Detailed Design

Name: Vasvi Agarwal

Andrew id: vasvia

Game Title: OG Smash N' Dash

Game Description: Dive into the fast-paced world of *Tile Break Frenzy*, a thrilling arcade game that tests your reflexes and strategic thinking. Your mission is simple yet challenging: control a moving tile platform and keep a bouncing ball in play as long as possible.

I am responsible for the additional components of the OG Smash N' Dash of the game. For all the components I will be getting a check based on which all these updates will be made in the game.

The additional components include:

- Lives of the player in the game
- Score check and update on the screen
- Level Ups in the game
- Power ups in the game which include extra lives, increasing or decreasing the speed of the ball (for few seconds) , Paddle width increase (for few seconds)

Pseudo Code/Algorithm:

```
// Global Variables
int score = 0;              //Default score when game starts
int lives = 2;              //Default lives when game starts
Int level = 1               //Default value for level
Bool up = false; //Level up check
float ball_speed = default_speed                //Speed from another file
float paddle_width = default_width              //Width from another file
bool power_up_active = false            //Check for how long the powerup is active for
time power_up_timer = 0                 //Timer to keep a check

//getting the check from another file and then adding one point for each tile that is hit
update_score_level(total_tiles) {
```

```
//Updating the score
score += 1;
if(score == total_tiles) {
        level = level_up();
        glRasterPos2i(200, 300);
                YsGlDrawFontBitmap20x28("Level " + std::to_string(level - 1 ) + " completed!");
                glRasterPos2i(200, 260);
                YsGlDrawFontBitmap20x28("Level " + std::to_string(level) + " starts now!");
}
return score, level, up;
}
//Update the levels
int level_up() {
        //Increase the ball speed
        Ball_speed +=0.5;       //Speed to be decided later

        //Decrease the paddle_width
        paddle_width -= 2.0     //Paddle_width to be decided later
        level +=1;
        Up = true;      //check to see if the level is up
        return level;
}

//Checking if the user lost a life
void check_ball_status(bool ball_fell) {
   if (ball_fell) {
      lives -= 1;           //reducing one life
      if (lives <= 0) {
         end_game();
      }
   }
}
// Applying the power ups in levels
void apply_power_up(std::string power_up_type) {
   if (power_up_type == "extra_life") {
      lives += 1;
   } else if (power_up_type == "increase_paddle_width") {
```

```
        paddle_width += 20.0f; // Temporary increase
        power_up_active = true;
        power_up_timer = std::time(0) + 5; // Power-up lasts for 5 seconds
    } else if (power_up_type == "decrease_ball_speed") {
        ball_speed -= 0.3f; // Temporary decrease
        power_up_active = true;
        power_up_timer = std::time(0) + 5;
    } else if (power_up_type == "increase_ball_speed") {
        ball_speed += 0.3f; // Temporary increase
        power_up_active = true;
        power_up_timer = std::time(0) + 5;
    }
}
void check_power_up_timer() {
    if (power_up_active && std::time(0) > power_up_timer) {
        // Reset values after power-up expires
        reset_power_up_effects();
    }
}
void reset_power_up_effects() {
    if (power_up_active) {
        // Assuming we only have one active power-up at a time
        // Reset paddle width or ball speed to default values
        paddle_width = 100.0f;
        ball_speed = 1.0f;
        power_up_active = false;
    }
}
void display_lives_on_screen() {
    // Display the number of lives on the screen for the user to track
    glRasterPos2i(10, 460);
    YsGlDrawFontBitmap16x20("Lives: " + std::to_string(lives));
}
void end_game() {
      Return false;
}
Int main {
```

```
//calling the functions to get ball_speed, paddle_width, total_tiles, ball_fell
Condition = True
display_lives_on_screen();
game_state = 1;
Switch (game_state) {
        //for scores and levels
        Case 0:
                If (tile_hit()) {
                        Score, level, up = update_score_level(total_tiles)
                        if (up == true) {
                                Up = false;
                                break;
                        }
                        Break;
                }
        //Checking for lives
        Case 1:
                If (ball_fell()) {
                        check_ball_status(true);
                }
                Game_state = 3;
                Break;
        //Power up check
        Case 2:
                 if (power_up_obtained()) {
                apply_power_up(get_power_up_type());
                }
                game_state = 4;
                 Break;
}
Return paddle_width, ball_speed, power_up, game_check
}
```