Crungulon's Crusade

Nicolas DiCosimo(CS '24), Laura Parente(MechE '24), Kush Poddar(Aero '24)

Rensselaer Polytechnic Institute | Rensselaer Center for Open Source

INTRODUCTION

Crungulon's Crusade is a platformer game under an open-source model, presenting meticulously designed levels that demand precision and skill in maneuvering through diverse challenges. The game prioritizes speed and fluid motion, engaging players in dynamic, physics-driven gameplay. Notably, the open-source nature of Crungulon's Crusade allows public access to its source code, fostering a collaborative environment for developers, enthusiasts, and gamers to contribute, innovate, and refine the game's mechanics and design. This model promotes an evolving and enduring gaming experience, ensuring the perpetuity and growth of the game within the gaming community.

OBJECTIVES

Preliminary Objectives

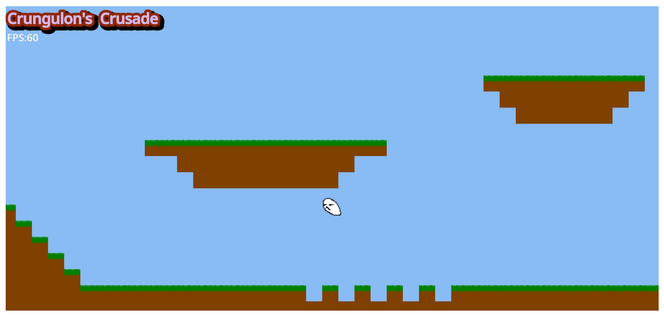
- Setup Github Repository and confirm every member can save their commits from Godot to a branch
- Get familiar with Godot and how it uses its personal script language, GDscript, and how it utilizes C#

Overall Objectives

- Create Character Controller
  - Implement a jumping system, including Coyote time
  - Implement a powerup for the playable character
- Make Basic Tile map
- Implement 2 basic enemies
- Create main menu

MATERIALS & METHODS

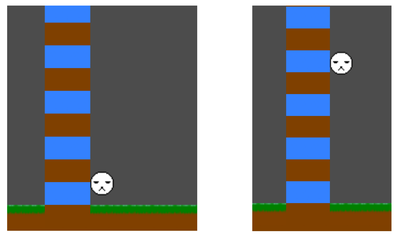
- GoDot
- C#
- Github
- Procreate



RESULTS

These results are what was accomplished by November 7th, 2023. The project is still being worked on and improved.

- A Github repository was created and has been actively used throughout the semester. Team members have gotten used to Godot and the 2 languages that Godot utilizes, GDscript and C#
- A basic character controller was created
  - This included a jumping system and coyote time, as well as a "Hold for higher jump" system.

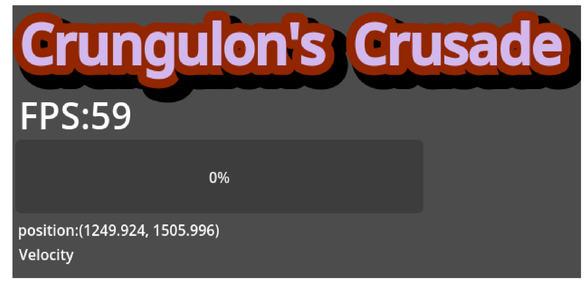


  - Also included a basic momentum system, where the player character speeds up while moving for a brief period, enabling "inching" without wildly running.

```
var velocity = Velocity;

//if jump button pressed
if(Input.IsKeyPressed(Key.Space)){
    //and on the floor
    if(IsOnFloor() || air_time < ct_thresh){
        //then you can jump
        velocity.Y = -jumpvelocity;
        air_time = ct_thresh;
        stopped_jumping = false;
    }
}
//if holding down jump
if(Input.IsKeyPressed(Key.Space) && !stopped_jumping){
    //and threshold not reached
    if(jump_time < jt_thresh){
        //keep jumping
        velocity.Y = -jumpvelocity;
        jump_time += (float)delta;
    }
}

if(!Input.IsKeyPressed(Key.Space)){
    jump_time = jt_thresh;
    stopped_jumping = true;
}

if(IsOnFloor()){
    air_time = 0;
    jump_time = 0;
    //can_jump = true;
    if(Input.IsKeyPressed(Key.Space)){
        velocity.Y = -jumpvelocity;
    }
}
else{
    velocity.Y += (float)delta*gravity;
    if(air_time < ct_thresh){
        air_time += (float)delta;
    }
}
```

- Several debugging tools were implemented, such as an FPS counter and limiter as well as a location tracker. These tools were needed in to allow systems such as coyote time to function properly. The velocity tracker, while necessary, wasn't able to be finished due to multiple bugs and errors.

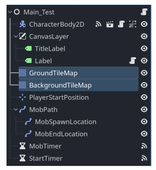
```
extends Label

func _process(_float):
    var fps = Engine.get_frames_per_second()
    text = "FPS:" + str(fps)
```

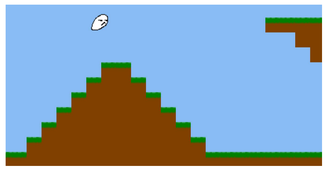



```
    {
        var canv = GetParent();
        var main = canv.GetParent();
        var node = main.GetNode<CharacterBody2D>("Node2D/CharacterBody2D");

        var velocityx = node.velocityX;
        GD.Print("velocityx");
    }
}
```

```
extends Label
func _process(_float):
    var can = get_owner()
    var node = can.get_child(0)
    var player = node.get_child(0)
    var pos = player.position
    text = "position:" + str(pos)
```

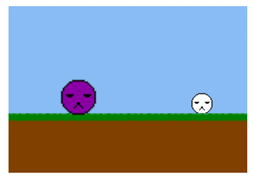
- Two basic tile maps were implemented where there was a foreground tile map and a background tile map. The foreground tile map had a collider which the player could interact with. The background tile map was for aesthetics and a place holder for future background art.



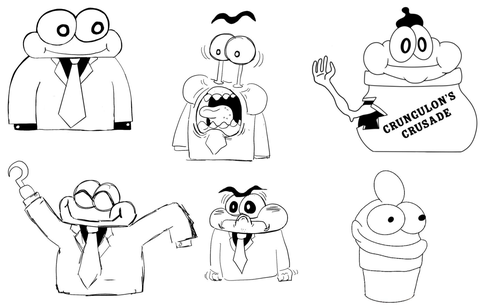


- Only one enemy was made and while it was successfully instantiated, we were not successful in making it move.

```
private void _on_mob_timer_timeout(){
    var mobSpawnLocation = GetNode<PathFollow2D>("MobPath/MobSpawnLocation");
    var mobEndLocation = GetNode<PathFollow2D>("MobPath/MobEndLocation");

    Vector2 startingPosition = mobSpawnLocation.GlobalPosition;
    Vector2 endingPosition = mobEndLocation.GlobalPosition;

    Mob mob = enemy_level.Instantiate<Mob>();
    Vector2 direction = (endingPosition - startingPosition).Normalized();
    float distance = startingPosition.DistanceTo(endingPosition);
    float timeToCoverDistance = 30.0f;
    float speed = distance / timeToCoverDistance;

    Vector2 velocity = direction * speed;
    mob.Velocity += velocity;
    AddChild(mob); }
```



- Preliminary Artwork for Crungulon was created.



CONCLUSIONS

- While we have accomplished much of what we initially set out to do, (Character Controller, enemy, debug tools, etc.) much of these feats still need improvement and more features. Specifically, the jumping looks "choppy", and coyote time is interfering with jump sensitivity, so a better implementation is needed.
- As a team, we have managed to get a lot done, even though we were using a game dev engine that was unfamiliar to all of us. However, we aren't stopping here! Throughout the rest of the semester, we will continue to implement and perfect various features, such as enemies and powerups, as well as maybe make a basic music theme for the background! With an extra 3 people on the team, we will be able to make much more progress next semester!

REFERENCES

- Godot Discord
- Godot Docs
- Godot Reddit
- Gamedev Reddit
- ChatGPT

ACKNOWLEDGEMENTS

Faculty Advisors: Prof. Turner, Prof. Goldschmidt, Prof. Kuzmin
Teaching Assistants: Anugraha Awasthi & Vansh Cheguri