Assignment 0: Creating a Game

Jm98757 – Juan Diego Mendez

Functionality:

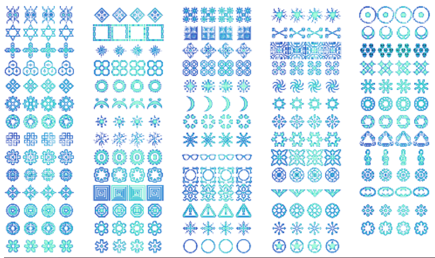
- The game features mechanics inspired by Celeste, including jump, dash, and wall slide.
- The dash is more like a teleport and is activated by right-clicking the mouse.
- Players can move left or right using the A or D keys.
- Jumping is performed using the spacebar.
- When jumping from a wall, the player is pushed away from it.
- The main enemies are parasites stuck in the walls with extending teeth.
- Touching these parasites deals 20 damage to the player's 100 life bar.
- There are Static Energy blocks that move extremely fast, effectively instant-killing the player.
- To win the game, the player must navigate a path, jump, dash, climb, and avoid parasites until reaching the end.
- Features a death screen allowing players to restart the level by pressing A or right.
- Includes a pause menu (activated by pressing Esc) with quit, restart, and resume buttons.

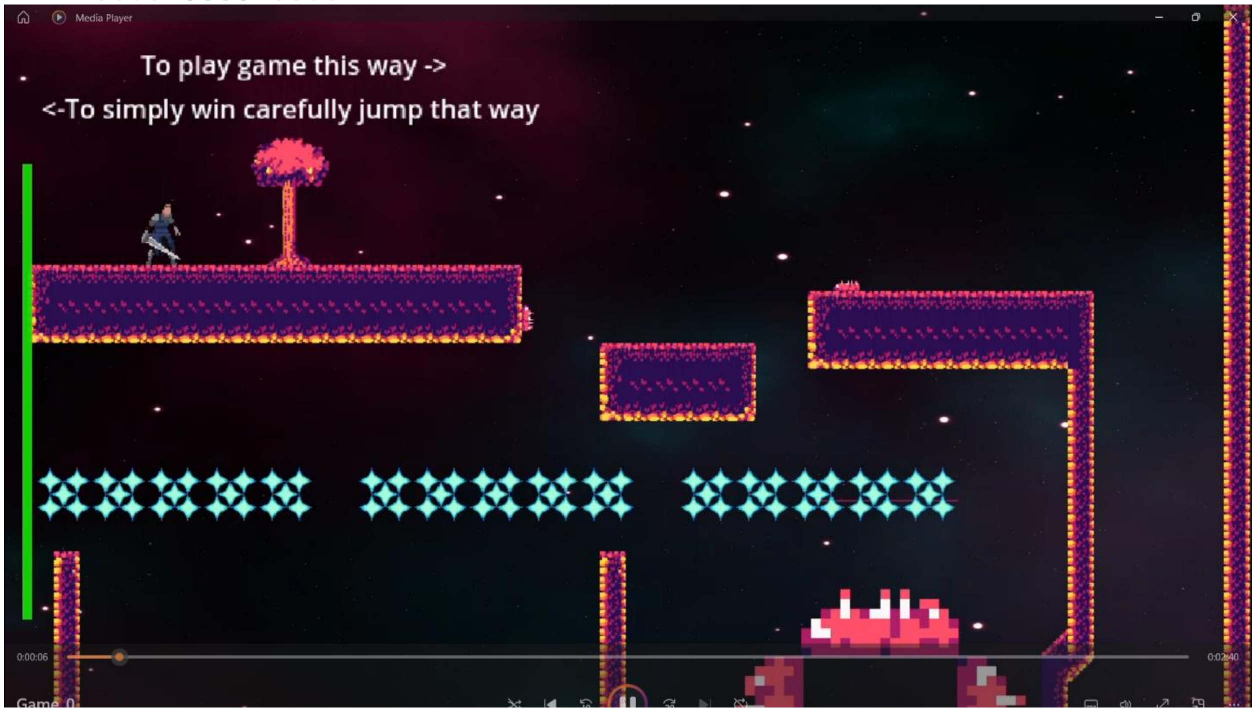
Issues:

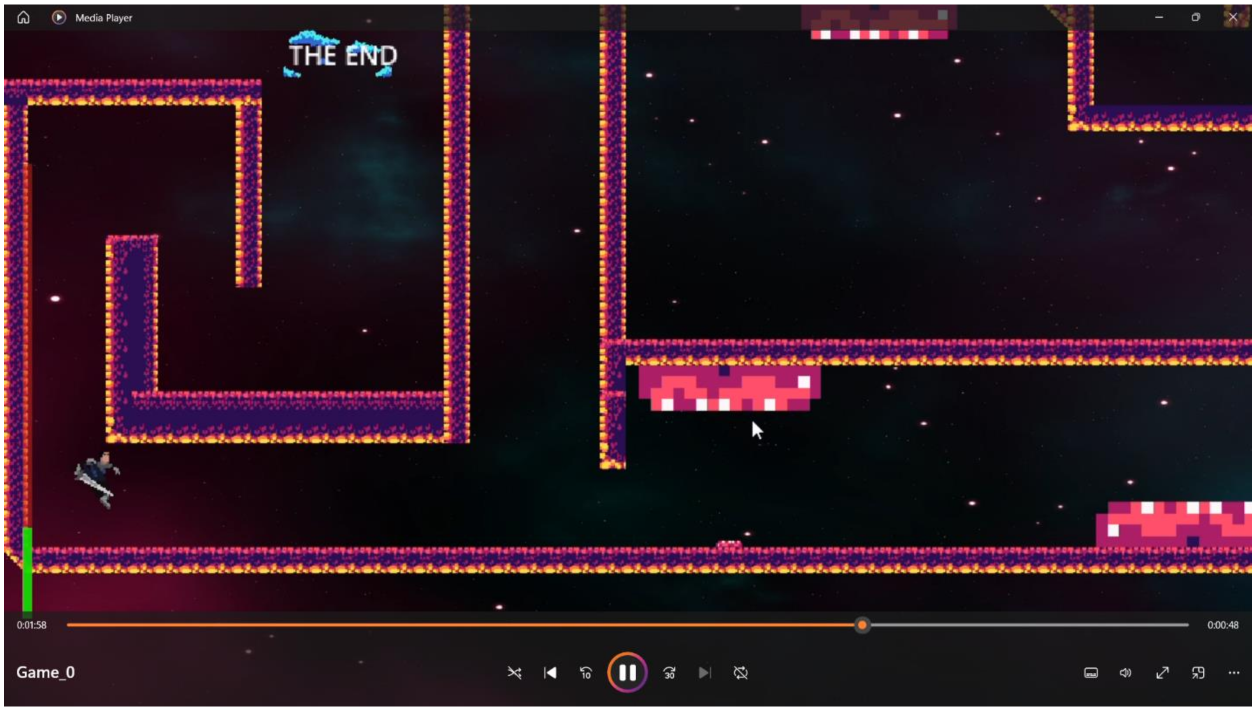
- The game mechanics and difficulty may need balancing for a better gaming experience.
- Level design and layout should be refined to provide engaging challenges.
- More instructions or tutorials may be needed to guide players on using the various mechanics effectively.
- The game's pacing and progression should be evaluated to ensure it remains enjoyable without being too frustrating.

How to Play the Game:

- Move left or right using the A or D keys.
- Jump with the spacebar. Wall jumps are possible, and you'll be pushed away from the wall.
- Dash by right-clicking the mouse.
- Avoid blue blocks and avoid the parasites; touching their teeth deals damage.
- Navigate the level until you reach the end.
- If you die, you can restart the level by pressing A or right.







Code Layout:

- The code consists of different scripts. These are Player_control, Level_0, HealthBar, game_manager, pause_menu, dangerous_obstacle and damage_block.
- Game_manager does the main things, it works a lot with pause_menu to make the pause settings work. Level_0 is the script for the main component where most stuff is.

- damage_block and dangerous_obstacle are for the energy blocks and the parasytes respectively, they send signals when colliding with player.
- HealthBar is where many signals reach, due to the fact that it is needed for defining the state of the game (win or lose)
- Player_control is also very important as it defines the most important mechanics of the game, those being running, jumping, wall sliding, wall jumping, and dashing.
- Here are some important snippets of the code:

```
            //SLIDE AND JUMP IN WALL
            if (lray.IsColliding() || rray.IsColliding()) {
                velocity.Y = gravity * (float)delta * 5f;
                if (Input.IsActionJustPressed("ui_accept")) {
                    animation.Play("Jump");
                    if (lray.IsColliding() &&
!Input.IsActionPressed("left")) {
                        velocity.X = -JumpVelocity*2.6f;
                        velocity.Y = JumpVelocity;
                    }
                    if (rray.IsColliding() &&
!Input.IsActionPressed("right")) {
                        velocity.X = JumpVelocity*2.6f;
                        velocity.Y = JumpVelocity;
                    }
                }
                wallSliding = true;
            } else {
                wallSliding = false;
            }

            // DASH
            if (Input.IsActionJustPressed("dash") && dashCooldown
==  0.0f) {
                animation.Play("Attack_1");
                if (Input.IsAnythingPressed()) {
                    Speed = 8000.0f;
                    velocity.X = animation.FlipH ? -1*Speed : 1*Speed;
                }
                // Start the timer when your code is executed.
                dashCooldown = 0.8f;
                dashTimer.Start(dashCooldown);
            } else {
                Speed = 300.0f;
            }
        } else {
            // DEATH
            animation.Play("Death");
        }
```

```
public override void _Process(double delta)
    {
        if (HealthBar != null)
        {
            AnimationPlayer ap = GetNode("AnimationPlayer") as
AnimationPlayer;
            CollisionShape2D dmg = this.GetNode("Damage") as
CollisionShape2D;
            if (ap.CurrentAnimationPosition > 0.4 &&
ap.CurrentAnimationPosition < 1.5)
            {
                dmg.Disabled = false;
            }
            else
            {
                dmg.Disabled = true;
            }
        }
    }

```

s