SIT102 - HD Report
2D-Platformer Game

Saatvik Sharma
S225158822

Contents

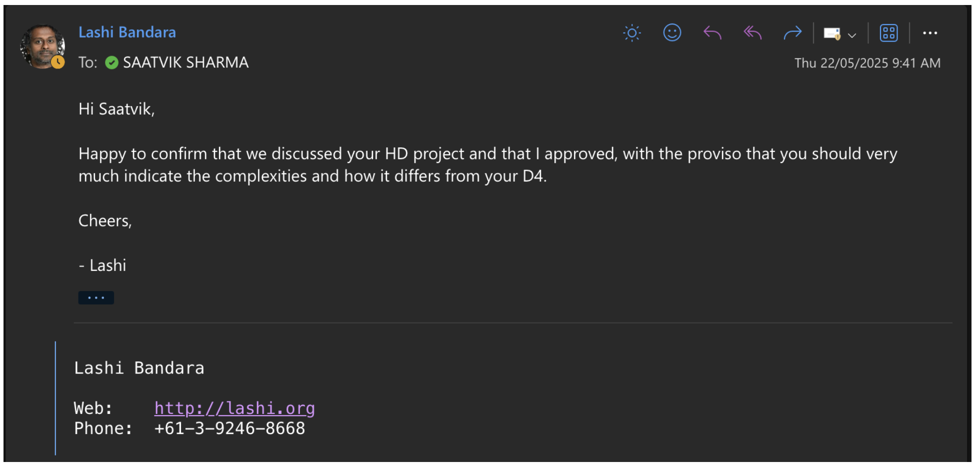
1. Introduction

This report is a summary and refelection on my 2d platformer game which is inspired 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 Implemetation

The game has various features such as player moverment 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, collisons 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 a 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 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 variosu UI elements for the paltforms, player, goal post and the ground. The player is drawn using a bitmap image of mario, and it changed according to the action player character is doing. On top of it,the player can reload the game from the very beginning by pression the space key, which resets the game and music plays play throughut the game when the game state is PLAYING.

3. Evidence of tutor interaction



Lashi Bandara
To: SAATVIK SHARMA
Thu 22/05/2025 9:41 AM

Hi Saatvik,

Happy to confirm that we discussed your HD project and that I approved, with the proviso that you should very much indicate the complexities and how it differs from your D4.

Cheers,

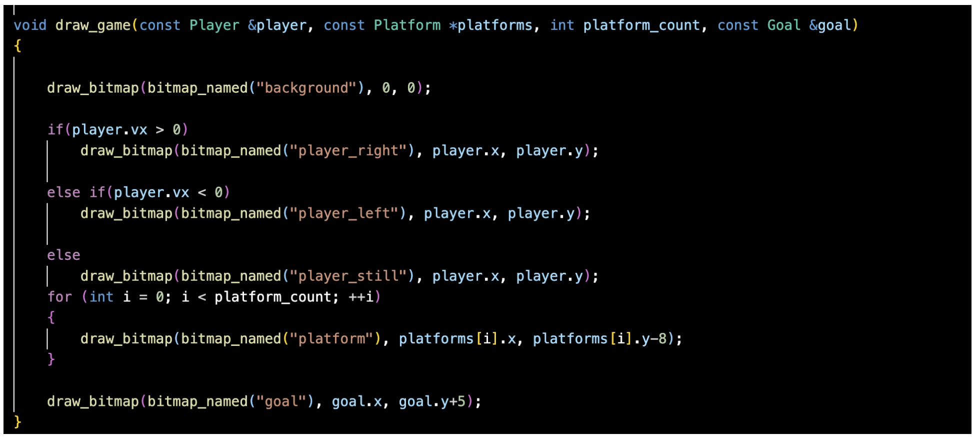
- Lashi

```
Lashi Bandara

Web:    http://lashi.org
Phone:  +61-3-9246-8668
```

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

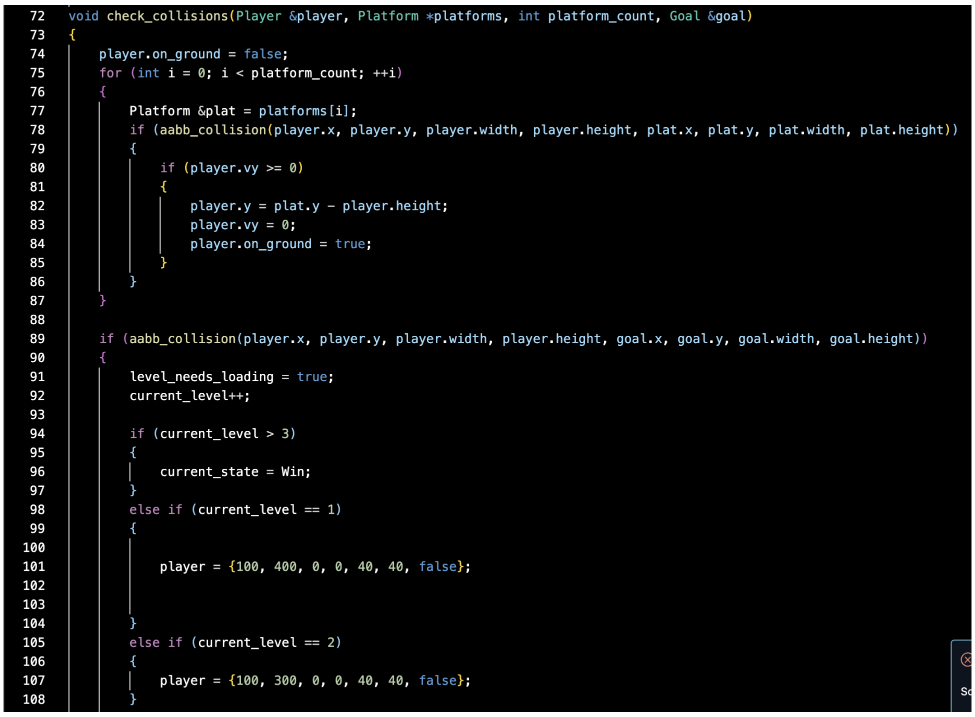
4. Iterative Development Process

Before the 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 simpel platforms and the player. But now I used the Bitmap images to make this less tedious and more pleasant to look at.



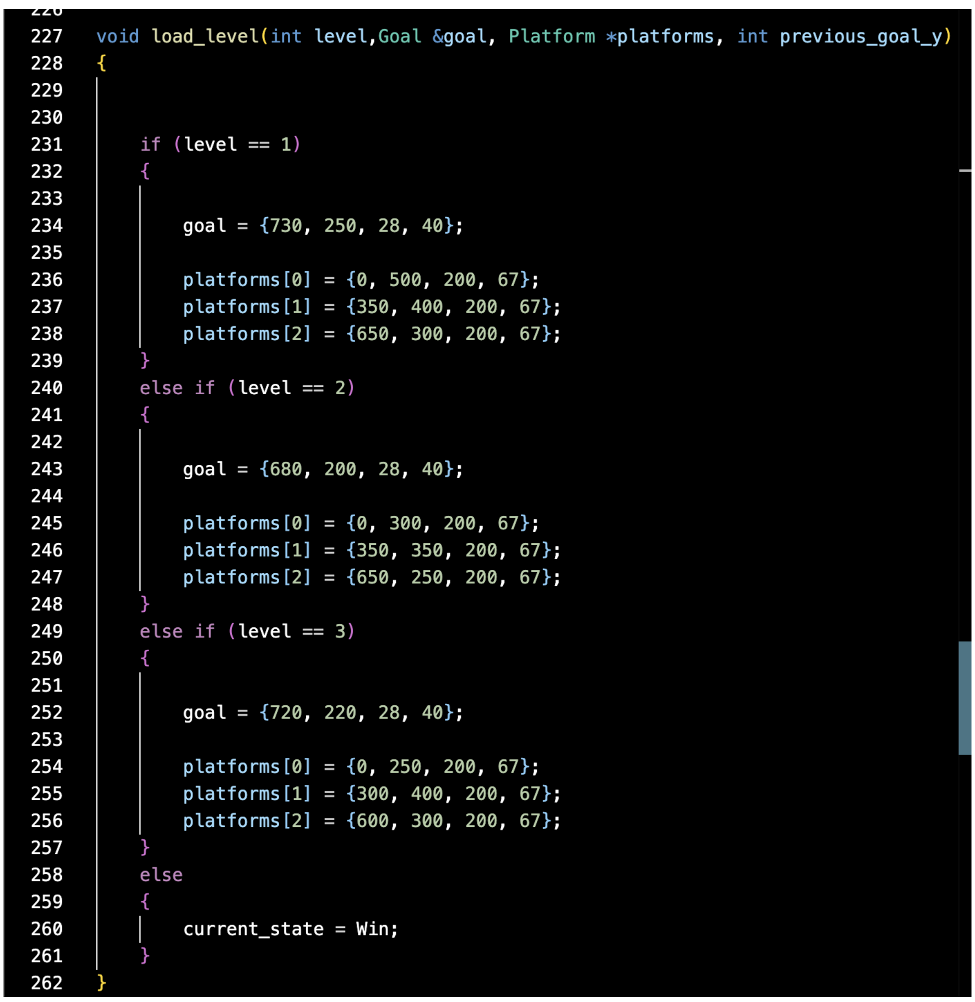
```cpp
void draw_game(const Player &player, const Platform *platforms, int platform_count, const Goal &goal)
{

    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.



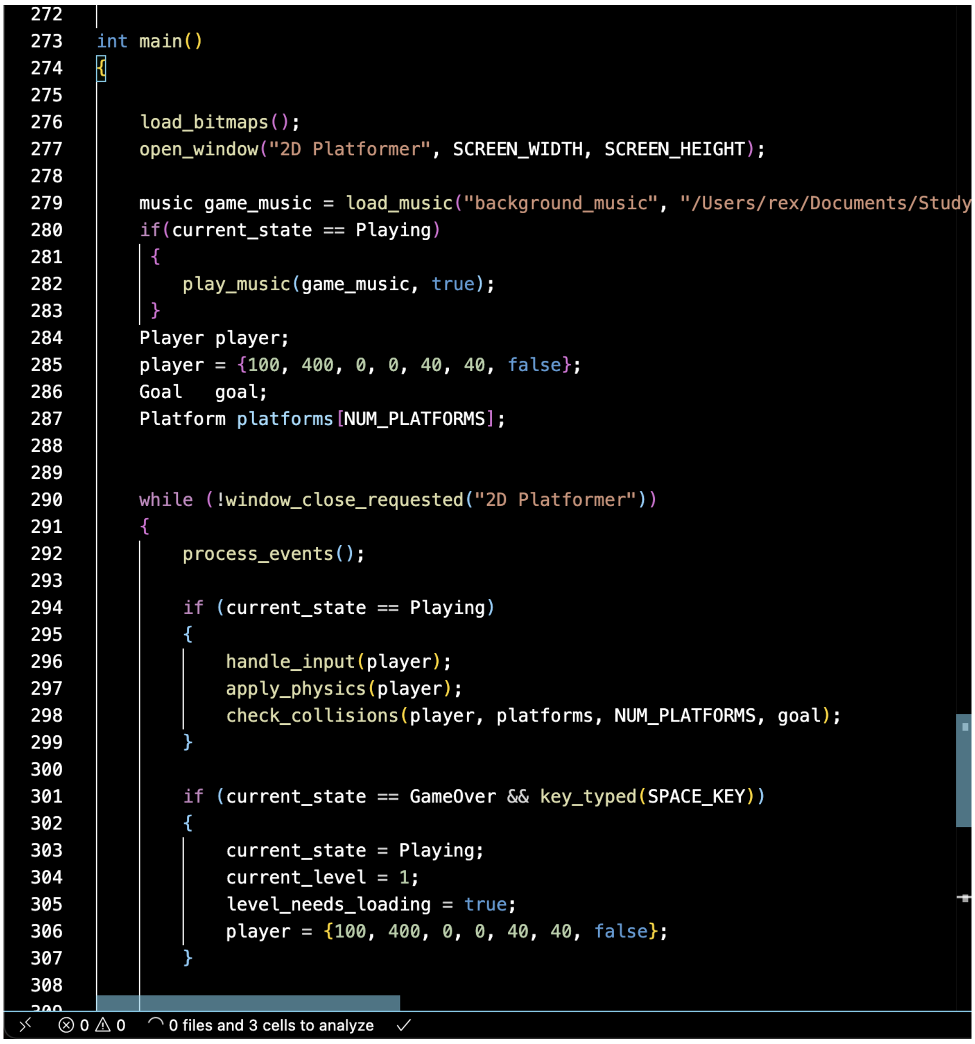
```cpp
void check_collisions(Player &player, Platform *platforms, int platform_count, Goal &goal)
{
    player.on_ground = false;
    for (int i = 0; i < platform_count; ++i)
    {
        Platform &plat = platforms[i];
        if (aabb_collision(player.x, player.y, player.width, player.height, plat.x, plat.y, plat.width, plat.height))
        {
            if (player.vy >= 0)
            {
                player.y = plat.y - player.height;
                player.vy = 0;
                player.on_ground = true;
            }
        }
    }

    if (aabb_collision(player.x, player.y, player.width, player.height, goal.x, goal.y, goal.width, goal.height))
    {
        level_needs_loading = true;
        current_level++;

        if (current_level > 3)
        {
            current_state = Win;
        }
        else if (current_level == 1)
        {

            player = {100, 400, 0, 0, 40, 40, false};


        }
        else if (current_level == 2)
        {
            player = {100, 300, 0, 0, 40, 40, false};
        }
```

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.



```
void load_level(int level,Goal &goal, Platform *platforms, int previous_goal_y)
{


    if (level == 1)
    {

        goal = {730, 250, 28, 40};

        platforms[0] = {0, 500, 200, 67};
        platforms[1] = {350, 400, 200, 67};
        platforms[2] = {650, 300, 200, 67};
    }
    else if (level == 2)
    {

        goal = {680, 200, 28, 40};

        platforms[0] = {0, 300, 200, 67};
        platforms[1] = {350, 350, 200, 67};
        platforms[2] = {650, 250, 200, 67};
    }
    else if (level == 3)
    {

        goal = {720, 220, 28, 40};

        platforms[0] = {0, 250, 200, 67};
        platforms[1] = {300, 400, 200, 67};
        platforms[2] = {600, 300, 200, 67};
    }
    else
    {
        current_state = Win;
    }
}
```

Figure 4: This is the picture of the fucntion 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.



```
int main()
{

    load_bitmaps();
    open_window("2D Platformer", SCREEN_WIDTH, SCREEN_HEIGHT);

    music game_music = load_music("background_music", "/Users/rex/Documents/Study
    if(current_state == Playing)
     {
        play_music(game_music, true);
     }
    Player player;
    player = {100, 400, 0, 0, 40, 40, false};
    Goal   goal;
    Platform platforms[NUM_PLATFORMS];


    while (!window_close_requested("2D Platformer"))
    {
        process_events();

        if (current_state == Playing)
        {
            handle_input(player);
            apply_physics(player);
            check_collisions(player, platforms, NUM_PLATFORMS, goal);
        }

        if (current_state == GameOver && key_typed(SPACE_KEY))
        {
            current_state = Playing;
            current_level = 1;
            level_needs_loading = true;
            player = {100, 400, 0, 0, 40, 40, false};
        }

```

⊗ 0 ⚠ 0   0 files and 3 cells to analyze

Figure 5: This is the picture of the main funciton 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.

5. Demonstration Video

You can watch the demonstration video here: Click to watch on YouTube.