

SIT102 - HD Report 2D-Platformer Game

Saatvik Sharma
S225158822

Contents

1	Introduction	3
2	Features and Implemetation	3
3	Added Features	3
4	Evidence of tutor interaction	3
5	Iterative Development Process	3
6	Bonus objective repository	8
7	Demonstration Video	8

1. Introduction

This report is a summary and reflection on my 2d platformer game which is inspired by the classical Super Mario game. The game was created using C++ and the SplashKit library tools. The aim of the game is to reach the final flag by going over the platforms and avoiding falling down. If the player reaches the flag, the game is won and player sees the message "You Win!" on the screen, but if player falls down the game is lost and a message shows up showing you lost, but you can always try again by pressing space.

2. Features and Implementation

The game has various features such as player movement towards left and right on the ground and jumping. There is gravity physics which pulls the player down when it is in the air. Moreover, collisions are detected using AABB (Axis-Aligned Bounding Box) method, which checks if the player is colliding with the ground or the goal post. Moreover, the game simulates level progression by using an invisible goal post which is placed at the end of the level and when that post is reached the platforms are replaced and their coordinates are changed to create a new frame. This can be made to look like a continuous level progression by using the same platforms coordinates in ending of previous level and starting of the next level. Moreover, the game uses various UI elements for the platforms, player, goal post and the ground. The player is drawn using a bitmap image of Mario, and it changes according to the action the player character is doing. On top of it, the player can reload the game from the very beginning by pressing the space key, which resets the game and music plays throughout the game when the game state is PLAYING.

3. Added Features

- **Bitmap Images:** The game uses bitmap images for the player, platforms, goals and the enemies. And it has different images for the player character based on the action it is doing.
- **Better music handling:** In the game has a music loop which repeatedly plays the music and stops it when game is over.
- **Enemy AI:** The game has an enemy which moves back and forth on the platform and if the player collides with it, the game is lost. The enemy checks if it has a place to step and moves back if there is no place in front of it. Player can jump and stomp on the enemies to make them despawn.

4. Evidence of tutor interaction

5. Iterative Development Process

Before I made my game for the HD I had to make a small scale program of the game to test the physics engine and the game mechanics. I used simple boxes to represent simple platforms and the player. But now I used the Bitmap images to make this less tedious and more pleasant to look at.

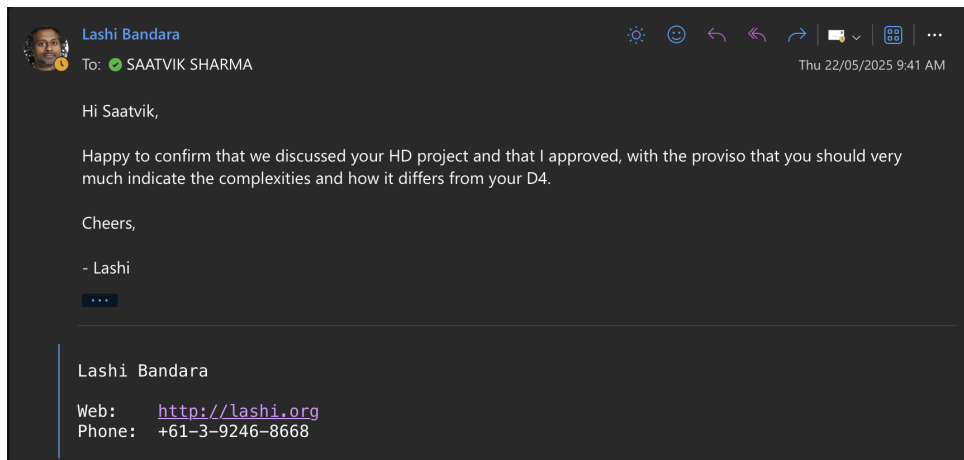


Figure 1: Screenshot of tutor interaction showing feedback on the game design and implementation.

```
void draw_game(const Player &player, const Platform *platforms, int platform_count, const Goal &goal)
{
    draw_bitmap(bitmap_named("background"), 0, 0);

    if(player.vx > 0)
        draw_bitmap(bitmap_named("player_right"), player.x, player.y);
    else if(player.vx < 0)
        draw_bitmap(bitmap_named("player_left"), player.x, player.y);
    else
        draw_bitmap(bitmap_named("player_still"), player.x, player.y);
    for (int i = 0; i < platform_count; ++i)
    {
        draw_bitmap(bitmap_named("platform"), platforms[i].x, platforms[i].y-8);
    }

    draw_bitmap(bitmap_named("goal"), goal.x, goal.y+5);
}
```

Figure 2: This is the the picture of the the final draw_game function of the game which implemented the different images for the player(for different action), platforms, background and the goal post.

```

72 void check_collisions(Player &player, Platform *platforms, int platform_count, Goal &goal)
73 {
74     player.on_ground = false;
75     for (int i = 0; i < platform_count; ++i)
76     {
77         Platform &plat = platforms[i];
78         if (aabb_collision(player.x, player.y, player.width, player.height, plat.x, plat.y, plat.width, plat.height))
79         {
80             if (player.vy >= 0)
81             {
82                 player.y = plat.y - player.height;
83                 player.vy = 0;
84                 player.on_ground = true;
85             }
86         }
87     }
88
89     if (aabb_collision(player.x, player.y, player.width, player.height, goal.x, goal.y, goal.width, goal.height))
90     {
91         level_needs_loading = true;
92         current_level++;
93
94         if (current_level > 3)
95         {
96             current_state = Win;
97         }
98         else if (current_level == 1)
99         {
100             player = {100, 400, 0, 0, 40, 40, false};
101         }
102         else if (current_level == 2)
103         {
104             player = {100, 300, 0, 0, 40, 40, false};
105         }
106     }
107 }
108

```

Figure 3: This is the picture of the final Check_collisions function which checks for the collisions between the player and the ground, goal post and the platforms. It also checks the level of the game and places the player automatically at the start of the level, giving the illusion of continuous level progression.

```

220
227 void load_level(int level, Goal &goal, Platform *platforms, int previous_goal_y)
228 {
229
230
231     if (level == 1)
232     {
233
234         goal = {730, 250, 28, 40};
235
236         platforms[0] = {0, 500, 200, 67};
237         platforms[1] = {350, 400, 200, 67};
238         platforms[2] = {650, 300, 200, 67};
239     }
240     else if (level == 2)
241     {
242
243         goal = {680, 200, 28, 40};
244
245         platforms[0] = {0, 300, 200, 67};
246         platforms[1] = {350, 350, 200, 67};
247         platforms[2] = {650, 250, 200, 67};
248     }
249     else if (level == 3)
250     {
251
252         goal = {720, 220, 28, 40};
253
254         platforms[0] = {0, 250, 200, 67};
255         platforms[1] = {300, 400, 200, 67};
256         platforms[2] = {600, 300, 200, 67};
257     }
258     else
259     {
260         current_state = Win;
261     }
262 }

```

Figure 4: This is the picture of the function which loads the level this was an addition to the previous game design which just has one level and the player had to reach the goal post to win. But now when player reaches the goal a new set of platforms and goal is created on the bases of which level the game is currently running on.

```

272
273 int main()
274 {
275
276     load_bitmaps();
277     open_window("2D Platformer", SCREEN_WIDTH, SCREEN_HEIGHT);
278
279     music game_music = load_music("background_music", "/Users/rex/Documents/Study
280     if(current_state == Playing)
281     {
282         play_music(game_music, true);
283     }
284     Player player;
285     player = {100, 400, 0, 0, 40, 40, false};
286     Goal goal;
287     Platform platforms[NUM_PLATFORMS];
288
289
290     while (!window_close_requested("2D Platformer"))
291     {
292         process_events();
293
294         if (current_state == Playing)
295         {
296             handle_input(player);
297             apply_physics(player);
298             check_collisions(player, platforms, NUM_PLATFORMS, goal);
299         }
300
301         if (current_state == GameOver && key_typed(SPACE_KEY))
302         {
303             current_state = Playing;
304             current_level = 1;
305             level_needs_loading = true;
306             player = {100, 400, 0, 0, 40, 40, false};
307         }
308
309

```

Figure 5: This is the picture of the main function of the game which runs the game but there are more things which are added here such as the music which plays throughout the game and the option to reload the game by pressing space key. Which resets the player position, level and the game state.

6. Bonus objective repository

You can view my repository and the incremental changes I made by clicking: [HERE](#).

7. Demonstration Video

You can watch the demonstration video here: [Click to watch on YouTube](#).