Sunsinger (Mechanical Alpha) Final Demo

Roberto Gonzalez, Eliakah Kakou, Michael Marinaro, Patrick Mayo and Joshua Yoon

Project Goals

- To create a simple mechanical alpha version of a game which serves as a proof-of-concept for core game features.
- Features a single playable character that utilizes pyrokinesis abilities to battle enemies and advance through the game.
- Design and implement three different demo levels that explore the different aspects of the game.
- Design several distinct enemies with unique fighting behaviors to challenge the player.
- Design several levels that challenge the player with a blend of platforming puzzles and enemy combat

GameMaker: Studio

- Cross-platform game engine
- Similar to game development software such as Unity
- Best suited for 2D, pixel-based games.
- Uses GameMaker Language (GML)

Game Maker Language (GML)

- Interpreted Scripting Language
- Similar to Python
- GML scripts allow for full control of objects and game mechanics

Event: pj.Step

Applies To: Self Other Object:

```
        tdest = instance_place(dirx, diry, b_destructible);
        if(tdest and tipo=='demolition') {
                image_index = 0;
                sprite_index = s_demolition_action;
                tiempo -= 0.5;
                with (tdest) instance_destroy();
                var destruction_frenzy;
                destruction_frenzy = true;
                while(destruction_frenzy){ //efecto dominó
                    if (dirx>x and !position_empty(dirx+42, diry)) or
                    (dirx<x and !position_empty(dirx-42, diry)) or
                    (diry>y and !position_empty(dirx, diry+42)) or
                    (diry<y and !position_empty(dirx, diry-42))
                    {
                        var tdomino;
                        if(dirx>x){
                            dirx += 42;
                        }; //derecha
                        if(dirx<x){
                            dirx -= 42;
                        };//izquierda
                        if(diry>y){
                            diry += 42;
                        };//abajo
                        if(diry<y){
                            diry -= 42;
                        };//arriba

                        //destruir y poner escombros
                        tdomino = instance_place(dirx, diry, b_destructible);
                        if(tdomino){
                            instance_create(tdomino.x,tdomino.y,b_destruido);
                            with(tdomino) instance_destroy();
                        }else{
                            destruction_frenzy = false;
                        }
                    }else{
                        //último escrombro
                        if(dirx>x and position_empty(dirx+42, diry)) instance_create(dirx+42,diry,b_destrui
                        if(dirx<x and position_empty(dirx-42, diry)) instance_create(dirx-42,diry,b_destrui
                        if(diry>y and position_empty(dirx, diry+42)) instance_create(dirx,diry+42,b_destrui
                        if(diry<y and position_empty(dirx, diry-42)) instance_create(dirx,diry-42,b_destrui
                        destruction_frenzy = false;
```

1/486: 1 INS 10 pt

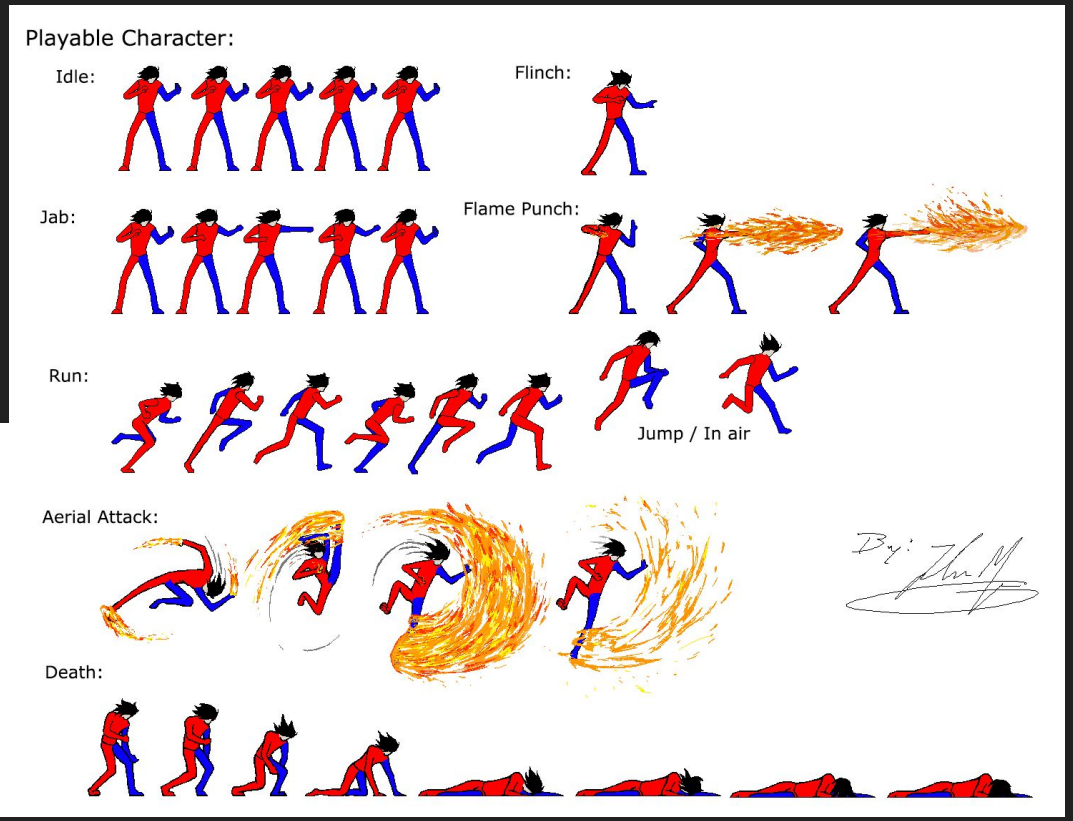
Popular GameMaker: Studio Games

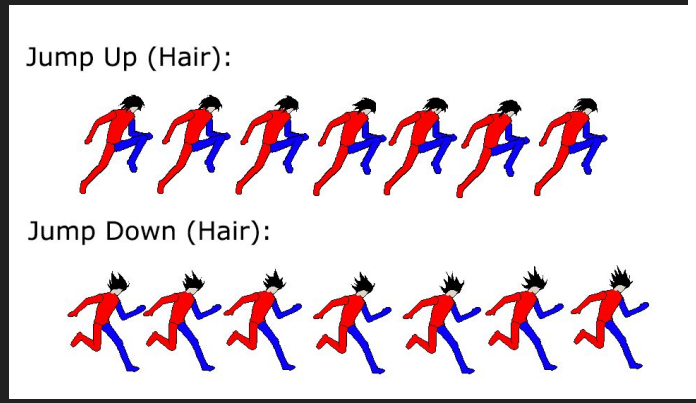
Art Assets

Player Sprites

Playable Character Sprite sheet:

- Player has in total (at the current moment in time) 46 usable sprites.
- Currently red and blue for visual and test purposes.
- All Character/Enemy Animations were drawn by Joshua Yoon.





Player Animation Examples

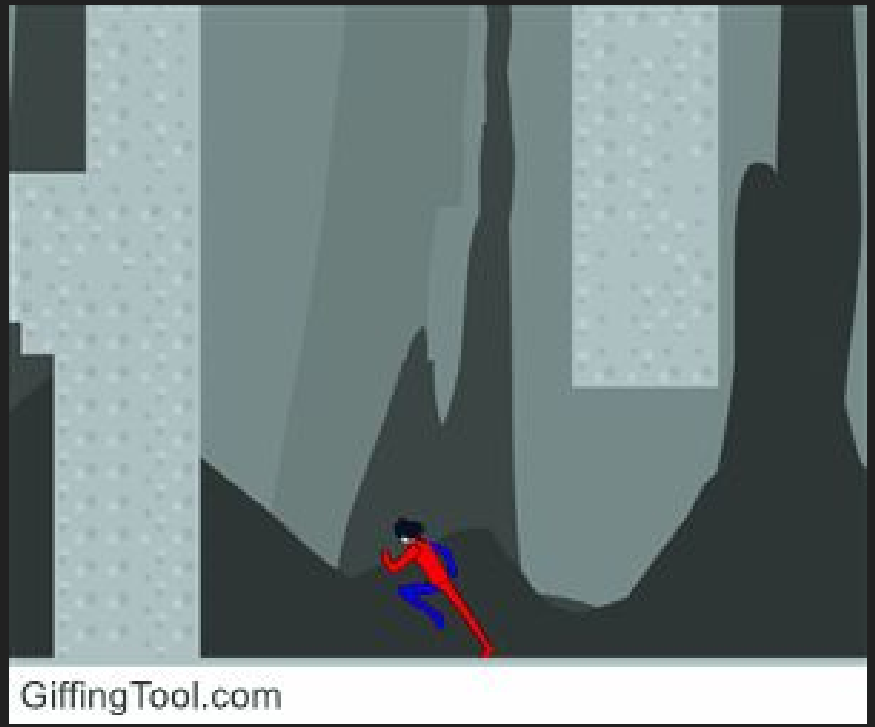
Running:

Death:

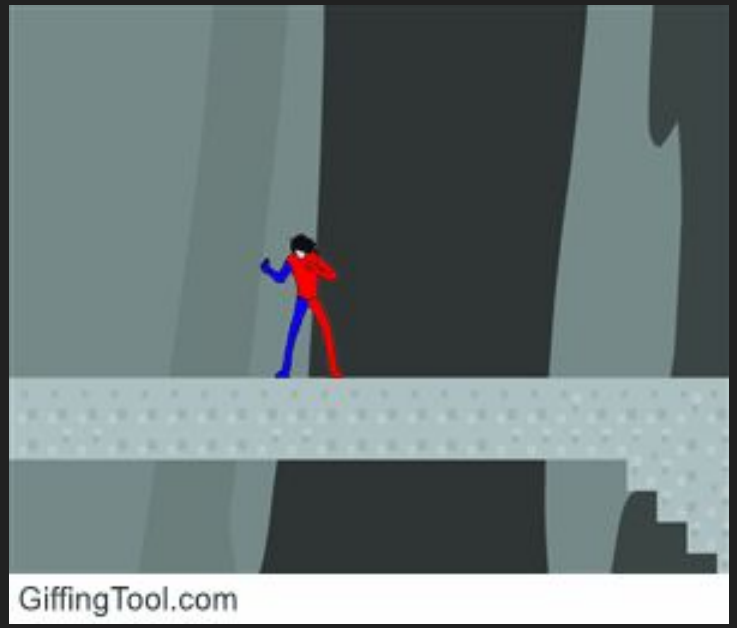
Aerial Attack:

Basic Gameplay Elements

Movement

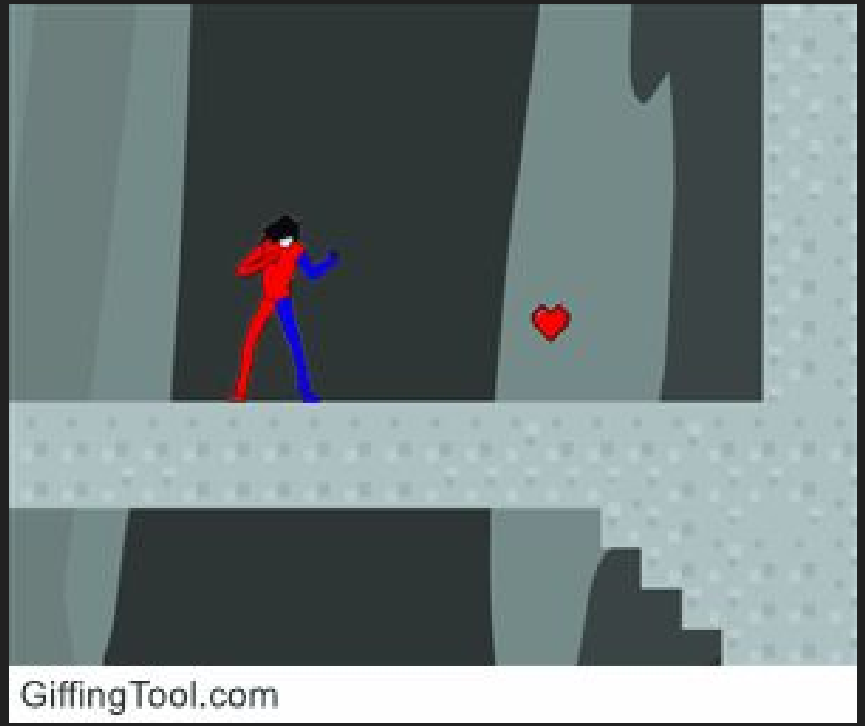


Combat



Combat - Continued

Actions



Enemies

There are currently three types of enemies:

Bird

- Flying enemy
- Attacks player that nears its patrol path

Slime

- Moves around and charges at the player.
- Spits projectiles at the player.

Minion

- Attacks the player at close range
- Jumps over short obstacles to follow player.
- Throws bouncing axes at the player.
- Attempts to dodge and counter-attack during combat.

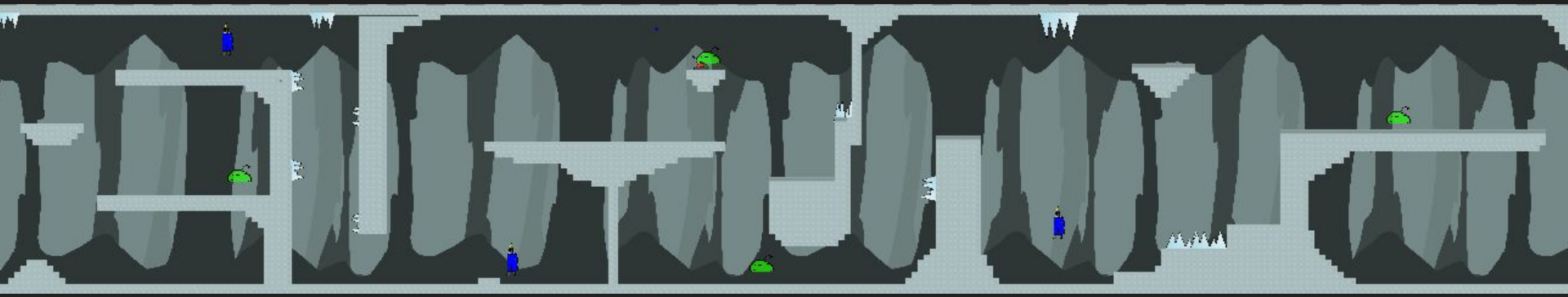
Health System

- Healthbar
  - The health-bar consists of 5 blocks each representing 1 health. It follows the player on the top left of the view.
  - A health-bar temporarily appears whenever the player is hit by an enemy.
- Health Pickups
  - On enemy death, there is a chance that they will spawn health. If the player is maxed out on health, they will not be able to pick the health up.
- Health System Functionality
  - During gameplay, the player is given 5 health blocks. When in combat with an enemy, if the player is hit, there is a brief “invincibility” period for the player to evade the enemy.
  - Once the player is out of lives, the game transitions to the Game Over screen.

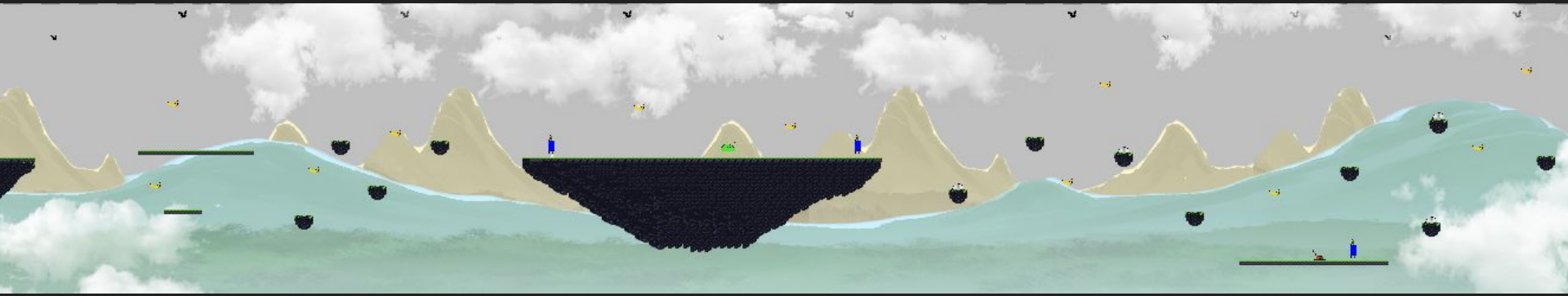
Game Environment

- Rooms
  - A level that carries a theme which is reflected in the types of obstacles that the main player encounters
- Objective
  - The objective of each room is to successfully switch the lever on, which will allow the player to exit the room after battling and avoiding traps
- Obstacles
  - Falling to death
  - Spikes
  - Enemies
- **All art assets not drawn by Joshua Yoon are free / open sourced from the web and are usable for personal / commercial use.

Cave Level



Sky Level



City Level

Demo Presentation

State Machines

- Without State Machines
  - Code quickly becomes unmanageable
  - Control flows quickly become hundreds line long If-Else statements
- With State Machines
  - Much more readable
  - Control flow separated into distinct and specific states
  - Easier to expand upon existing states and add new ones

Enemy AI

- State Based AI
  - Idle - Wait for something to happen
  - Chase - Chase down the player
  - Attack - Attack the player
  - Jump - Jump over obstacles
  - Injured - Take damage
  - Death - Play death animation, destroy instance.

- Path AI
  - Patrols a path instead of sitting in an Idle state.
  - Switches state depending on environment conditions.

Scripts

- Scripts are used to allow for modular programming
  - A script can be called inside any object instead of duplicating code inside objects
- Scripts in our project are called when objects enter a certain state
- The “take damage” script for example, is called whenever the player enters the damaged state after collision with an enemy or projectile.

Any Questions?