CPT205

Computer Graphics

Xiaofei Chen

2252798

Information and Computer Science

Assessment 2 – 3D Modelling Project

Rainbow

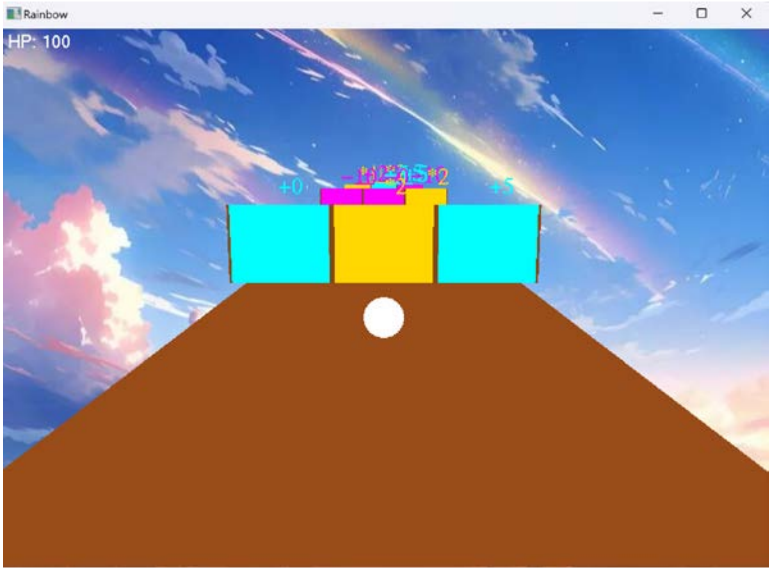
## 1 Scenario introduction

This project was inspired by the little adverts that pop up when you swipe a short video. In the advert, the player controls a little man who can shoot automatically, and by moving left and right, he chooses to shoot at different targets to strengthen himself, and finally defeats the boss to win the game. The spokesperson usually makes a mistake on purpose to get a comment, piquing our interest in clicking on the advert to give it a try. Often, however, we click on the advert and find that it's not the same game, and we feel cheated. I wanted to find out more about this and finally found the prototype for the mini-game in this advert. It's the hot foreign game, **Count master: Crowd Runners 3D**. This game has a very interesting style of drawing and is very educational, only the gameplay is slightly single, only simple addition, subtraction, multiplication and division to change the number of miniatures. So for this project, I decided to try to replicate some of the content from the advert and innovate based on the original game.

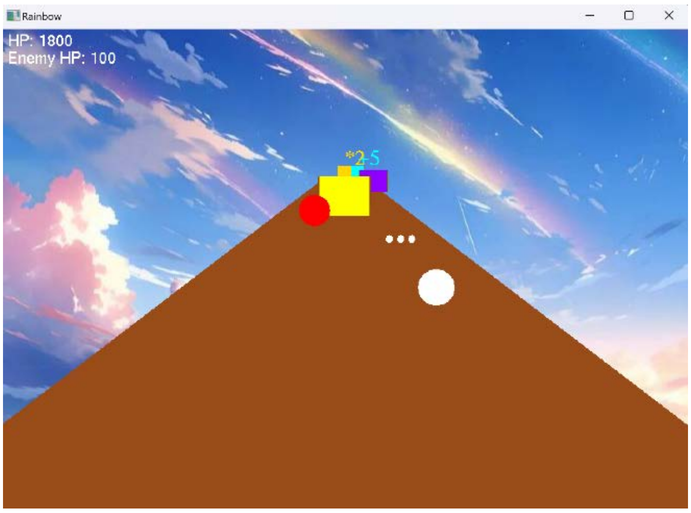
My 3D game is called **Rainbow(七彩球)**. The player will control a small white ball that can change its HP through four arithmetic gates, corresponding to **addition, subtraction, multiplication and division,** and gain different gains through seven special effect holes, corresponding to seven different attributes. **Red for power, orange for speed, yellow for defence, green for recovery, blue for control, indigo for stealth and purple for special**. Meanwhile, the ball will face seven enemies in turn, each with abilities corresponding to the colour attributes. The ball can attack the enemy by firing bullets, or you can dodge enemy bullets by jumping. When all seven enemies are eliminated, the player wins the game. At any moment, when the player's HP is 0, the player loses.

## 2 Content description

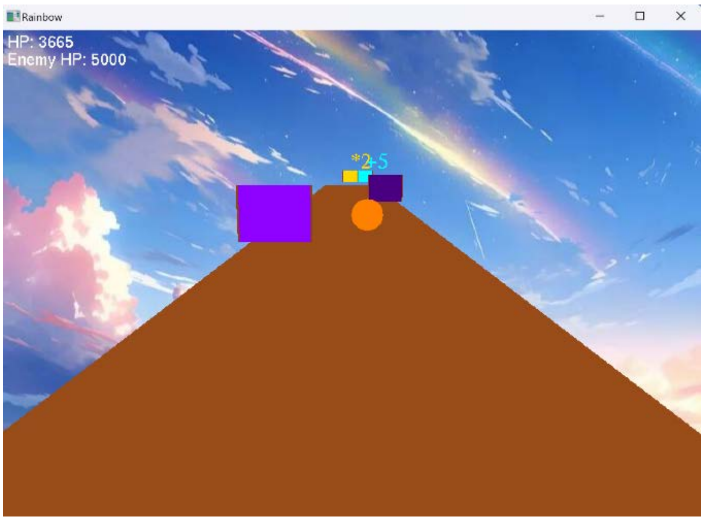
The project begins by defining a series of global variables that will be used later. After that, it defines structures such as doors, holes, bullets, and enemies. Next, we'll start by rendering the background image. First, we **prepare a bmp image as a texture map** and process the texture using the method taught by our professor in the lab. Note that bmp images cannot be compressed, and the number of bytes in each line of pixel data needs to be a multiple of 4. And we read the pixel data from the 54th byte of the file, because in a bmp image, before the 54 bytes are the file header and information header. Also, since the background image is at the bottom, we need to disable the depth test to make sure that the background image is not obscured by other objects and recovered after rendering the background. Then we **pick a suitable viewpoint and fix it**. In this case, we'll just choose an overhead view. Next we render the brown track and the white ball in the middle of the view, and make sure that the ball is in the middle of the track at the beginning, and that it can't move out of the track. We then need to render four different doors corresponding to addition, subtraction, multiplication and division. If the player passes through the door, it will change the HP according to its value. Passing through the door there is a green **flashing effect**, i.e. there is a timer that dynamically changes the shade of the colour depending on the time, which is similar to a sine function. It is worth mentioning that in order to restore the original and take into account the balance of the game, a **weighted random method** is used here to simulate the generation of values. The values of these gates appear in a certain ratio, which enhances the gameplay to a certain extent. After that we initialise some parameters and render the HUD in the top left corner, completing a prototype of this game. It's going to look like this:



Next we need to use the **keyboard and mouse interaction** functions. Clicking the left mouse button allows the player to shoot white bullets, the only means of defeating the enemy. As for the keyboard interaction, we use A and D to control the movement of the ball to the left or right, space to control the jumping of the ball and R to reset the game. Reset means to restart the game when it fails. Every time the ball passes through some doors, seven enemies will appear in turn. At this point the normal arithmetic doors will disappear and in their place **seven holes with different attributes** will appear, which the player can pass through to gain a buff that lasts for a while and then disappears. The generation of holes is completely random, with equal probability for the seven colours. At the same moment, the player can only have a maximum of one gain, and gains cannot be stacked. These holes will temporarily disappear after each enemy is defeated and will reappear when the next enemy arrives. Different buff will change the player in different ways, specifically: red will give the player a power-up by changing bullet damage from 100 to 200; orange will double the player's horizontal x-axis movement speed to better dodge enemy bullets; yellow will halve the player's damage from enemies, i.e., 100 becomes 50; green will allow the player to continually regain HP, i.e., self-increase in HP every frame; blue will halve the enemy's movement speed; blue will increase the player's HP by half; and blue will increase the player's HP by half. Blue halves the enemy's horizontal x-axis movement speed for a control effect. Special, indigo will make the player go into stealth for a short period of time, at which time the enemy will not fire bullets; purple will make the player's bullets have a scattering effect, i.e., a deflected trajectory. These two special effects would look something like this:



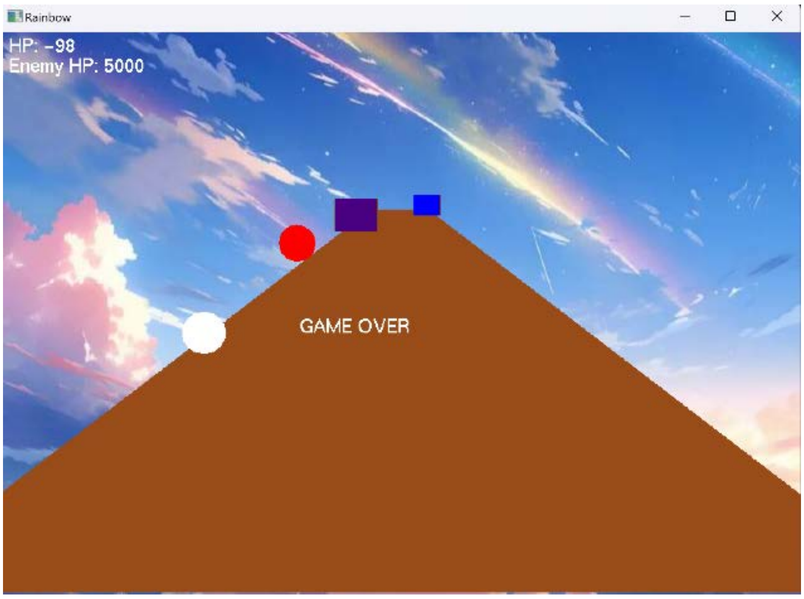
HP: 1800
Enemy HP: 100

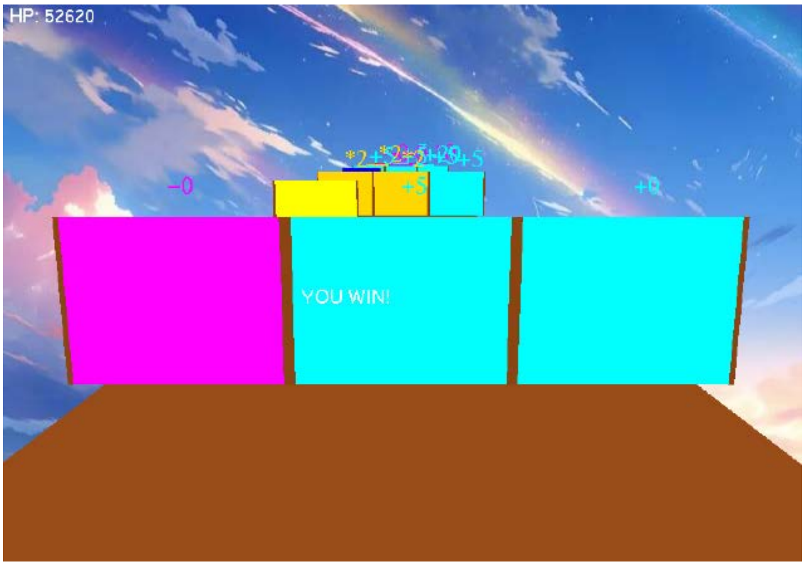


HP: 3665
Enemy HP: 5000

Additionally, enemies of different colours also have gain effects corresponding to their colour and their left and right movements are randomised, which will make the game more challenging. Of course, since there are bullets that cause damage, the **collision detection** function is also important. This simply means that if the z-coordinate of the enemy bullet is the same as the player, the lateral x-coordinate of the two should be compared, and if it happens to be within a certain range, i.e. it causes a reduction in HP. Of course, if the player has a jump operation, that is, the y coordinate is greater than a certain value, even if the x coordinate in the range of the collision should be, will not happen HP reduction, so that the effect of dodging bullets. The opposite is also true for the collision between the player's bullet and the enemy.

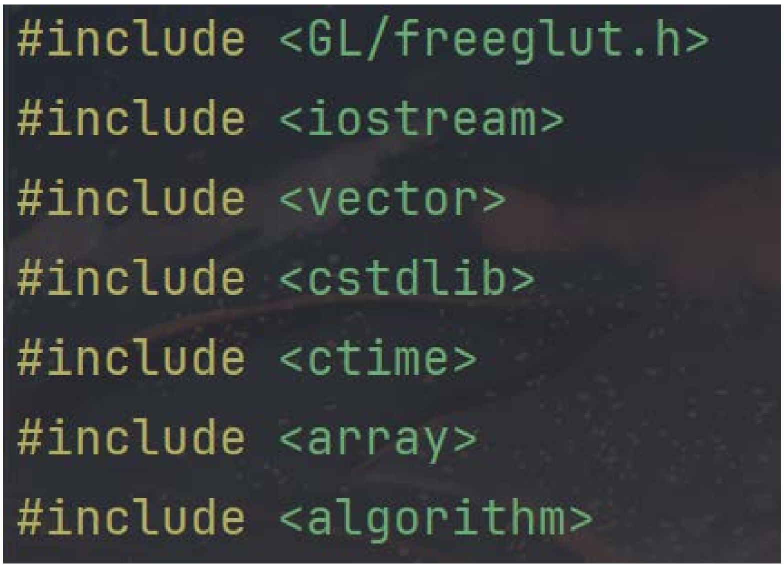
Finally, there are the conditions for **determining victory and defeat**. If the player's HP is less than or equal to 0, he will fail, and at the same time, all objects will stop, and the words GAME OVER will appear in the middle of the screen, at this time, you can press the R button to restart the game. When the player has defeated the seven types of enemies in turn, the player wins the game and the words YOU WIN appear in the middle of the screen.





That's all for the coursework.

## 3 Used libraries



```
#include <GL/freeglut.h>
#include <iostream>
#include <vector>
#include <cstdlib>
#include <ctime>
#include <array>
#include <algorithm>
```