PROJECT REPORT

Ball Platformer Game

DELHI TECHNOLOGICAL UNIVERSITY
(Formerly Delhi College of Engineering)

Submitted to:

Dr. Trasha Gupta

Submitted By:

~ Aaron Paul Jacob
2K18/SE/003

~Ajay Kumar
2K18/SE/014

DELHI TECHNOLOGICAL UNIVERSITY, DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

MC318 COMPUTER GRAPHICS
2020-21 SEMESTER VI
BALL PLATFORMER GAME

ABSTRACT:

Under this project, a 2D Ball platformer was implemented through python. The workflow uses the Pygame library to facilitate the graphic layout and smooth construction of the game. The game developed has 3 levels to play. In game the main target of the player is to reach the exit gate by jumping over the platforms and the obstacles to reach the next level.

INTRODUCTION:

WHAT IS A PLATFORMER GAME?

A Platformer game is one in which

1. The player jumps and climbs between various platforms on the game field
2. The platforms often feature uneven terrain and uneven height placements.
3. Obstacles are placed in the player's path and must be overcome to reach a goal.
4. Multiple levels of increasing difficulty
5. Rewards available throughout the game
6. Ability to jump over obstacles

TOOLS AND TECHNOLOGIES USED:

1. Pygame Library:

- Pygame is a cross-platform set of Python modules that are used to create video games.
- It consists of computer graphics and sound libraries designed to be used with the Python programming language.
- Pygame was officially written by Pete Shinners to replace PySDL.

- Pygame is suitable to create client-side applications that can be potentially wrapped in a standalone executable

The general working of any game using Pygame or any other platform in general involves:

- **Input handling:** Input like pressed buttons, mouse motion, and VR controllers' position is gathered and then handled. Depending on the game, it can cause objects to change their position, create new objects, request the end of the game, and so on.

- **Game logic**: This is where most of the game mechanics are implemented. Here, the rules of physics are applied, collisions are detected and handled, artificial intelligence does its job, and so on. This part is also responsible for checking if the player has won or lost the game.

- **Drawing:** If the game hasn't ended yet, then this is where the frame will be drawn on the screen. It will include all the items that are currently in the game and are visible to the player.

2. Bresenham's Line Algorithm:

Bresenham's Line Algorithm is used for scan converting lines in the game. It is an efficient method because it involves only integer addition, subtractions, and multiplication operations. These operations can be performed very rapidly so lines can be generated quickly.

3. Painter's Algorithm:

Painter's Algorithm (it is an indirect application of painter's algorithm)
The painter's algorithm (also depth-sort algorithm and priority fill) is an algorithm for visible surface determination in 3D computer graphics that works on a polygon-by-polygon basis rather than a pixel-by-pixel, row by row, or area by area basis of other Hidden Surface Removal algorithms.

METHODOLOGY:

Pygame Clock

Times are represented in milliseconds (1/1000 seconds) in Pygame. Pygame clock is used to track the time. Time is essential to create motion, play a sound, or, react to any event. In general, we

don't count time in seconds. We count it in milliseconds. The clock also provides various functions to help in controlling the game's frame rate. The few functions are the following:

clock.tick()

- To regulate and manage the frames per second (FPS) we have used clock.tick() function which is used to set the number of frames per second.
- If no such regulation of frames is done the while loop will keep running depending system of processor speed. That is why when processor usage is high the speed of the while loop will gradually slow down resulting in the buffering of the game. And if processor is working fast the while loop will run fast resulting the faster movement the ball the usual. Which at the end can bring unevenity in the speed of the ball.
- Adding clock ensures that the game runs with a consistent refresh rate and with a continuous speed

STEPS IN BUILDING THE GAME:

1. **Game Window**
   First, the game window screen was made with the Pygame, and to set the resolution of the game window the screen height and screen width variables were initialized, and a game window of 1000 x 800 px was created. After the successful creation of the window, the world was set up.

2. **Drawing object on the screen**
   After the world creation different images and objects were loaded in the world. For this purpose, image.load() and screen.blit() functions were used.
   - image.load() method was used to load different images in the world.
   - Screen.blit() method was used to place objects in the world

3. **World Class**
   The whole game world window is divided into a grid of 20 x 16 with a tile size of 50px and each tile can used to display different objects (i.e. platform, grass, obstacles, coins, ball, etc.) of the games. Whatever image is displayed in the tile their respective sequence number as per their ordering the image resource folder is stored in the world data matrix.

4. **Player Class**
   For defining the player and its characteristics a Player Class was created and all the characteristics such as velocity along the x-axis and y-axis, collision detection, gravity, and the image of the player are assigned.

For creating horizontal movement in the ball, we have used a sequence of photos which were obtained by rotating the original picture at different angles with the help of a rotation. Then we added gravity in the player so that it can get back to the surface after every jump in the air and also restrict its jump up to a certain height only. A limit was put that the player can only jump up to 2 blocks and a restriction was imposed on the double jump if the player is in the air the user presses the jump button again it will not jump further. The player can jump again only when he is landed on the surface.

The ball has a collide function which helps in the detection of the collision of the ball with other objects of the game and based on that the further movement and other things are decided.

5. **Level Setter:**

   To design different levels in the game a level setter was created.

   It detects mouse clicks of the user and on each subsequent click, the images of the tile got changed their respective order number as per they are present in the resources folder is store in the world data matrix.

   Its interface consisting of the load button and the save button. The load button is used to load the different levels and the level is changed with help of the up and down arrow key and the save button is used to save the edited level data.

6. **Menu Interfaces:**
   - **Start screen**: the start screen interface consists of a play button that will start the game an exit button to exit the window
   - **End screen:** end screen interface has a replay button to play the same level again and an exit button to exit the game window
7. **Sound**

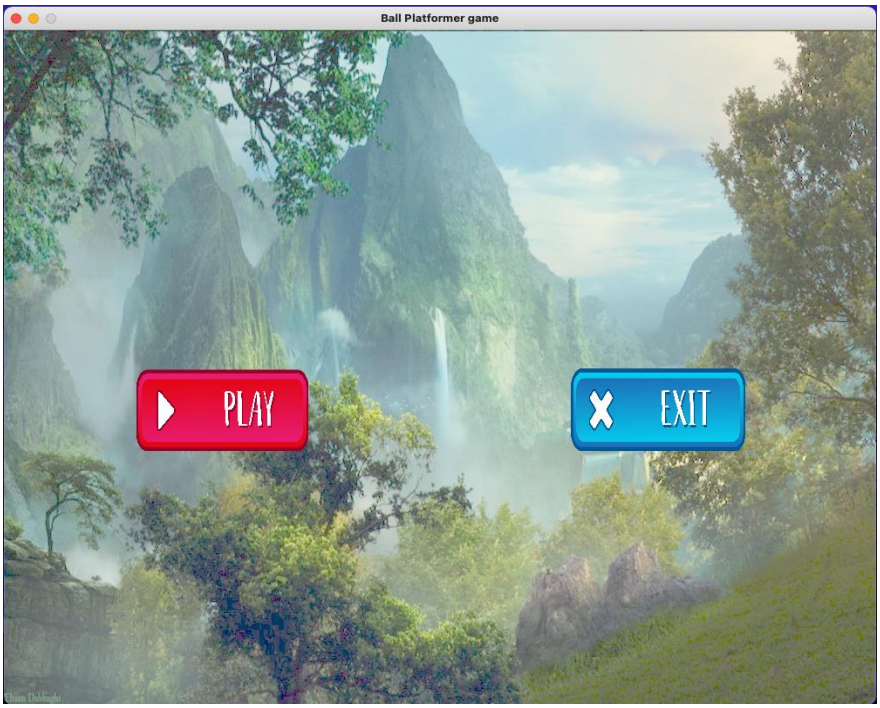
   Custom choice sounds were added to the game
   1. Background sound
   2. Jumping sound
   3. Dead sound
   4. Coin collection sound

Milestones in increasing complexity of the level:

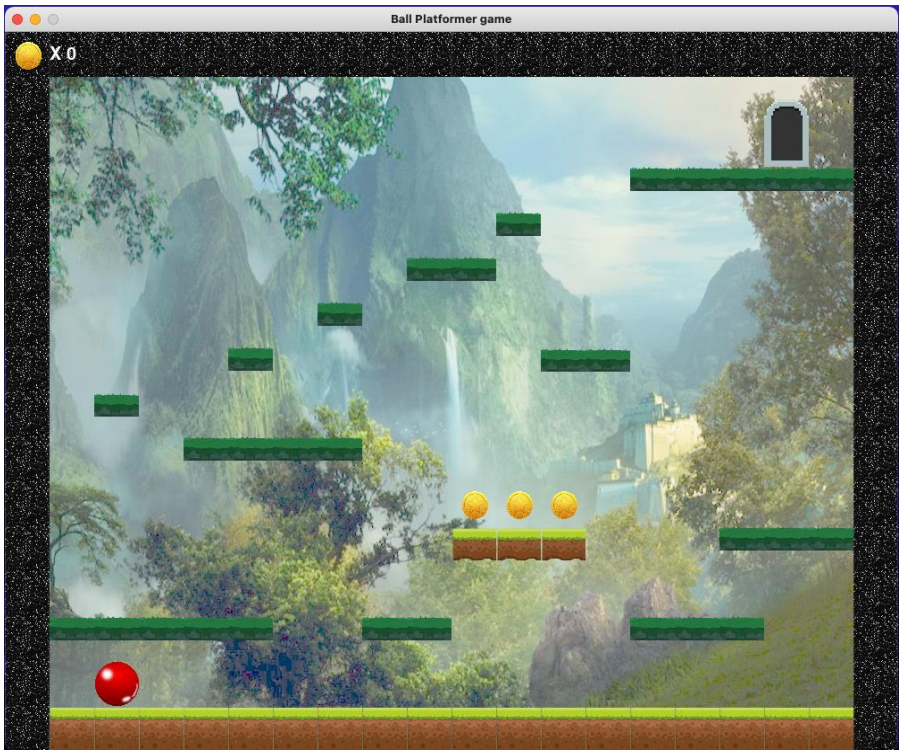
1. Simple level with coin collection
2. Moving enemies added
3. Moving enemies as well as platforms

GAME INTERFACES:

1. **Start screen**



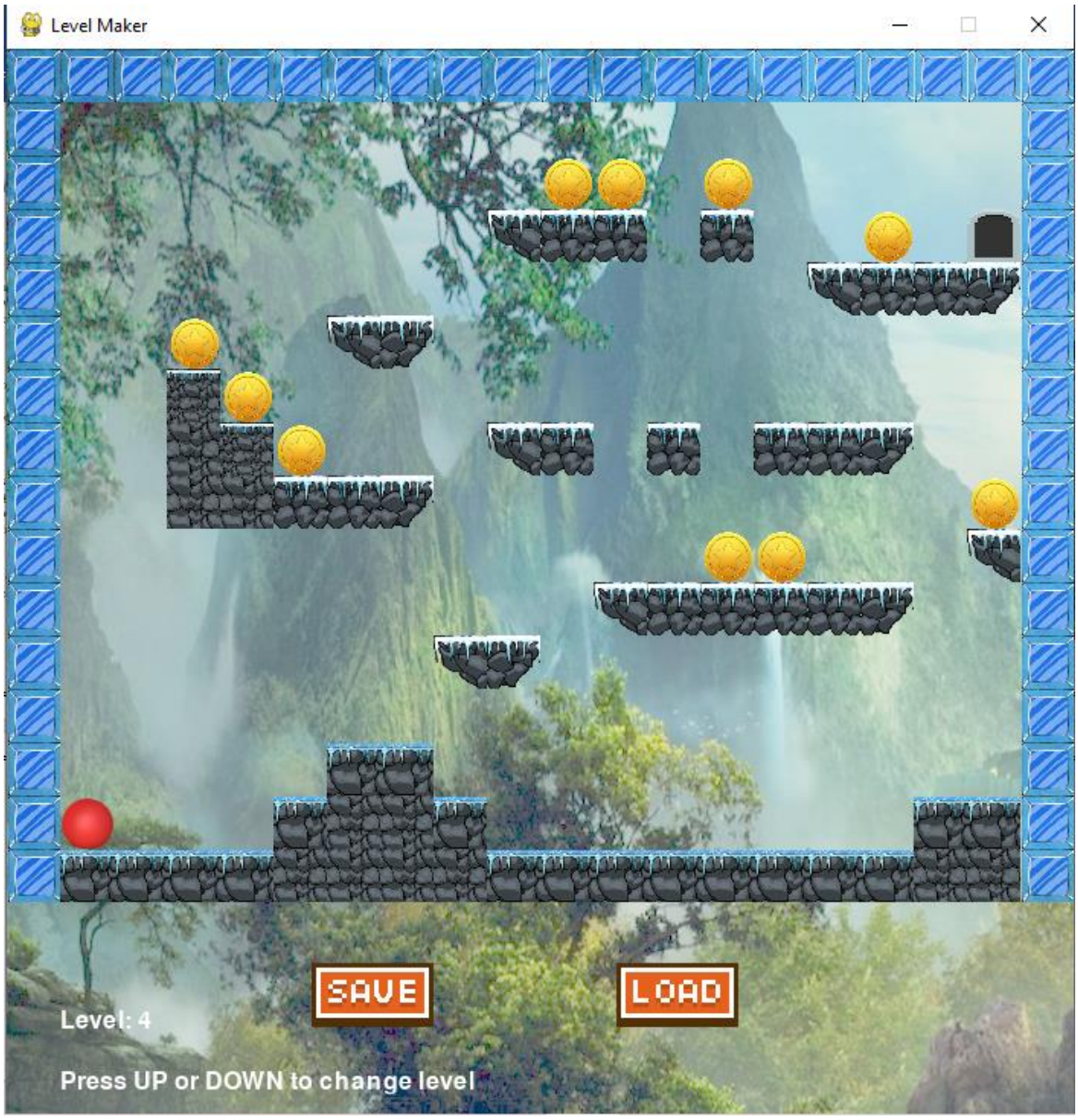
2. **Level 1**



3. **Level 2**

4. **Level 3**

5. Level setter



****