Project Report

Team Info:

Team Name: Untitled			
Team Members	Email	Github Handle	Student ID
Yutong Li	liyutong2000@g.ucla.edu	tAnother	205117428
Senyang Jiang	senyangjiang@yahoo.com	SenyangJiang	505111806
Yiqiao Jin	ahren2040@g.ucla.edu	Ahren09	305107551

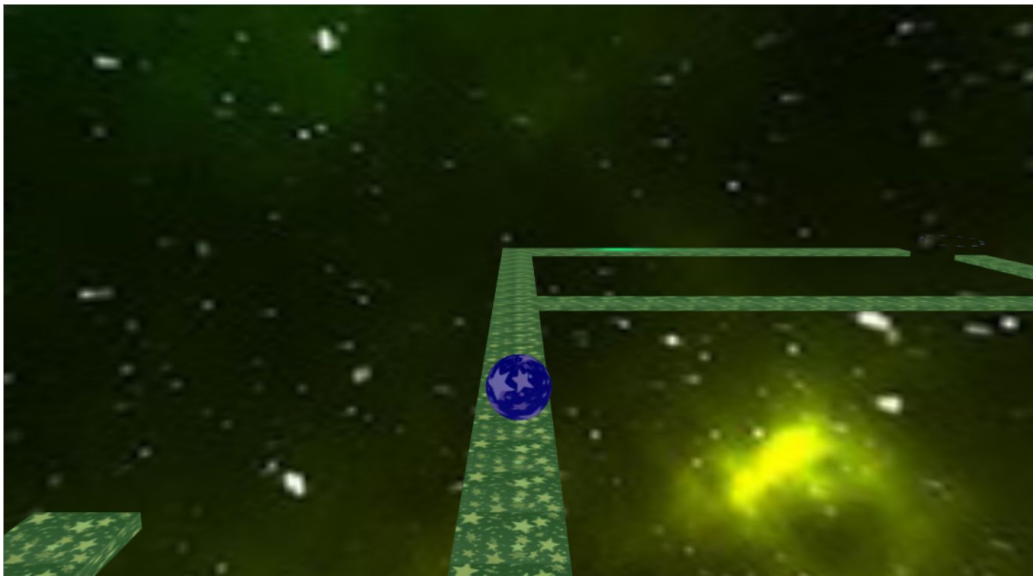
Our Idea:

We try to make a simple version of the game Ballance (2004), where the player controls a ball via keyboard to go through a course and try not to fall off screen. Following the setting in assignment 3, we place the scene in the space.

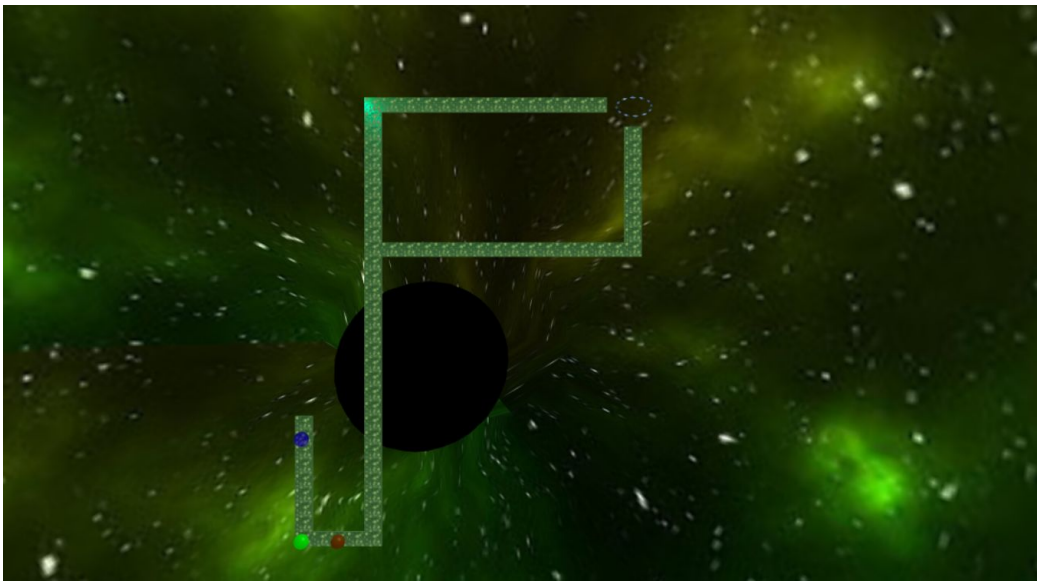
How to play:

1. To control the ball, press the keyboard:
   - I: forward
   - K: backward
   - J: left
   - L: right
2. To switch views, press the keyboard:
   - P: panorama view (camera over the whole route)
   - B: fixed view (camera attached near the ball)

   See pictures below for illustration.



Fixed view: The camera is fixed relative to the position of the ball



Panorama view: The player could see the whole course independent of the ball position.

Advanced features:

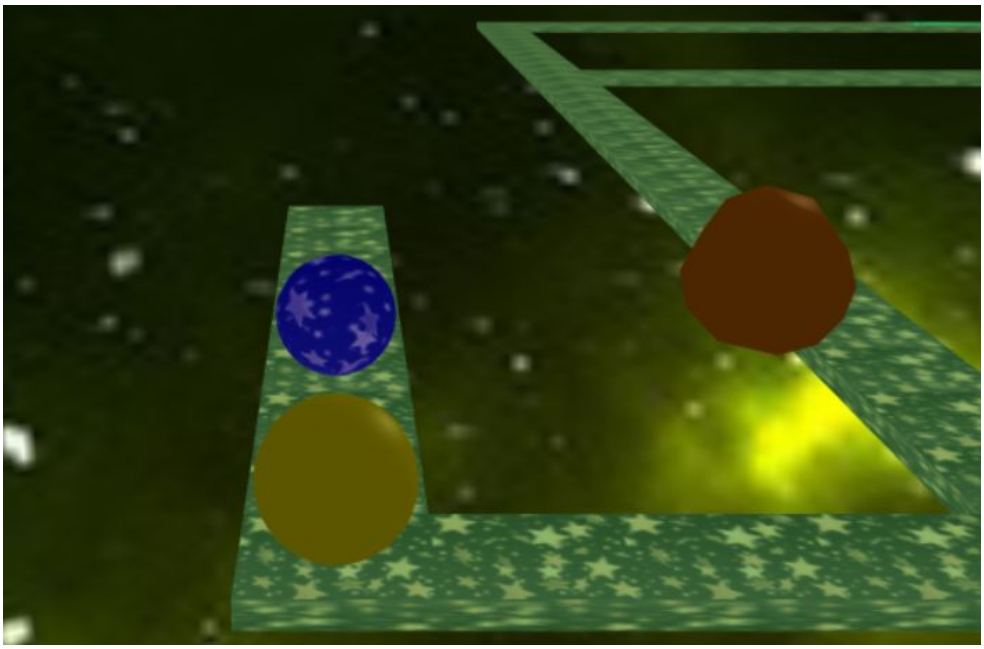
- ➢ Collision Detection:
  - Detect when the ball hit the bonus balls. This is implemented by measuring the distance between the rolling ball and the bonus balls.
  - Other related features: Detect when the ball reaches the goal / falls off the course. This is implemented by measuring if the ball's position is in certain ranges, and if the diamond ring is at a feasible angle for the ball to pass through.

- Physics-based Simulation:
    - Firstly, the ball has two velocity vectors in the plane. When any movement key is pressed, we increase or decrease the velocity linearly, and in each frame we update the position of the ball according to its current velocity. This imitates real-life scenarios where a ball obtains constant acceleration from a force, which results in accelerated motion.
    - Secondly, to implement the rolling motion of the ball, the angle displacement of the ball is calculated from the displacement of the ball from the origin, and in each frame we rotate the ball using its angular displacement.
    - Lastly, to imitate the friction between the ball and the course. When no key is pressed, the speed of the ball in both directions would decrease until it reaches 0.
    - Besides, when the ball falls off the course, it falls with an acceleration (g) till its downward velocity reaches a certain level.

We chose these features because they are rather easy to implement but can still enhance the interactivity of the game.

Special Interactions:

- Bonus Points: There are two interactive objects in the middle of the course. Using collision detection, when the ball touches these objects, the player can gain or lose points. Below are the pictures of these objects.



Two Bonus Spheres with different features and interactions

- The object on the left is a sphere that changes its color over time. When the ball that the player controls collides with the sphere, if the sphere's color is closer to green, the player gets one bonus point; if the sphere's color is closer to red, the player loses one bonus point.
- The object on the right is a sphere that moves up and down periodically. When the ball collides with the sphere, the player would get one bonus point.

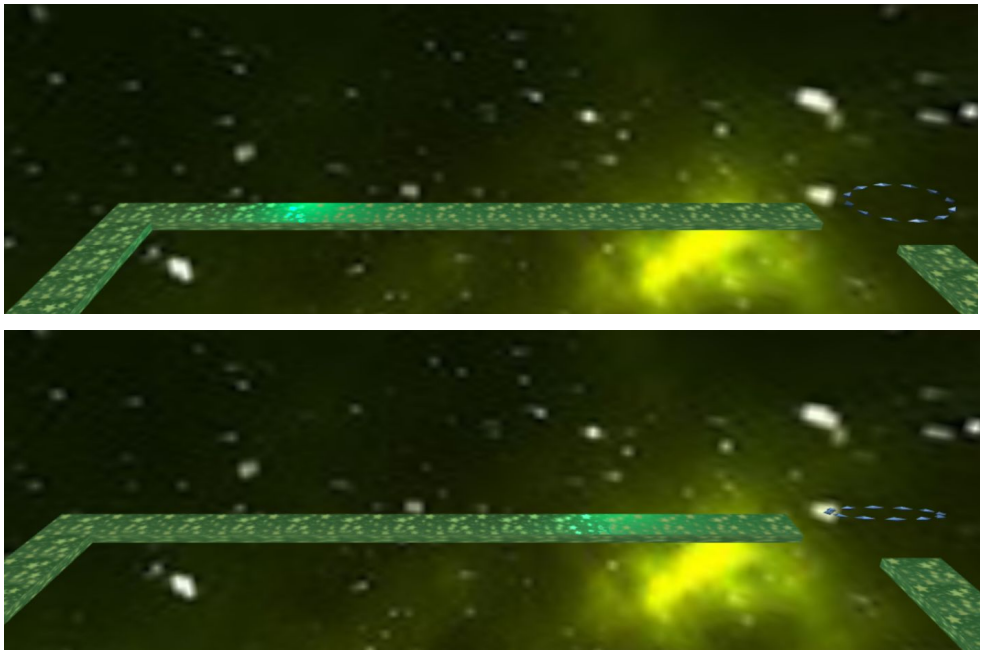
➢ Win Display: At the end of the route there is a diamond ring that resembles a destination portal. When the player goes through the course and finally falls into the ring, the diamond ring will stop flipping and start rotating around the ball, and a large win icon would appear indicating that the player has cleared the game. This win icon can be seen from both the fixed view and the panorama view.

Win icon displayed in fixed view and panorama view

Other:

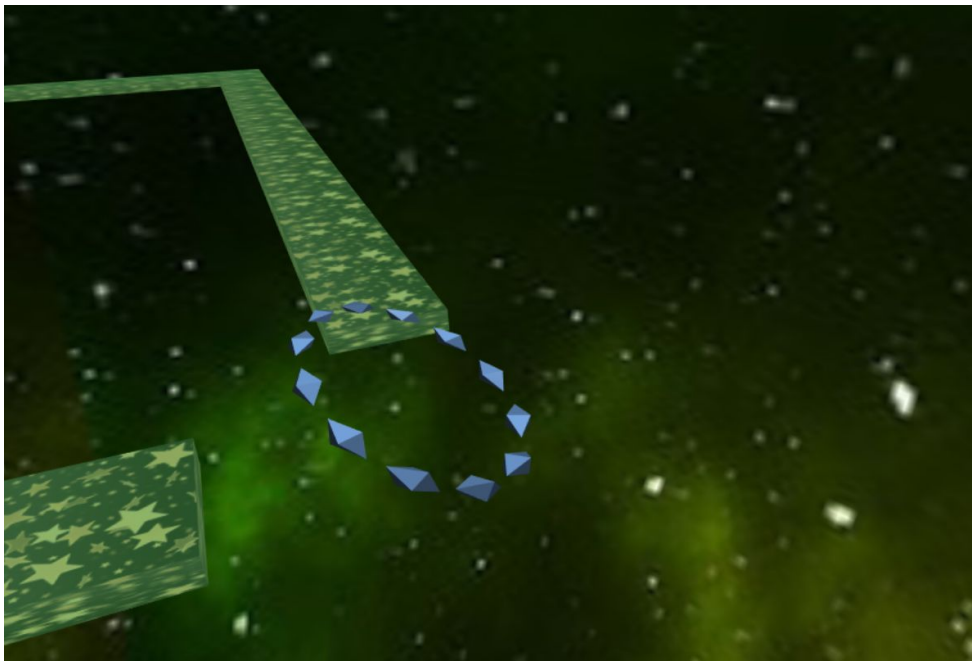
➢ Texture mapping: Many objects in our game utilize textures.
- The ball and the whole course uses the star picture from tiny graphics
- Win icon is a large square that uses the 'win' picture as its texture.
- The 'stellar' background is implemented by a large sphere which encloses the whole course, using a space picture as its texture.

➢ The moving light is inspired by the "many-light-demo" in tiny-graphics examples. It moves periodically on one of the roads that leads to the end of the orbit.



Moving light

➢ The diamond ring is made up of Cone_Tip in the tiny-graphics library using flat shading and the insert_transformed_copy_into() function. We put a fixed light source to cast light on the diamond ring so that we can see its texture better.



A closer look at the diamond ring

Contribution:

Team Members	Contribution	Link to Github Commits
Senyang Jiang	Ball movement simulation Background Camera transition Win display	**https://github.com/intro-graphics/team-project-team-untitled/commits?author=SenyangJiang**
Yutong Li	Collision / position detection Diamond ring Effects when hit the bonus / the goal	**https://github.com/intro-graphics/team-project-team-untitled/commits?author=tAnother**
Yiqiao Jin	Bonus shapes Moving lighting effects The orbit of ball	**https://github.com/intro-graphics/team-project-team-untitled/commits?author=Ahren09**