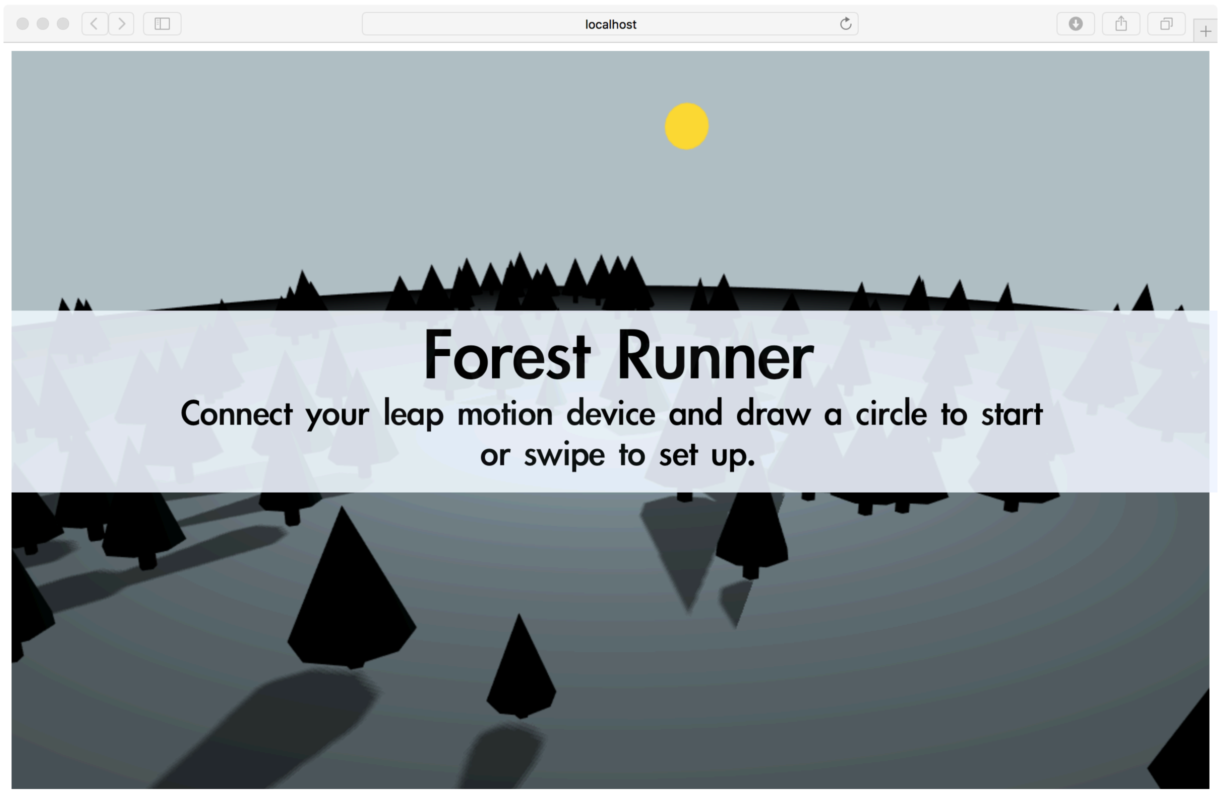
Overview

‣ Summarization

Forest Runner is a 3D leap motion based interactive webGL game. It is built with three.js. Player need to dodge the tree and collect coins to win scores, by moving hands over leap motion device to control the cube in game scene.

‣ Player's Instruction

1.First open "startScene.html".

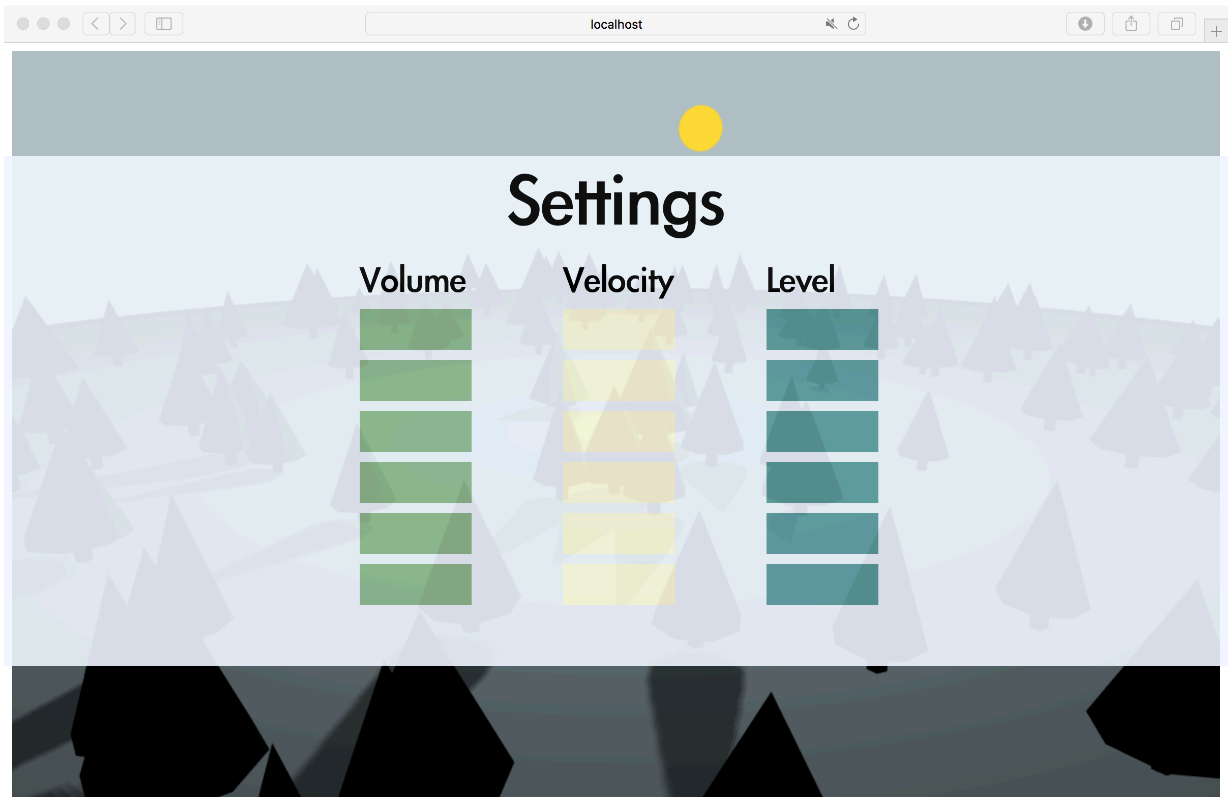

localhost
Forest Runner
Connect your leap motion device and draw a circle to start or swipe to set up.

2.To start game, draw a circle with your hand. If the website does not jump to the game scene, then check that if leap motion device is properly connected. If this does not work, open the leap motion visualizer and make sure that your hands are in the visualizer window.

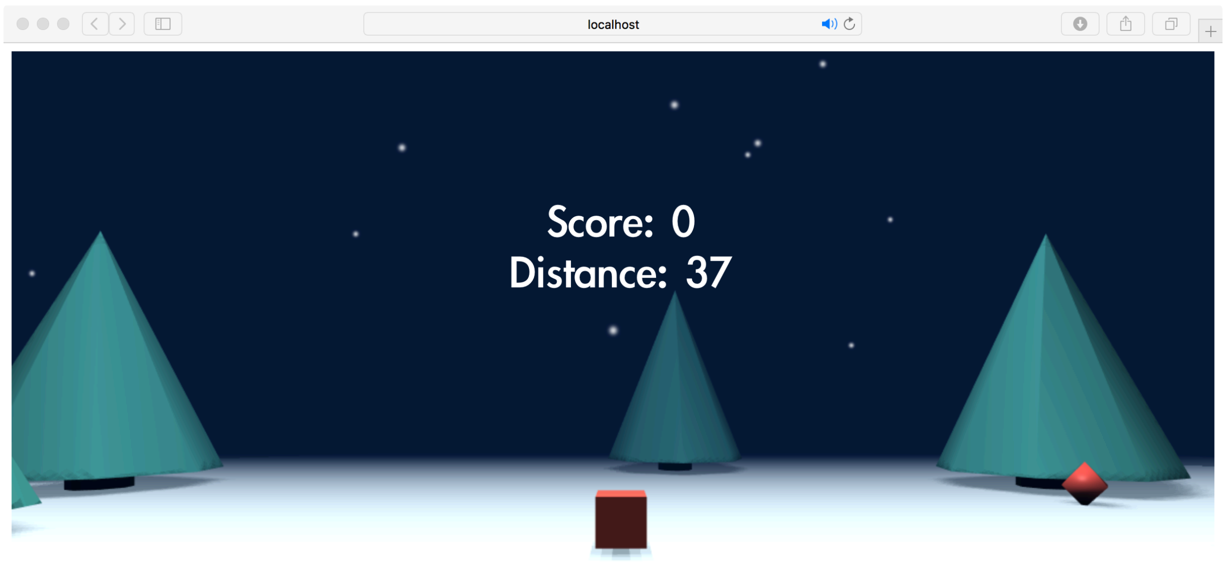
If you want to set up game volume or game level, make a swipe gesture.

To set game volume, put your hand at left of leap motion's detection area.When the "volume" turns to green, it means that the volume panel is selected, and you can put your hand higher to increase volume or put your hand lower to decrease volume. The number of blocks which turns to white implies current volume intensity.

To set game level, put your hand at right of leap motion's detection area. When "level" turns to blue, it means that the level panel is selected. You can put your hand higher to get higher level, or put your hand lower to get lower level. The number of blocks which turns to white implies current level. The higher, the more difficult the game is.



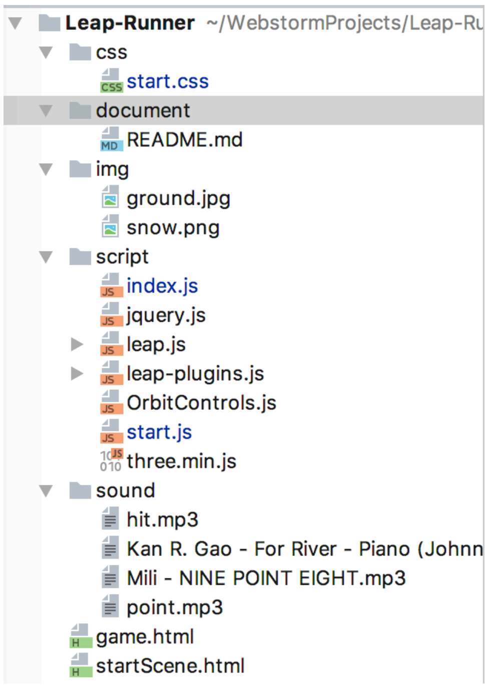
3.When entering game, move your hands over leap motion to control the cube to move left or right. Dodge the trees and collect the coins.When bumping into a tree, the score will decrease, but if you quickly get far away from the tree, your score decreased will be less. When accounting a coin, your score will increase.When the score decreases to 0, you'll die. You can choose to make a swipe gesture to restart the game.



*ATTENTION:The detection area of leap motion is limited, so you can't keep your hands too far from the device.If you cannot play the game properly, you could check that if your hands are displayed in the visualizer of leap motion, or that if the leap motion device is properly connected.

Program Structure and Modules

‣ Program structure



```
▼ Leap-Runner ~/WebstormProjects/Leap-Ru
  ▼ css
      start.css
  ▼ document
      README.md
  ▼ img
      ground.jpg
      snow.png
  ▼ script
      index.js
      jquery.js
    ▶ leap.js
    ▶ leap-plugins.js
      OrbitControls.js
      start.js
      three.min.js
  ▼ sound
      hit.mp3
      Kan R. Gao - For River - Piano (Johnn
      Mili - NINE POINT EIGHT.mp3
      point.mp3
    game.html
    startScene.html
```

1.Css folder: store style sheets
2.Document folder: store README file
3.Img folder: store pictures used in game
4.Script folder: store javascript files
index.js: Script for initializing game scene and game objects, adding game logical judgement. Leap motion controller of game player is also included.
start.js:Script for initializing the starting game scene. Includes leap motion controller of settings panel.
5.Sound folder: store game background and effect audios

‣ Program modules

1.Leap motion module
The connection of leap motion device is implemented in start.js (connection of leap motion and the start scene) and index.js(connection of leap motion and the game scene), so that on entering both of the start and game website , the device can be connected.
In this module gestures can be recognized, and position of object in game scene can be controlled by moving hands over leap motion.

2.Game scene and object initialization module
In this module, game scene, camera, light, ground, player, coins, obstacles is created by using objects of three.js.
Create game score, moved distance and move speed variable.
Use webGL renderer to render game scene and use function “update” to move game player and other game objects according to time and moving speed.
The camera moves with the player and is always behind the player ,making sure that the player is always within view. Create trees and coins randomly in front of the game player.

3.Game logic module
Increase or decrease game score when player accounts with a coin or bumps into a tree. When score decreases to 0, game ends.

4.Game audio module
Add game background music and effect music.

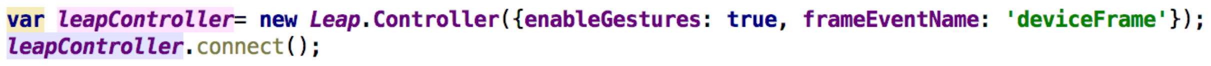
5.Game settings module
Use div element of css to create blocks representing volume, velocity and level .Combine them to leap motion for controlling.

Core Algorithm

▸ Leap motion integration in javascript

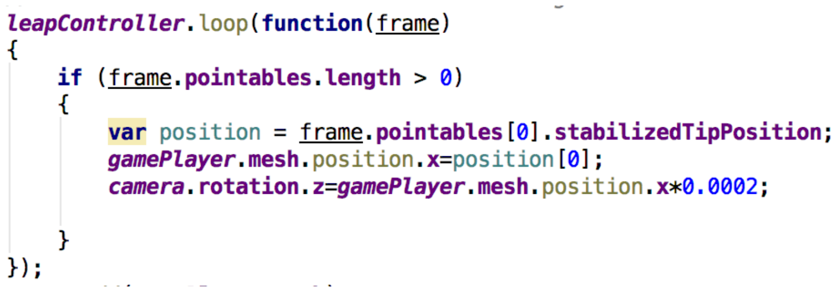
1.We choose javascript for writing this project. At first we have thought of unity, but after trying many times, we found that leap motion SDK is not very compatible with unity. One of us use mac, and this makes development even more difficult, because leap motion's orion beta SDK is not compatible with OSX. So we choose javascript and use three.js to build a 3D webGL game. This makes loading and rendering the game faster .

2.Leap motion connection
First, introduce the leap.js file into html.
Then connect to leap motion controller.The LeapJS library connects to the LeapMotion service through a WebSocket.



```
var leapController= new Leap.Controller({enableGestures: true, frameEventName: 'deviceFrame'});
leapController.connect();
```

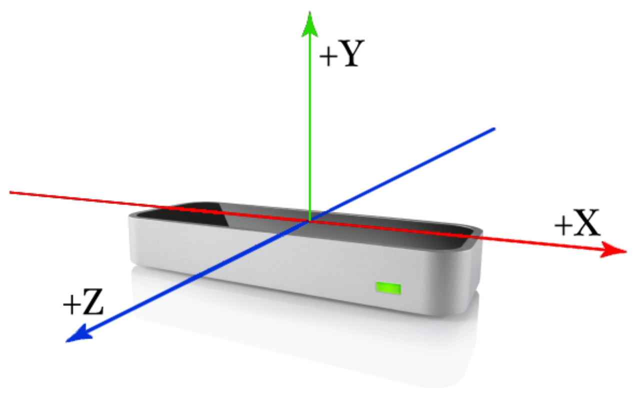
3.Gesture & position recognition
In script for the main game scene, we use the pointables class of leap.js. This class reports the physical characteristics of a detected finger. In our program we get the first detected finger tip and use stabilizedTipPosition to get a (x,y,z) scale."stabilizedTipPosition" is based on the velocity of the pointable to make precise positioning easier.Then get the x scale of the position and link it to the x scale of game player, so that the player could move right and left according to the x position of hand. The camera rotation sentence is written because when the player move, the camera will rotate with it, looking as if the ground leans when the player moves. This makes the scene looks more real.



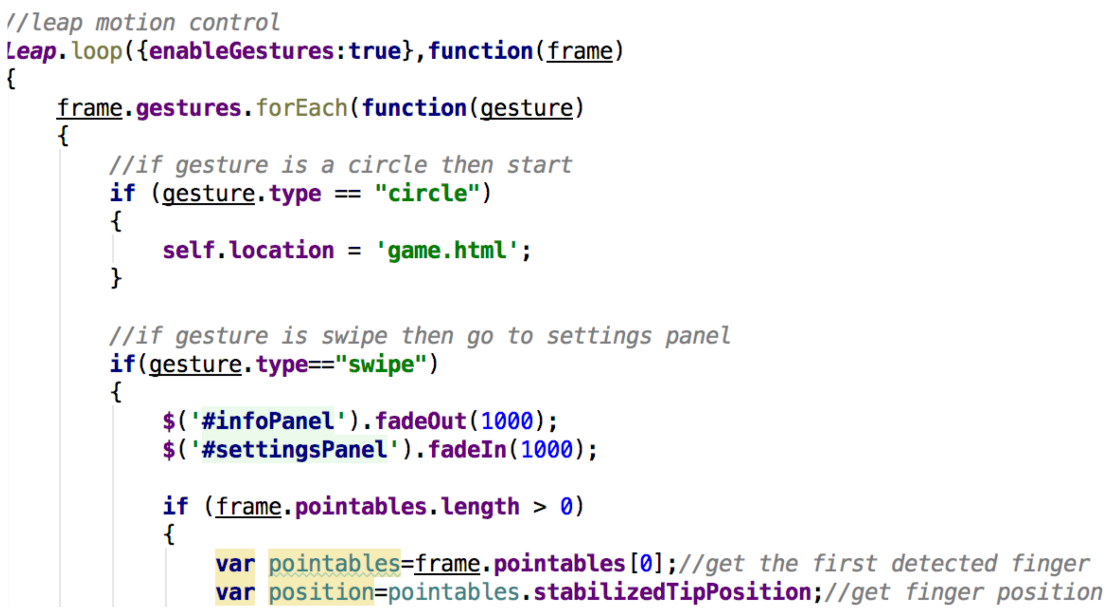
```
leapController.loop(function(frame)
{
    if (frame.pointables.length > 0)
    {
        var position = frame.pointables[0].stabilizedTipPosition;
        gamePlayer.mesh.position.x=position[0];
        camera.rotation.z=gamePlayer.mesh.position.x*0.0002;

    }
});
```

Leap motion's coordinate system is similar to three.js's coordinate system.



In script for the starting game scene, use the following code to judge gesture type. If gesture type is a circle , then jump to game scene. If gesture type is swipe then jump to settings scene.
Then get the stabilizedTipPosition like we do in the game scene for volume, velocity and level selection.
To make sure that the interaction of human and computer using leap motion accords with user's interaction habit, firstly we get the x scale of tip position to simulate the horizontal ordinate on screen. Different range of x scale simulate different settings choices. For example, when x scale is less than 0 then the volume setting panel on the left of the screen is selected. Secondly we get the y scale of tip position to simulate the vertical ordinate on screen. The bigger the x scale is, the more blocks representing volume/velocity/level amount is selected, and the variables for volumes, velocity and level will be adjusted accordingly. For example, as user's hand is put higher, the higher the volume is.



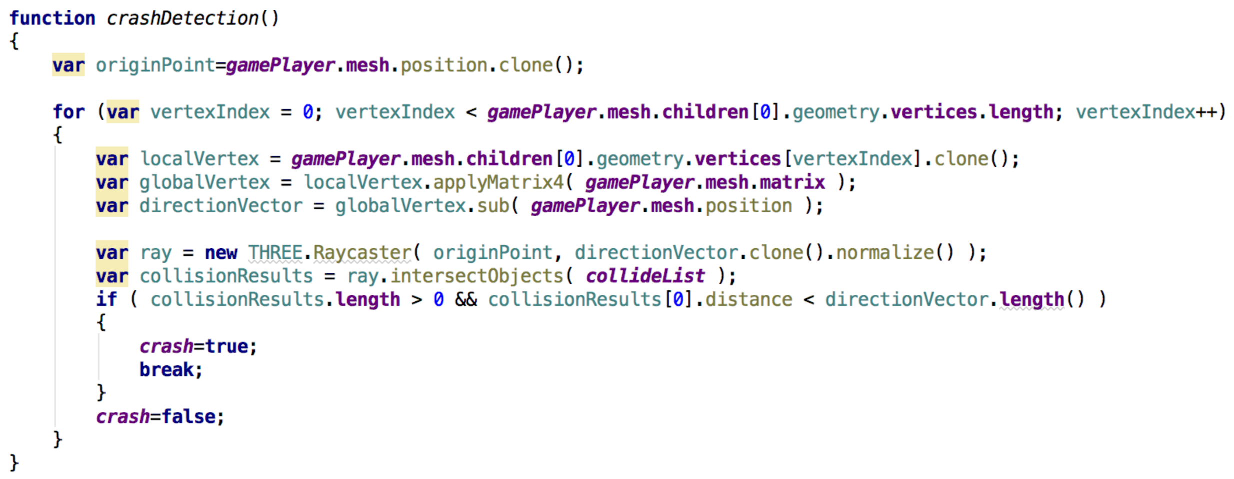
```
//leap motion control
Leap.loop({enableGestures:true},function(frame)
{
    frame.gestures.forEach(function(gesture)
    {
        //if gesture is a circle then start
        if (gesture.type == "circle")
        {
            self.location = 'game.html';
        }

        //if gesture is swipe then go to settings panel
        if(gesture.type=="swipe")
        {
            $('#infoPanel').fadeOut(1000);
            $('#settingsPanel').fadeIn(1000);

            if (frame.pointables.length > 0)
            {
                var pointables=frame.pointables[0];//get the first detected finger
                var position=pointables.stabilizedTipPosition;//get finger position
```

‣ Collision detection

Use raycaster class of three.js for collision detection.
The principle of using raycaster for collision detection of object A and object B is : begin from the center of A ,send ray from center to each vertex of A and check if the ray intersects with B. If intersection happens, then check the distance between the nearest intersection point and the center of A. If the distance is smaller than the distance between center of A and vertex of A, then collision happens.
We add all of tree objects to array collideList and all of coin objects to array coinList for easier detection.



```
function crashDetection()
{
    var originPoint=gamePlayer.mesh.position.clone();

    for (var vertexIndex = 0; vertexIndex < gamePlayer.mesh.children[0].geometry.vertices.length; vertexIndex++)
    {
        var localVertex = gamePlayer.mesh.children[0].geometry.vertices[vertexIndex].clone();
        var globalVertex = localVertex.applyMatrix4( gamePlayer.mesh.matrix );
        var directionVector = globalVertex.sub( gamePlayer.mesh.position );

        var ray = new THREE.Raycaster( originPoint, directionVector.clone().normalize() );
        var collisionResults = ray.intersectObjects( collideList );
        if ( collisionResults.length > 0 && collisionResults[0].distance < directionVector.length() )
        {
            crash=true;
            break;
        }
        crash=false;
    }
}
```

Implemented Requirements

- Development environment

IDE: Webstorm
Browser: firefox, safari
Leap motion device
Leap motion SDK (version2.3.1)
leap.js (version 0.6.4)
jquery.js (version 1.10.2)
three.js

- User environment

To run the game on your computer, you need to:

1. Connect the leap motion device.
2. Download leap motion driver from https://www.leapmotion.com/setup/desktop/osx
3. Setup leap motion. Open visualizer. Make sure your hands are within the visualizer window.
4. Open the html file with a browser which support webGL, like safari, firefox, chrome, etc.

Advantages & Disadvantages

▸ Advantages

1.Faster loading speed

We use three.js to build a 3D webGL game and integrate it with leap motion controller. In some leap motion integrated unity games, we found that the loading speed of the game is too slow, and sometimes the gestures and hand positions will not be quickly recognized, leading to the failure of interaction experience. Using javascript makes loading and rendering the game faster, and the environment-building step is easier than that in unity.

2.Free of keyboard and mouse control

In the game user could use leap motion to start a game, play the game with moving hands above leap motion restart a game when game player dies, or go to the settings panel and move hands to set the volume/velocity/level. This create a mouse-less and keyboard-less interaction experience for users. This experience has much to offer in its novelty and freshness in human-computer interaction method.

3.Concise user interface

Our game has a user-friendly user interface and a low-poly game scene.Although all of our game models are just combination of geometries, the game does not look dull. Instead, the low-poly scene provides with users a pleasant view.

4.Potential appliance in clinical medicine area

Because users need to move hands position or make gestures for game control, it may training the moving ability of hands' muscles. We've thought that this game could be used for rehabilitation training of patients whose hands are injured.

▸ Disadvantages

1. Smoothness of movement

We simulate the player's movement on axis x by moving the x position of the cube according to hand position, and simulate the player's movement on axis z by increasing the z position of the cube as the time passes, but this movement looks a little rigid.

2.Functions uncompleted
Function of using hand position for setting volume, velocity and level is uncompleted. We only complete the volume controlling system as a demo of how leap motion can be used to control user interface. Because implementing the velocity system and difficulty system would need a much more complicated logic and more coding work, what's more, time is limited, so we only complete the volume controlling system.

Potential improvement

1.Import game models
We constructed all of the game models by writing code to put together varies of geometries. For further improvement, we could download and import some game models to make the scene more attractive.

2.Use physics engine
Due to the rigidness of player movement and collision, we would consider using javascript physics engine such as physijs, for movement controll and collision detection. This would make the interaction experience more satisfying.

3.Complete the difficulty and velocity system
Set game levels according to the user's selection on the settings panel.The higher the level is, the more distance the user need to reach for wining the game. The higher the velocity is , the moving speed of the cube is faster, and dodging the trees and collecting the coins will become more difficult. User could choose the level and velocity according to their own preference.

Reference

- Leap motion javascript API:

https://developer.leapmotion.com/documentation/javascript/api

- Three.js:

souce: https://github.com/mrdoob/three.js

API: https://threejs.org

- jquery.js:

source:https://jquery.com