Interactive Grid GDExtension minimal demo project

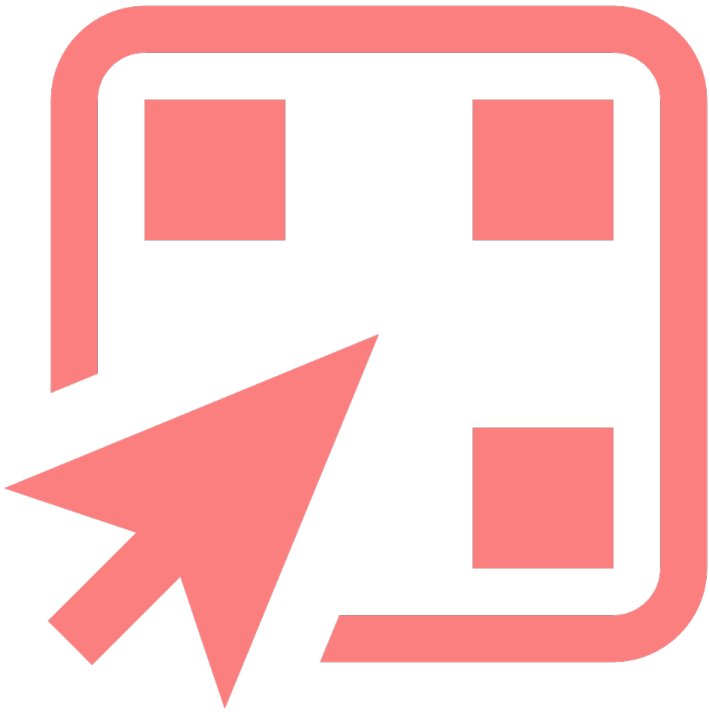
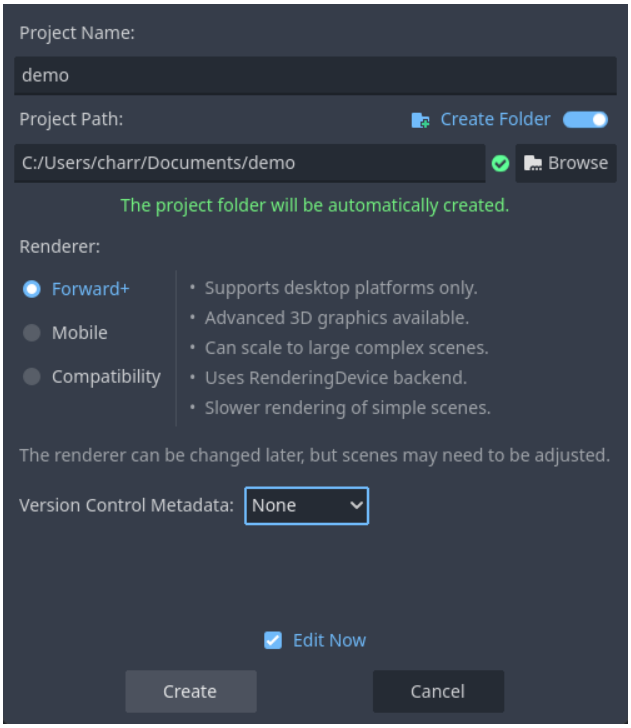


Table of Contents

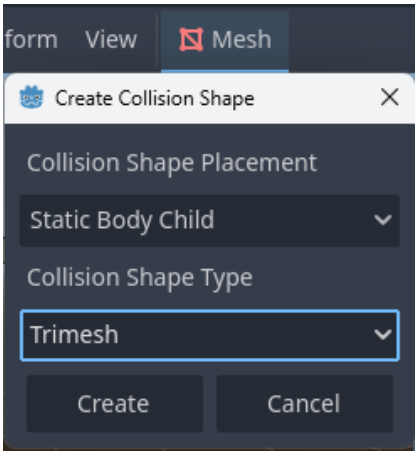
1 - Setting up the game project

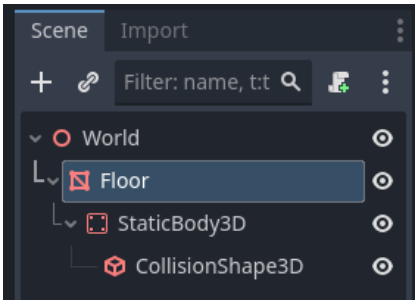
Launch Godot, create a new project, choose a location, and give it a name.



2 - Setting up the playable area

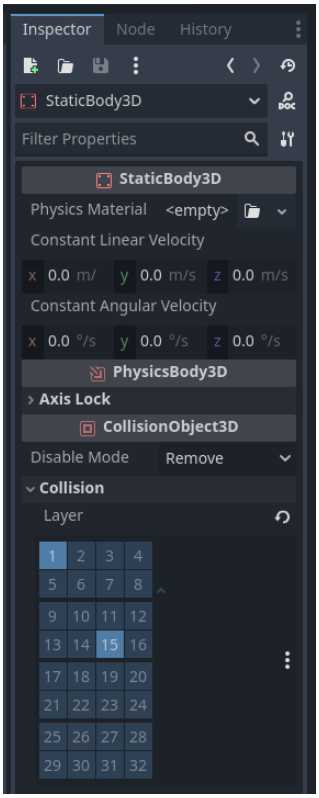
- **Create the root node.**
  - Click + and select 3D Scene.
  - Rename the root node Node3D → “World”.
- **Add the floor.**
  - Select World, Click +, choose MeshInstance3D.
  - Rename it “Floor”.
  - In the Mesh property, select PlaneMesh.
  - Set Transform → Scale to 20.0, 20.0, 20.0
- **Add collision to the floor.**
  - With Floor selected, click Mesh → Create Collision Shape.
    * Collision Shape Placement: Static Body Child.
    * Collision Shape Type: Trimesh.



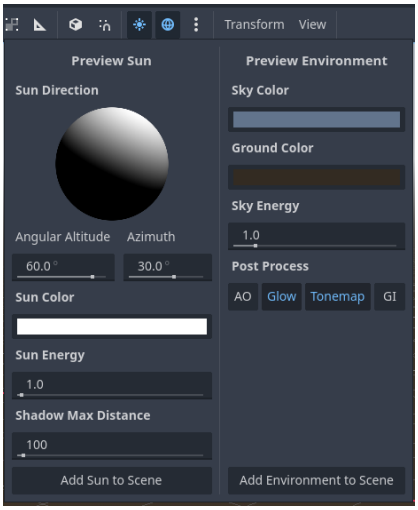


- **Set the collision layer for the floor.**
  - Select the StaticBody3D node that was created for the Floor.
  - In the Collision → Layer property, set it to 15.

> ⚠️ **Important :**
>
> Assign it to Collision Layer 15. This is important to ensure proper alignment of the grid on the floor.



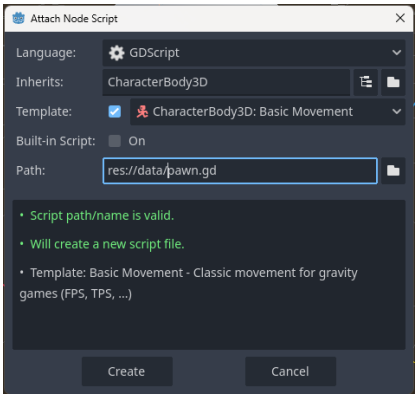
- **Add light.**
  - Add a Sun (Directional Light).
  - Add an Environment.



3 - Player scene and input actions

- **Create the player scene.**
  - New scene, Select 3D Scene.
  - Choose Node3D
  - Rename it “PawnPlayer”.
- **Add the player body.**
  - Select PawnPlayer (Node3D) node, click +, choose CharacterBody3D.

    - Rename it ““Pawn”.
    - Add a visual mesh.
        * With Pawn selected, click +, choose MeshInstance3D.
        * In the Mesh property, select CapsuleShape3D.
        * Hold the Control key and move the CapsuleShape3D up.
- **Attach a CollisionShape3D to the player.**
    - Select the Pawn (CharacterBody3D) node, click +, and add a CollisionShape3D node.
    - In the Mesh property, select CapsuleShape3D.
    - Hold the Control key and move the CapsuleShape3D up.
- **Attach a Camera3D to the player.**
    - Select the Pawn (CharacterBody3D) node, click +, and add a Camera3D node.
    - Set the Transform → Position to 6.0, 10.0, 6.0.
    - FOV 60.0
    - Set Rotation X to -45.0° and Rotation Y to 45.0°.
- **Moving the player with code.**
    - Attach a script to the player.
        * Select the Pawn (CharacterBody3D) node.
        * Click on the Attach Script icon.
        * Choose the template CharacterBody3D.gd.
        * Confirm to attach it.



pawn.gd

https://github.com/antoinecharruel/interactive_grid_gdextension/blob/master/minimal_demo/pawn.gd

```gdscript
extends CharacterBody3D


const SPEED = 5.0
const JUMP_VELOCITY = 4.5


func _physics_process(delta: float) -> void:
    # Add the gravity.
    if not is_on_floor():
        velocity += get_gravity() * delta

    # Handle jump.
    if Input.is_action_just_pressed("ui_accept") and is_on_floor():
        velocity.y = JUMP_VELOCITY

```

```
    # Get the input direction and handle the movement/deceleration.
    # As good practice, you should replace UI actions with custom gameplay actions.
    var input_dir := Input.get_vector("ui_left", "ui_right", "ui_up", "ui_down")
    var direction := (transform.basis * Vector3(input_dir.x, 0, input_dir.y)).normalized()
    if direction:
        velocity.x = direction.x * SPEED
        velocity.z = direction.z * SPEED
    else:
        velocity.x = move_toward(velocity.x, 0, SPEED)
        velocity.z = move_toward(velocity.z, 0, SPEED)

    move_and_slide()
```

- Add a Raycast3D node.
  - Select PawnPlayer.
  - Click + and add a Raycast3D node.
  - Rename it “ “RayCastFromMouse”.
- Attach the script.
  - Select RayCastFromMouse.
  - Click on the Attach Script icon.
  - Choose the script ray_cast_from_mouse.gd.
  - Fill in the script

ray_cast_from_mouse.gd

https://github.com/antoinecharruel/interactive_grid_gdextension/blob/master/minimal_demo/ray_cast_from_mouse.gd

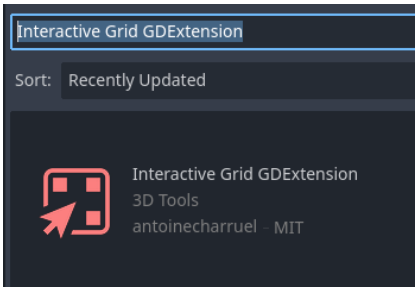
```
extends RayCast3D

@export var debug_sphere_ray_cast_: MeshInstance3D
@onready var camera_3d: Camera3D = $"../Pawn/Camera3D"

func _ready() -> void:

    # Create a sphere for raycast debugging.
    debug_sphere_ray_cast_ = MeshInstance3D.new()
    debug_sphere_ray_cast_.mesh = SphereMesh.new()
    var mat_target = StandardMaterial3D.new()
    mat_target.albedo_color = Color.GREEN
    debug_sphere_ray_cast_.material_override = mat_target
    debug_sphere_ray_cast_.scale = Vector3(0.3, 0.3, 0.3)
    add_child(debug_sphere_ray_cast_)

func _process(delta: float) -> void:

    # Position the debug sphere at the ray intersection point from the mouse.
    if(self):
        debug_sphere_ray_cast_.global_transform.origin = get_ray_intersection_position()

func get_ray_intersection_position() -> Vector3:

    var intersect_ray_position: Vector3 = Vector3.ZERO

    var mouse_pos:Vector2 = get_viewport().get_mouse_position()
    var ray_origin:Vector3 = camera_3d.project_ray_origin(mouse_pos)
    var ray_direction:Vector3 = camera_3d.project_ray_normal(mouse_pos)
    var ray_length:int = 2000

    # Position and orient the RayCast.
    self.global_position = ray_origin
    self.target_position = ray_direction * ray_length
    self.collide_with_areas = true

```

```
    self.collision_mask = 0 # Reset.
    self.set_collision_mask_value(1, true)
    self.set_collision_mask_value(15, false) # Ignore this layer.

    var debug_sphere_raycast: MeshInstance3D

    self.force_raycast_update()

    # Force an immediate RayCast update.
    if self.is_colliding():
        var collider:Node3D = self.get_collider()

        intersect_ray_position = self.get_collision_point()
        #print("[GetRayIntersectionPosition] Collision detected at: ", intersect_ray_position)
        #print("[GetRayIntersectionPosition] Collision detected with: ", collider.name)

    return intersect_ray_position
```

- **Save and add the player to the master scene.**
  - Save the player scene as pawn_player.tscn.
  - Open world.tscn, and drag pawn_player.tscn into the scene as an instance.
  - Set the Transform → Position to 0, 0, 0.

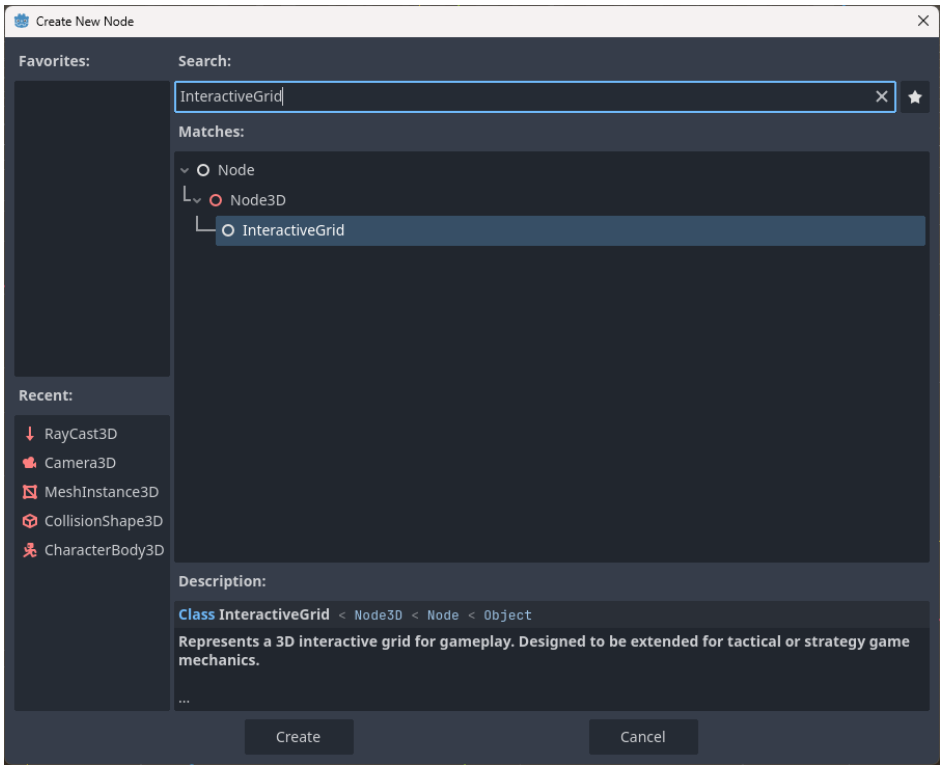
4 - Install interactive grid addon

- **In Godot, click AssetLib.**
  - Search for “Interactive Grid GDExtension”.
  - Download and install.



5 - Setup interactive grid addon

- Open the PawnPlayer scene.
- Select Pawn (CharacterBody3D), click +, and add a InteractiveGrid node.



If you see the error:

> ERROR: servers/rendering/renderer_rd/storage_rd/mesh_storage.cpp:1827 - Condition "multimesh->mesh.is_null()" is true.

Don't worry—this is normal. It simply means that the InteractiveGrid node does not yet have a multimesh assigned. You can fix it by adding a mesh in the Cell Mesh property.

- **Add a cell_mesh**
  - Select InteractiveGrid, go to the Inspector → Cell Mesh property.
  - Click on the mesh field and select BoxMesh.
  - Set Transform → 0.8, 0.1, 0.8.
- **Set grid size**
  - Rows: 15
  - Columns: 15

6 - Interactive grid scripting

- **Attach a script**
  - Select the InteractiveGrid3D node.
  - Click Attach Script.
  - Choose or create the script interactive_grid_3d.gd.
  - Fill in the script.

interactive_grid.gd

https://github.com/antoinecharruel/interactive_grid_gdextension/blob/master/minimal_demo/interactive_grid_3d.gd

```
1 extends InteractiveGrid3D
```

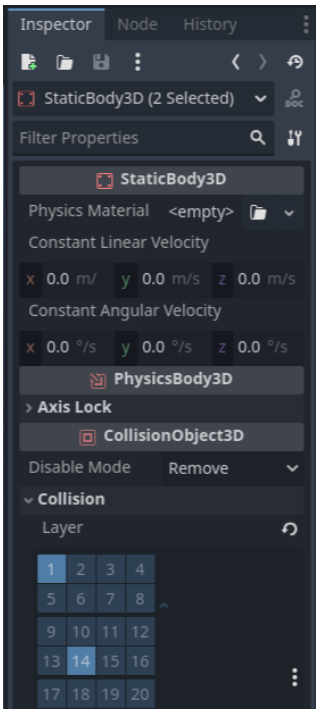
```
2
3  @onready var pawn: CharacterBody3D = $".."
4  @onready var ray_cast_from_mouse: RayCast3D = $"../../RayCastFromMouse"
5  @onready var camera_3d: Camera3D = $"../Camera3D"
6
7  func _ready() -> void:
8      pass
9
10 func _process(delta: float) -> void:
11     if pawn && ray_cast_from_mouse:
12         # Highlight the cell under the mouse.
13         if self.get_selected_cells().is_empty():
14             self.highlight_on_hover(ray_cast_from_mouse.get_ray_intersection_position())
15
16 func _input(event):
17     if event is InputEventMouseButton and event.button_index == MOUSE_BUTTON_RIGHT:
18     # ------------------------------------------------------------------
19     #  RIGHT MOUSE CLICK.
20     # ------------------------------------------------------------------
21         if event.pressed:
22             print("Right button is held down at ", event.position)
23
24             if pawn && ray_cast_from_mouse:
25                 # Makes the grid visible.
26                 self.set_visible(true)
27                 # Centers the grid.
28                 # ! Info: every time center is called, the state of the cells is reset.
29                 self.center(pawn.global_position)
30
31                 var pawn_cell_index: int = self.get_cell_index_from_global_position(pawn.
    global_position)
32
33                 # Manually set cell as unwalkable.
34                 # set_cell_walkable(75, false);
35
36                 # Check if the cell is walkable
37                 # print("Cell 75 is walkable ? : ", is_cell_walkable(75))
38
39                 # Hides distant cells.
40                 self.hide_distant_cells(pawn_cell_index, 6)
41
42                 # Manually set cell color.
43                 # var color_cell = Color(0.3, 0.4, 0.9)
44                 # self.set_cell_color(65, color_cell)
45         else:
46             print("Right button was released")
47
48
49     if event is InputEventMouseButton and event.button_index == MOUSE_BUTTON_LEFT:
50     # ------------------------------------------------------------------
51     #  LEFT MOUSE CLICK.
52     # ------------------------------------------------------------------
53         if event.pressed:
54             print("Left button is held down at ", event.position)
55
56             if pawn && ray_cast_from_mouse:
57                 # Select a cell.
58                 if self.get_selected_cells().is_empty():
59                     var selected_cell_index = get_cell_index_from_global_position(
    ray_cast_from_mouse.get_ray_intersection_position())
60                     self.select_cell(selected_cell_index)
61
62                 # Retrieve the selected cells.
63                 var selected_cells: Array = self.get_selected_cells()
64                 if selected_cells.size() > 0:
65
66                     get_cell_global_position(selected_cells[0])
67
```

```
68                      var pawn_cell_index = self.get_cell_index_from_global_position(self.
    get_grid_center_global_position())
69                      print("Pawn index: ", pawn_cell_index)
70
71                      # Retrieve the path.
72                      var path: PackedInt64Array
73                      path = self.get_path(pawn_cell_index, selected_cells[0]) # only the
    first one.
74                      #path = self.get_path(pawn_cell_index, self.get_latest_selected()) # the
     last one.
75                      print("Last selected cell:", self.get_latest_selected())
76                      print("Path:", path)
77
78                      # Highlight the path.
79                      self.highlight_path(path)
80          else:
81              print("Right button was released")
```

7 - Setup World Scene for interactive grid addon

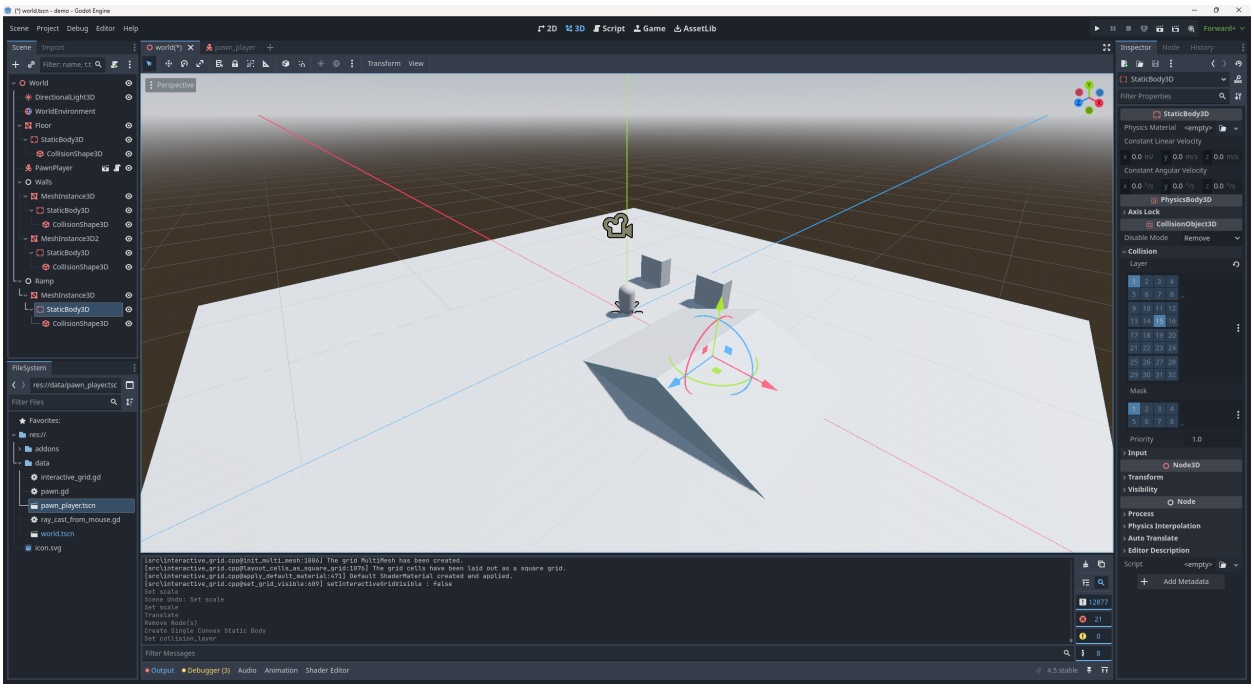
- **Create a wall.**
  - **Add a parent node for the walls.**
    * Click +, select Node3D.
    * Rename it “Walls”.
  - **Add the wall mesh.**
    * Select Walls, click +, choose CSGBox3D.
    * Set Transform → Scale to 3.0, 3.0, 0.5
- **Add collision.**
  - In the Inspector, check Use Collision.
  - Set the Collision Shape Type to Single Convex.
  - Assign the StaticBody3D to Collision Layer 14.

> ⚠️ **Important :**
>
> Assign it to Collision Layer 14. This is important to ensure that the grid correctly detects obstacles.



- **Create a ramp.**
  - **Add a parent node for ramps.**
    - Click +, select Node3D.
    - Rename it “Rampes”.
  - **Add the ramp mesh.**
    - Select Rampes, click +, choose MeshInstance3D.
    - In the Mesh property, select PrismMesh.
    - Set Transform → Scale to 10.0, 2.0, 5.0
  - **Add collision.**
    - Create Collision Shape.
      - Collision Shape Placement: Static Body Child.
      - Collision Shape Type: Trimesh.
      - Assign the StaticBody3D to Collision Layer 15 (same as the floor).

> ⚠️ **Important :**
>
> Assign it to Collision Layer 15. This is important to ensure proper alignment of the grid on the floor.



Here is what the World scene structure looks like after setting up walls, the ramp, the floor, and the interactive grid:

8 - Run the game and test the grid

Enjoy testing your interactive grid!

You should be able to move the player using the arrow keys.