Report for the Practical 1 – Phidgets (Physical Computing)

Haoda Miao
The UNIVERSITY OF St. Andrews
St. Andrews, UK
hm212@st-andrews.ac.uk

INTRODUCTION

This is a game that player operate the car to avoid colliding with police on the road. The player can make the car run left, right and go ahead. There are three roads in the window. The car can avoid knocking on the police cars through running to the other roads that do not have police on it. The car that operates by the player will always show in the window, but the police cars will disappear when they run to the nether border of the window. If the car collides with the other cars, the game will be over. Moreover, the car has 1000 value for the fuel. The fuel will decrease along the distance that the car runs over. The game will be over as well if the fuel of the car runs out. In order to keep the car running, there will be some fuel bins appear on the roads. The player should try to collect the fuel bins on the road to supplement the fuel of the car. In this way, the car will run further. Besides, there are different degrees for the speed of cars, from one to nine. The car will be slow in level one, if the player wants more challenges, they can change different level. During the game, the player can see the distance that he had run and the speed level will also show in the window. At the end of the game, the player will see the distance that the car runs over, namely the score of the player.

IMPLEMENTATION

The hardware used in this project:

1. Rotation Sensor
2. Joystick Controller
3. Phidget

There four class files in this project folder, which are the car class, the fuel class, the police cars class and the main class. The params of the car define in the car class, includes the coordinate of the car, the fuel value and the width and height of the car. The car class also involves some method that can get or set params and a show method that draw the car object in the window. The fuel class and the police class are similar to the car class. Both have the coordinate and the width and length in the class. The fuel class has a fuel param, which is the value of the fuel supplement. In the main class, the first thing is to initialise the object car, police and fuel. Then initialise the object VoltageRat to get the input from the hardware. There are two input value should receive during the game, the move position of the car and the speed level. Following, the setup method creates a window and set the hardware channel, device serial number and open the object VoltageRat. After the setup is the draw method. In the draw method, it draws the game background initially, like the roads and the frame that shows the distance and the speed. Then, reset the y coordinate based on the value received from the hardware. There will be an upper limit for the y coordinate of the car, because the player cannot move the car run down. Therefore, the judge condition for whether the y coordinate equals the hardware value is if the value from the hardware over the upper limit. In the case of the hardware value is bigger than the y upper coordinate of the car, the y coordinate will equal to the upper limit, otherwise, y will equal the hardware value. As for the x coordinate, there is no limit because the car can run left and right. To make the game run normally when the player does not turn the speed controller. There will be a judgment for the speed value if the value is less than or equals to one, then set the speed value to one, or else set the speed value to the value obtained from the speed controller. After getting these values from the hardware, the objects' params can reset by calling relevant methods. Then add all objects into the window. Due to the game will be over when the car collides with the police whatever from which direction (front or sides), it needs some judgment conditions. For the situation that the car knocks on the police from the front. This case will happen when the x coordinate of the car is in the range from the x coordinate of the police to the police's x coordinate plus the width of the police, at the same time, the y coordinate of the car is in the range from the police y coordinate to the police's y coordinate plus the height of the police. It only changes the order of the judgment for the situation that the car collides the police from sides. After the car collides with police, the window will show the distance and other information, then execute the reset method which will reset the value of the car and the distance value. The y coordinate of the object police will reset by a random number in the reset method as well. The way to judge if the car gets the fuel bin is the same as the judgment method for police. The difference is the loop will not stop when the car gets the fuel bin, it will disappear immediately, and the fuel value of the car will increase. Regardless of the situations mentioned above, there is also some situation that the car touches neither police nor fuel bin. These objects will run to the upper limit value of y coordinate. The y coordinate will be reset to a random value when these objects run down to the lower bound of the window. At the end of the draw method, it will judge the fuel value of the car. If the fuel value is less than or equals to zero, then stop the loop, or else decrease the fuel value and increase the distance.

Evaluation

At the beginning I want to create a game like Mario, but I found that it is difficult to make it descent after it jump up. Therefore, I made this car game. During the developing processing, I found that I was not very clear to the loop things. Sometimes, I cannot figure out why the value was missing. The most confusing thing was that I did not know how to restart the game. I want to loop the programme without pressing mouse to make the game start again, but I did not find the method for that. Furthermore, I hope my codes will be clearer and more logical, because in this project, some codes in the main class should be written in the specific class. And the appearance of the police and the fuel also should be random.