Godot 4

Godot Showcase

- Why Godot ? (easy to learn, open source, lightweight, GDScript, C#)
- Godot Showcase
- Jungle Demo
- Realtime lighting Demo
- Godot Indirect Illumination Demo

Godot Tutorials

Essential concepts

Watch these in given order:

1. Godot 4 Essentials (playlist)
2. Godot 4 beginner tutorial
3. Google Chrome T-Rex Style Sidescroller In Godot

Installation

- Godot Engine 4
- Online Godot Editor

Today's Lecture Contents

Lecture Source Code

- Github Repo: Godot 4 Basics

Godot Engine Keywords

- Terms and Concepts (related to Unity 3D)
  - Scene
  - Nodes vs Game objects
  - Scenes vs Prefabs
  - Signals vs Events
  - Scripting

Project 1: Basics

1. Creating a new project
2. Exactly one root node per scene
3. Adding a new node (Sprite2D)

    1. Assign a texture
    2. Save the scene as Player.tscn (prefab ??)

4. Explain:

    1. Scene hirarchy
    2. FileSystem
    3. Inspector
    4. 2D view, viewing area, running the scene

5. Create a new scene (main):

    1. Create root node (Node2D)
    2. import Player as it's child
    3. Save the scene as Main.tscn and run it

        - You can change the scene dimensions in:

          `project settings -> display -> window`

6. Scripting

    1. Open Player scene. Attach a C# script to the `Sprite2D` node *(it'll fill in some template code)*. Add the following:

```
// Just like Start() in Unity
public override void _Ready()
{
    GD.Print("Player _Ready (load)");
}
// Just like Update() in Unity
public override void _Process(double delta)
{
    GD.Print("Player _Process (update)");
}
```

    Run scene, observe output.

    2. Reacting to user input: *($+ve$ y is down)*

```
public override void _Process(double delta)
{
    float AMOUNT = (float)(500.0 * delta);
    if (Input.IsActionPressed("ui_up"))
    {
        this.Position += new Vector2(0, -AMOUNT);
    }
    if (Input.IsActionPressed("ui_down"))
    {
        this.Position += new Vector2(0, AMOUNT);
```

```
        }
        if (Input.IsActionPressed("ui_left"))
        {
            this.Position += new Vector2(-AMOUNT, 0);
        }
        if (Input.IsActionPressed("ui_right"))
        {
            this.Position += new Vector2(AMOUNT, 0);
        }
    }
```

3. Explain the `Input Map` in `Project Settings` and how to add new actions.

4. Explain _Input in Godot.

```
    public override void _Input(InputEvent @event)
    {
        if (@event is InputEventKey keyEvent && keyEvent.Pressed)
        {
            if (keyEvent.Keycode == Key.T)
            {
                GD.Print("T was pressed");
            }
        }
    }
```

Nodes

```
- Game
    - Scene 1
    - Scene 2
        - Node 1
            - Node 1a
        - Node 2
            - Scene 3

- Lot of builtin nodes
    - can extend these by attaching scripts

- Rules:
    - Exactly one root node per scene
    - When a parent moves, so does child
    - When a parent is removed from the scene, so are children
    - When a parent is freed, so are children

- Use `GetNode<>(path)` to get a reference to a node in the scene
    - Use `Getparent<>()` to get a reference to the parent node
- Each node has:
```

```
    - Position
    - Rotation
    - Scale etc
    - _Ready()
    - _Process(delta)

- Demo:
    - Add a child Sprite2D node to the player _(rename to 'child'.. this
cold be a gun, weapon, item etc)_

- Getting a reference to the child node from the parent node:
    ```C#
    public override void _Process(double delta)
    {
        Sprite2D child = GetNode<Sprite2D>("child");
        child.RotationDegrees += 1;
        ...
        ...
    ```
```

Project 2: Scenes

1. Create a new scene (Enemy.tscn)
    1. Add a Sprite2D node, rename to Enemy.
    2. Assign a texture, attach C# script
    3. Save the scene
2. `Enemy.cs` script `_Process` method:

```
public override void _Process(double delta)
{
    uint randomNumber = GD.Randi() % 4;
    float AMOUNT = 5;
    if(randomNumber == 0)
    {
        this.Position += new Vector2(0, -AMOUNT);
    }
    else if(randomNumber == 1)
    {
        this.Position += new Vector2(0, AMOUNT);
    }
    else if(randomNumber == 2)
    {
        this.Position += new Vector2(-AMOUNT, 0);
    }
    else if(randomNumber == 3)
    {
        this.Position += new Vector2(AMOUNT, 0);
```

```
    }
}
```

3. Create a new scene (Game.tscn)
    - Add a Node2D as root node
    - Add 3 Enemy scenes as children (right click: Instance Child Node, or click the chain button)
    - Save the scene
4. Spawning enemies programatically:
    - Add a script to the root node of the Game scene
    - add the following method:

```
PackedScene packedScene;

public override void _Ready()
{
    packedScene = GD.Load<PackedScene>("res://Enemy.tscn");
}

public override void _UnhandledInput(InputEvent @event)
{
    if (@event is InputEventMouseButton mouseEvent){
        // create instance of Sprite scene
        Enemy enemy = packedScene.Instantiate<Enemy>();
        // Set the position of the instance
        enemy.Position = GetGlobalMousePosition();
        // Add the instance to the scene
        AddChild(enemy);
    }
}
```

Project 3: Signals and Slots

- Signals are like events in Unity
    - Slots are like event handlers in Unity
- Create a new project
    - Add a new scene (GDGuy.tscn)
        - Add a Sprite2D node, rename to `GDGuy`.
        - Add a `Timer` node as child, rename to `Clock`.
        - Assign a texture, attach C# script
    - Add a new scene (Game.tscn)
        - Add a Node2D as root node
- Add child node to Enemy: the `Timer` node, change the name to `Clock`

- Add the following code to the `GDGuy.cs` script:

```
public override void _Ready()
{
    Timer timer = GetNode<Timer>("Clock");
    timer.WaitTime = 1.0f;
    // Corrected Connect call using a Callable
    timer.Connect("timeout", new Callable(this,
nameof(OnClockTimeout)));
    timer.Start();
}

private void OnClockTimeout()
{
    float randX = (float)GD.RandRange(0, GetViewportRect().Size.X);
    float randY = (float)GD.RandRange(0, GetViewportRect().Size.Y);
    Position = new Vector2(randX, randY);
}
```