In this guide we will be experimenting with *Game Feel* using a 3rd person character with modifiable properties for speed, jump, etc.

# Step 1 - Creating the Project

A screenshot of a computer program

Description automatically generated

You can ignore a pop-up window asking to open the Asset Library as you have no projects.

Click ***+*** *New* in the top left. Navigate to the *GameFeelTemplate* folder and select the *project.godot* file.

A screenshot of a computer program

Description automatically generatedA screenshot of a computer program

Description automatically generated

Name the project in *Project Name* and navigate to the folder you want to save the project in.

Click *Create Folder* to add an empty folder in the *Project Path* folder with the name of your project. The project will be placed in this created folder.

# Step 2 - Importing Assets

Start by dragging the *Environment* folder from the Windows File Explorer into the *FileSystem* window in Godot. In the *Base* folder there are many models which can be used to build out a level.

Drag in the *Textures* folder, which includes textures for the environment models and the player drop shadow.

Drag in the *Audio* folder which contains some footsteps and jump sound effects.

Next drag in the *models* folder which contains a model of a character.

Next drag in the *Scripts* folder. Errors will appear in the *Output* window, but these will be fixed in the next step.

Drag in the *Scenes* folder which contains particles, *UI* and a *GameManager* which we will set up later.

# Step 2 - Setting up Project Settings

A screenshot of a computer

Description automatically generated

Under *Project* select *Project Settings* and open the *Input Map* tab.

To add new input actions, enter the name in the *Add New Action* text box and click *Add*.

Add input actions named *MoveLeft*, *MoveRight*, *MoveForward*, *MoveBack*. These actions form a 2D vector used for the player movement direction. Map these to WASD and/or the Joypad left stick, or however you wish.

Also add input actions named *CameraLeft*, *CameraRight*, *CameraUp*, *CameraDown*. These actions form a 2D vector used to rotate the camera with a controller, but are not necessary for mouse movement.

Add input actions *Jump*, *Sprint* for additional movement controls.

Finally add an input action named *Pause*, mapped to the *Escape* key and/or the *Joypad +* button.

A screenshot of a computer

Description automatically generated

The *GameManager* is a script which handles events in the game. It needs to be set up as an *Autoload* in Godot. In the *Autoload* tab in Project Settings, click the folder icon and navigate to *GameManager.gd* in the imported *Scripts* folder. Finally click *Add*.

In the *General* tab and the category *Display* > *Window* in the left panel, the *Viewport Width* and *Viewport Height* can be set to change the size of the game window. E.g. *1280 x 720* or *1920 x 1080*. Mode can also be set to *Windowed* or *Exclusive Fullscreen*.

# Step 3 - Creating the Main Scene

The Main scene will be the root scene opened when the game is run. This scene will contain the level scene, player scene, as well as the UI and particles.

A screenshot of a computer

Description automatically generated

In the *Scene* window select *Other Node* and then select *Node*. The level, player, etc scenes will later be placed inside of this main scene.

In the *General* tab of *Project Settings*, under the *Application* category and then *Run*, set the *Main Scene* to this scene.

Drag and drop the *UI.tscn* scene from the Scenes folder into the Main scene.

# Step 4 - Creating the Level Scene

A screenshot of a computer

Description automatically generated

To create a new scene, open a new tab along the top. In the *Scene* panel, select *3D Scene*. This will be the root node of the level scene.

A screenshot of a computer

Description automatically generated

Double-click on the created node in the *Scene* panel to rename it, e.g. to *Level* and save it with the same name.

A screenshot of a computer

Description automatically generated

Click the **+** in the *Scene* tab to add a *StaticBody3D*, with a *CollisionShape3D* and *MeshInstance3D* as child nodes. Double-click on the root *Node3D* node to rename it, e.g. to *Level*.

A screenshot of a computer program

Description automatically generated

While selecting the *CollisionShape3D* in the *Scene* hierarchy, in the *Inspector* add a *BoxShape3D* to the *Shape* property of the *CollisionShape3D*. Set its *Size* to be wider, e.g. 100, 0.5, 100. This will be the shape of the level floor which the player will collide with.

**Make sure to set the *Size* of the *CollisionShape3D Shape Size* using shape size, not the scale of the *Node3D*.**

A screenshot of a computer

Description automatically generated

Add a *BoxMesh* to the *MeshInstance3D* with the same *Size* as the collision *BoxShape3D*. This will be the visual representation of the floor.

Add a *New StandardMaterial3D* to the *Material* property of the *Mesh*.

Set the colour of the ground using the *Albedo* property of this material.

A screenshot of a computer

Description automatically generated

Click the three vertical dots along the panel above the viewport, and click *Add Sun to Scene* and *Add Environment to Scene* to add a default *DirectionalLight3D* and a default *WorldEnvironment*.

We now have a basic level to place a player in. Drag and drop the *Level.tscn* from the *FileSystem* into the *Main* scene.

Additionally from the FileSystem panel you can drag and drop objects from the *Base* folder in the *Environment* folder to build out a level.

# Step 5 - Creating the Player Scene

To create the player scene, first open a new tab.

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

In the *Scene* panel, select *Other Node* and search for *CharacterBody3D*.

A screenshot of a computer

Description automatically generated

Click the **+** in the *Scene* tab to add *CollisionShape3D* node just like with the level scene.

Do not worry about the warning when adding the CollisionShape3D, it is only because a shape has not yet been added. Selecting the *CollisionShape3D*, in the Inspector add a *CapsuleShape* to the *Shape* property.

Double-click on a node to rename it, e.g. *CharacterBody3D* to *Player*.

A screenshot of a computer

Description automatically generated

Drag the sophia*\_skin.tscn* from the imported *models* folder from the *FileSystem* window into the scene.

You can change the radius and height of the capsule collision shape, as well as the vertical y position, to align it with the character model.

A screenshot of a video game

Description automatically generated

You should now have a *Player* scene like this.

Add a *Node3D* and name it *CameraPivotHorizontal*, then add a child *Node3D* to this name named *CameraPivotVertical*. To this node, add a child of type *SpringArm3D* (by searching in the list of node types). These nodes handle pivoting around the camera and the distance from the camera. As a child of the *SpringArm3D*, with this you can define the distance the camera will be from the character with *Spring Length*. Finally add a child *Camera3D* node.

A screenshot of a computer

Description automatically generated

Next select the Player root node (*CharacterBody3D*) and in the *Inspector* click where it says *<empty>* in the *Script* property.

Select *Quick Load* and select the imported *Player.gd*.

* Camera variable unnecessary
* Remove JumpParticlesRayCast
* Exclude Sophia from springarm

# Step 6 - Sprint Particles

Sprint particles are already set up in the player script, so all that is needed is for the *SprintParticles.tscn* file in the imported Scenes folder to be dragged from the *FileSystem* into the *Main* scene.

# Step 7 - Audio Players

Drag and drop the *AudioPlayers.tscn* into the player scene. This will automatically play footsteps and jump audio using the player script.