Assignment 1: Platform Game Development - Godot Engine

Objective:

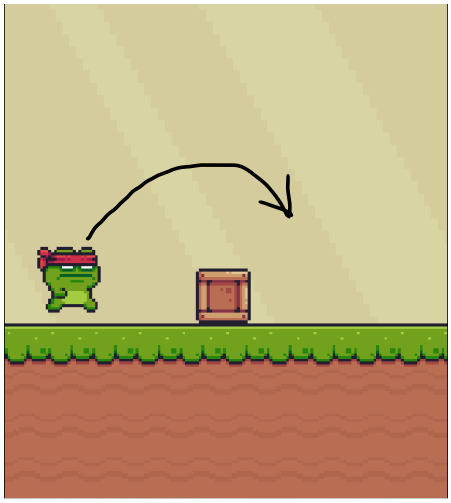
The objective of this assignment is to develop a platform game using the Godot Engine. The game should involve player movement, platform mechanics, and basic enemy interactions. You will learn the basics of game development using Godot and create a playable platform game.

1. Godot Engine:

- Install and configure the Godot Engine on your computer.
https://godotengine.org

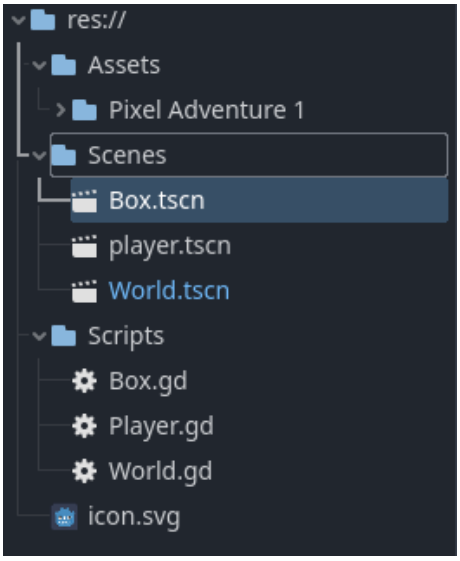
1.1 Game Overview:

- You will create a platform game using the Godot Engine.
- The Game will be an infinite scroller game, (Ex: Flappy bird or Chrome Dino Game)



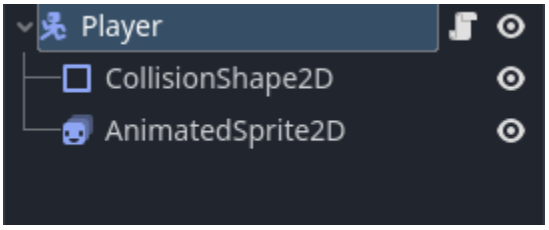
1.2 Godot Project Structure:

- Set up a new Godot project for your platform game.
- Organize your project into appropriate scenes and nodes.
- Use scenes for different levels, the main character, platforms, and enemy entities.
- Implement appropriate collision detection and physics for character movement and interactions.



1.3 Player:

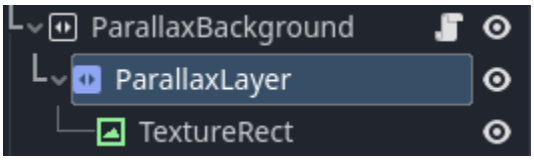
- Create a CharacterBody2D as your main Node for the player.
- Implement keyboard inputs so the player can jump.



Note: AnimatedSprite is a BONUS

1.4 Background:

- Using a ParalaxBackground + paralaxLayer to create an “infinite” background effect.



1.5 Obstacles:

- Create a new instanced scene, to create an obstacles/box to spawn. This node should contain:
    - StaticBody2D (optional)
    - Area2D (should check for collision with the player.)
    - Sprite2D
    - A Script that moves the position of the box/obstacle towards the player.
- Pick a node to spawn the obstacles from, then attach a script to that node.
- In the script, it should:
    - Using @onready, preload an obstacle node.
    - Using a TIMER node, randomly spawn those obstacles.
        - Use “instantiate()”, and “add_child()” to spawn obstacles.

Bonus:

- Add a pause functionality by pausing the tree.
- Add Coins/Collectables using Area2D.
- Add an input that allows the player to click once before starting the game, so the obstacles don’t start spawning until the player clicks on the screen (similar to flappy bird).
- Add animations to the player for running and jumping using any of the following:
    - AnimatedSprite(Recommended)
    - AnimationPlayer(Recommended for future assignments)
    - AnimationTree(Difficult but possible!)

Note:

- This assignment focuses on learning game development using the Godot Engine and creating a platform game.
- Work on this assignment individually and avoid sharing your code or solution.
- Avoid copying code directly from online sources; instead, use them as references to understand the concepts and implement your own solution.
- Consult Godot Engine's official documentation, tutorials, and online resources to learn and understand the engine's features and capabilities.
- Include comments in your code to explain your implementation and cite any external resources used.
- Maintain an organized project structure and follow best practices for code readability and maintainability.
- Test your game thoroughly and iterate on its design and mechanics to create an enjoyable gameplay experience.

Remember to refer to the Godot Engine's documentation, tutorials, and examples to help you complete the assignment successfully. Have fun and good luck with your platform game development!

MORE HELP:

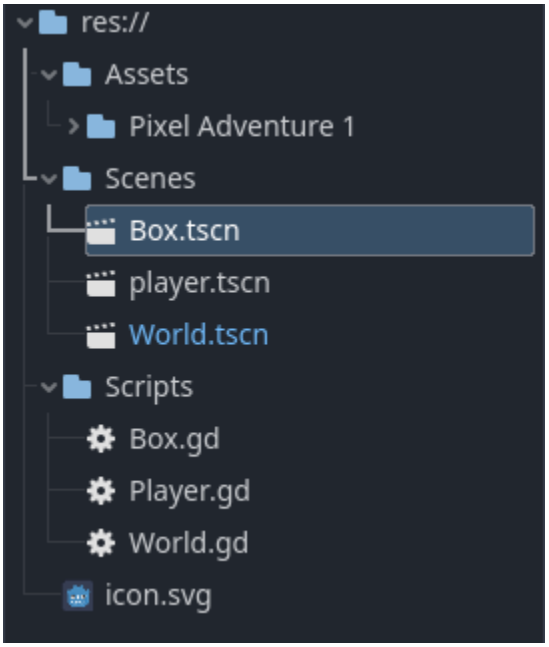
Here's a more detailed guide on how a student should navigate Godot and create their project using the nodes and concepts listed:

Finding 2D Assets:

- Before you start creating your platform game, you need 2D assets such as character sprites, platform tiles, and enemy images.
- You can find 2D assets online on various websites or create your own using software like GIMP or Aseprite.
- Once you have the assets, organize them into folders within your project for easy access.

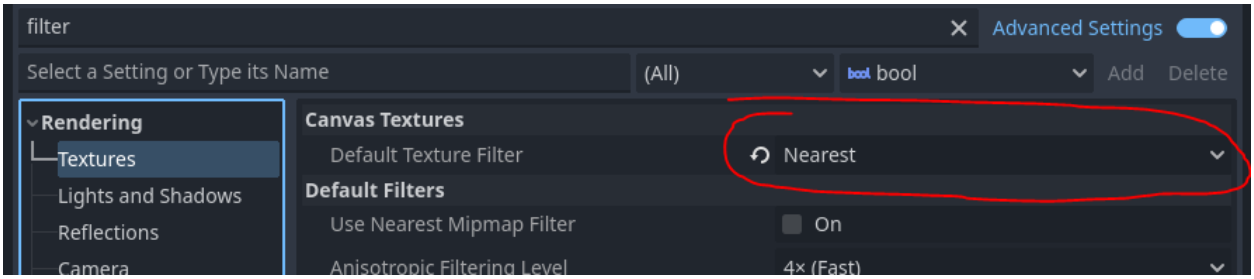
Project Structure:

- Create a new Godot project and set up the project structure.
- Organize your project into folders for scenes, assets, scripts, and any other resources you plan to use.
- A well-organized project structure makes it easier to manage and navigate your game development.



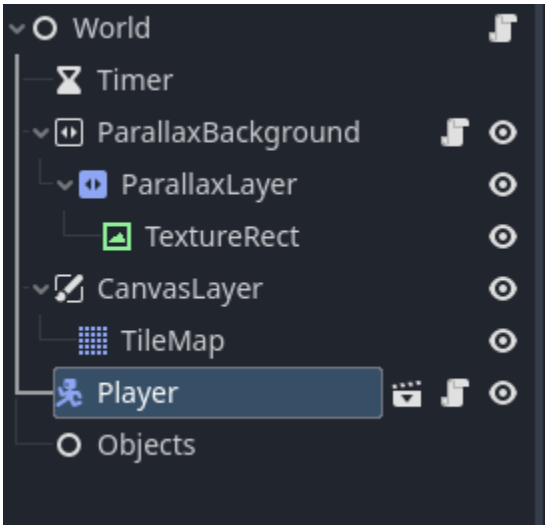
Project Settings:

- Configure the project settings. This includes setting the project name, display resolution, input controls, and more.



Scenes:

- In Godot, a scene is a collection of nodes that make up a part of your game, such as a level or a character.
- Create a new scene for your platform game's main menu, levels, character, and any other significant elements.
- You can also instance scenes within other scenes to build your game's hierarchy.



Script Templates:

- Godot uses GDScript for scripting. Create script files for nodes to add behavior to your game elements.
- Use script templates to save time and ensure you follow best practices. For example, you can use a CharacterBody2D script template for your character's movement.

Locking Nodes:

- You can lock nodes in your scene to prevent accidental modifications.
- Locking is especially useful for essential nodes that shouldn't be moved or altered once your level design is complete.

@onready var:

- @onready var is used in GDScript to create a variable that only initializes once the node it references is ready.
- This is useful for accessing nodes and resources within a scene without the need for extensive setup functions.

RECOMMENDED Nodes:

- I would recommend you get familiar with the nodes listed bellow. You should have used most these nodes for this assignment but if not, that's ok! You'll most likely get to them eventually.

1. **Node vs. Node2D**:
   "Node" is a generic node type. "Node2D" is a specialized node for 2D games. Use "Node2D" for your 2D platformer.

2. **CharacterBody2D**:
   A KinematicBody2D node used for player characters. It handles character physics, collision detection, and movement.

3. **CollisionShape2D**:
   Use CollisionShape2D nodes to define the collision shape of objects, which is essential for detecting collisions between objects.

4. **Sprite2D**:
   Sprite2D nodes are used for displaying 2D images, such as character sprites, platform tiles, and enemy graphics.

5. **Camera2D**:
   A Camera2D node provides a view of the game world. It can follow the player character to keep them in view.

6. **StaticBody2D**:
   StaticBody2D nodes represent non-moving, solid objects, such as platforms and walls.

7. **Panel**:
   Panel is a Control node used for creating UI elements like menus and HUDs.

8. **CanvasLayer**:
   CanvasLayer is a Control node for creating GUI elements that appear on top of the game world.

9. **TextureRect**:
   TextureRect nodes are used to display 2D textures on the screen.

10. **AnimationPlayer**:
    AnimationPlayer is used to create and control animations within your game, such as character animations.

By understanding and using these nodes and concepts, you can efficiently navigate Godot and create a platform game with various elements like characters, platforms, and animations. Make sure to refer to Godot's official documentation and tutorials for more in-depth information on each of these nodes and concepts.