## How to Create an Exercise

Here is a step-by-step walkthrough on how to create a simple exercise for Auditory Ace using GDscript in Godot.

### Step 1: Plan the Exercise

1. **Define Exercise Objective**: Clearly define the purpose and objective of the exercise. Understand what skills or knowledge you want the user to acquire or practice.

2. **Design User Interface**: Sketch or design the user interface layout for the exercise. Determine the elements, such as buttons, labels, and graphics, needed to create the exercise interface.

### Step 2: Set Up the Scene

1. **Create a New Scene**: In Godot, create a new scene.tscn for the exercise. This scene will contain all the elements required for the exercise.

2. **Add Nodes**: Add necessary nodes to the scene, such as buttons, labels, color rectangles, and any other UI elements needed for the exercise.

Example button code:

**[node name="Button1" type="Button" parent="Button"]**

**offset\_left = 173.0**

**offset\_top = 268.0**

**offset\_right = 473.0**

**offset\_bottom = 370.0**

**theme\_override\_colors/font\_color = Color(0, 0, 0, 1)**

**theme\_override\_fonts/font = ExtResource("1\_1671l")**

**theme\_override\_font\_sizes/font\_size = 70**

**theme\_override\_styles/normal = SubResource("StyleBoxFlat\_8dnn1")**

**theme\_override\_styles/hover = SubResource("StyleBoxFlat\_yyuig")**

**text = "Cat"**

Example label code:

**[node name="AppTitle" type="Label" parent="Background/ColorRect"]**

**layout\_mode = 0**

**offset\_left = 163.0**

**offset\_top = 126.0**

**offset\_right = 943.0**

**offset\_bottom = 206.0**

**theme\_override\_colors/font\_color = Color(0, 0, 0, 1)**

**theme\_override\_fonts/font = ExtResource("13\_s6hby")**

**theme\_override\_font\_sizes/font\_size = 60**

**text = "Tap to hear again"**

**horizontal\_alignment = 1**

**vertical\_alignment = 1**

Example Color Code:

**[node name="ColorRect" type="ColorRect" parent="Background"]**

**offset\_right = 1024.0**

**offset\_bottom = 576.0**

**color = Color(0.631373, 0.823529, 0.945098, 1)**

3. **Organize Nodes**: Organize the nodes within the scene hierarchy according to their relationships and functionality.

### Step 3: Implement Exercise Logic

1. **Extend Node2D Class**: Create a new script.gd file that extends the Node2D class. This script will contain the logic for the exercise.

2. **Define Variables**: Declare variables for nodes and other necessary data structures. Use @onready to initialize node references.

3. **Implement Callback Functions**: Implement callback functions for user interactions, such as button presses. These functions will handle user input and trigger appropriate actions.

4. **Define Game Logic**: Write functions to generate words, check answers, update UI elements, and manage game flow based on user input.

### Step 4: Integrate Audio and Visual Assets

1. **Prepare Audio Assets**: If the exercise involves audio, prepare audio files for words or feedback sounds. Load and play these audio files as needed in the exercise logic.

2. **Prepare Visual Assets**: Prepare any visual assets required for the exercise, such as icons, buttons, background images, and UI elements. Ensure they are appropriately sized and formatted.

### Step 5: Test and Debug

1. **Test Exercise Flow**: Playtest the exercise to ensure it functions as intended. Verify that user interactions trigger correct responses and UI updates.

2. **Debug Logic**: Debug any issues or errors encountered during testing. Use debugging tools provided by the game development environment to identify and fix problems.

### Step 6: Add in Pre- and Post- Exercise Screens

1. **Add in Pre-Exercise Screen:** Create a modified version of the existing pre-exercise screens that will fit in with the current exercise.

2. **Add in Post-Exercise Screen:** Add in the existing post-exercise screen to appear after the exercise is completed.

Post-Exercise Code:

**extends Node2D**

**func \_on\_cancel\_pressed():**

**get\_tree().change\_scene\_to\_file("res://Scenes/pre\_exercise\_one\_screen.tscn")**

**func \_on\_done\_pressed():**

**get\_tree().change\_scene\_to\_file("res://Scenes/main\_menu.tscn")**

**func \_on\_profile\_pressed():**

**get\_tree().change\_scene\_to\_file("res://Scenes/profile.tscn")**

**func \_on\_help\_pressed():**

**get\_tree().change\_scene\_to\_file("res://Scenes/help.tscn")**

**func \_on\_home\_pressed():**

**get\_tree().change\_scene\_to\_file("res://Scenes/home.tscn")**

### Step 7: Review and Iterate

1. **Review Feedback**: Gather feedback from users or testers who interact with the exercise. Use this feedback to identify areas for improvement or refinement.

2. **Iterate on Design**: Iterate on the exercise design and implementation based on user feedback and testing results. Make necessary adjustments to improve the exercise experience.

By following these steps, you can create a well-designed and functional exercise within your game or application, providing users with an engaging and effective learning experience.