Interactive Grid GDExtension minimal demo project

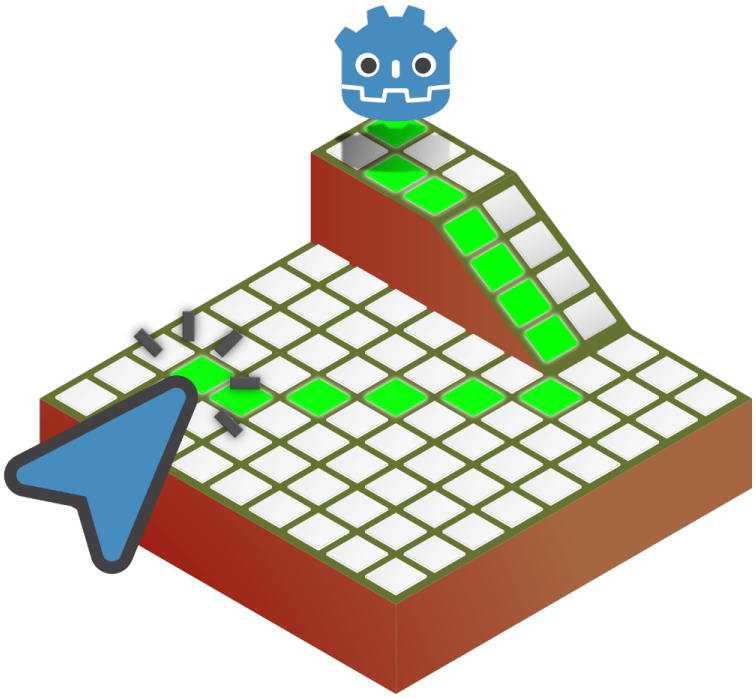
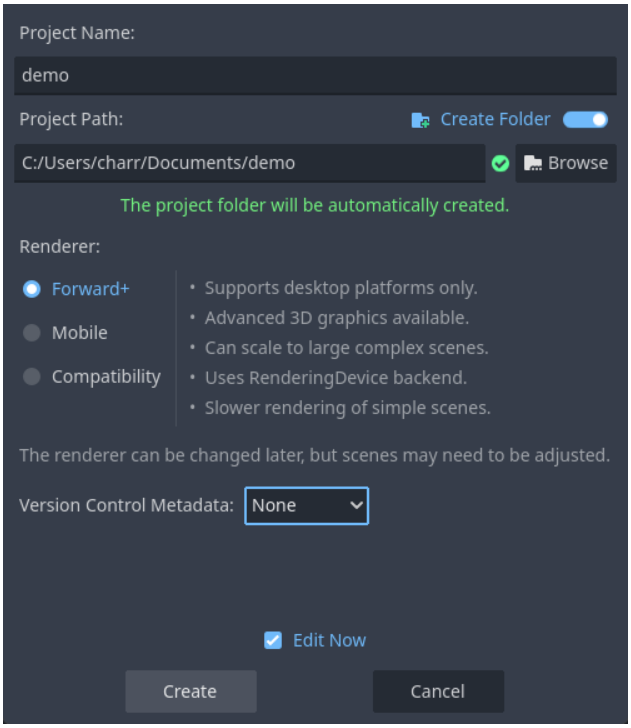


Table of Contents

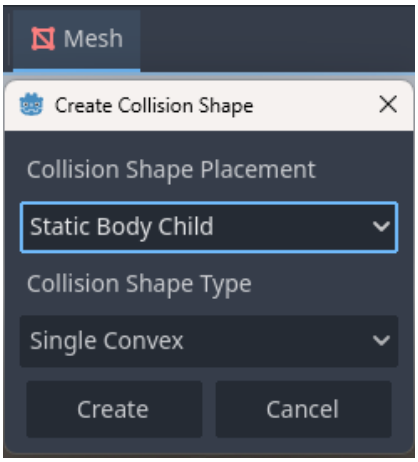
1 - Setting up the game project

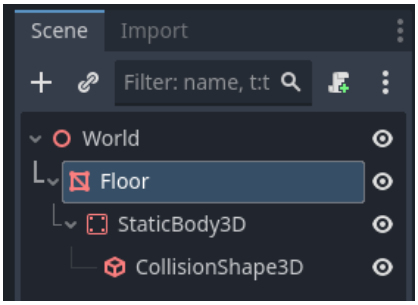
Launch Godot, create a new project, choose a location, and give it a name.



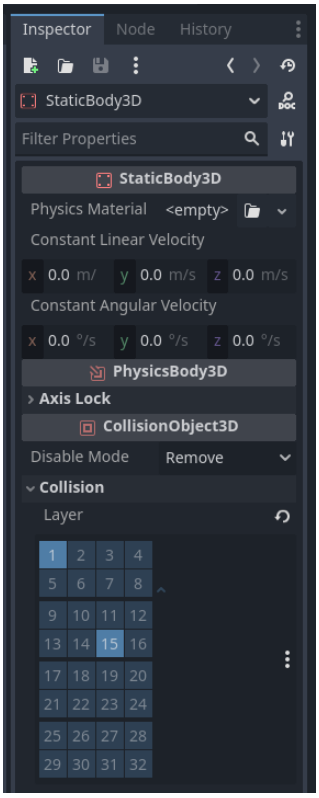
2 - Setting up the playable area

- 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, 20, 1.
- Add collision to the floor.
  - With Floor selected, click Mesh → Create Collision Shape.
  - Set Collision Shape → Type to Single Convex.

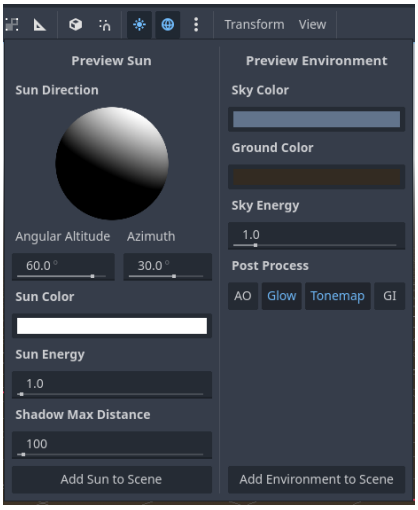




- Set the collision layer for the floor.
  - Select the StaticBody3D node that was created for the Floor.
  - In the Collision → Layer property, set it to 15. (This is important to ensure proper alignment of the grid on the floor.)

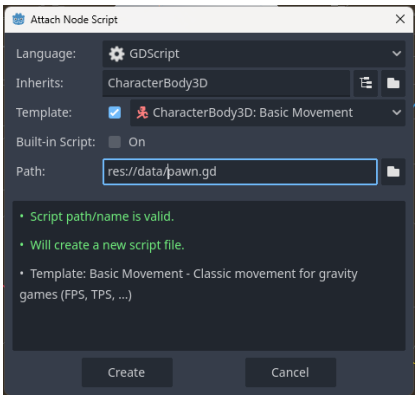


- Add light.
  - Add a Sun (Directional Light).
  - Add an Environment.



3 - Player scene and input actions

- Create the player scene.
  - Click +, 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 Collision-Shape3D 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 8, 12, 8.
  - Set Rotation X to -45° and Rotation Y to 45°.
- 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.




```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

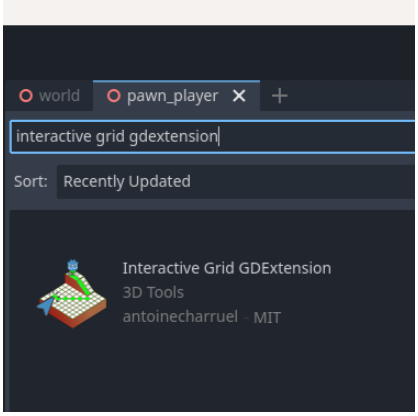
```gdscript
extends RayCast3D

@onready var ray_cast_from_mouse: RayCast3D = $"."
@export var debug_sphere_ray_cast_: MeshInstance3D
@onready var camera_3d: Camera3D = $"../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(ray_cast_from_mouse):
        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.
    ray_cast_from_mouse.global_position = ray_origin
    ray_cast_from_mouse.target_position = ray_direction * ray_length
    ray_cast_from_mouse.collide_with_areas = true

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

    var debug_sphere_raycast: MeshInstance3D

    ray_cast_from_mouse.force_raycast_update()

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

        intersect_ray_position = ray_cast_from_mouse.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 main 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.

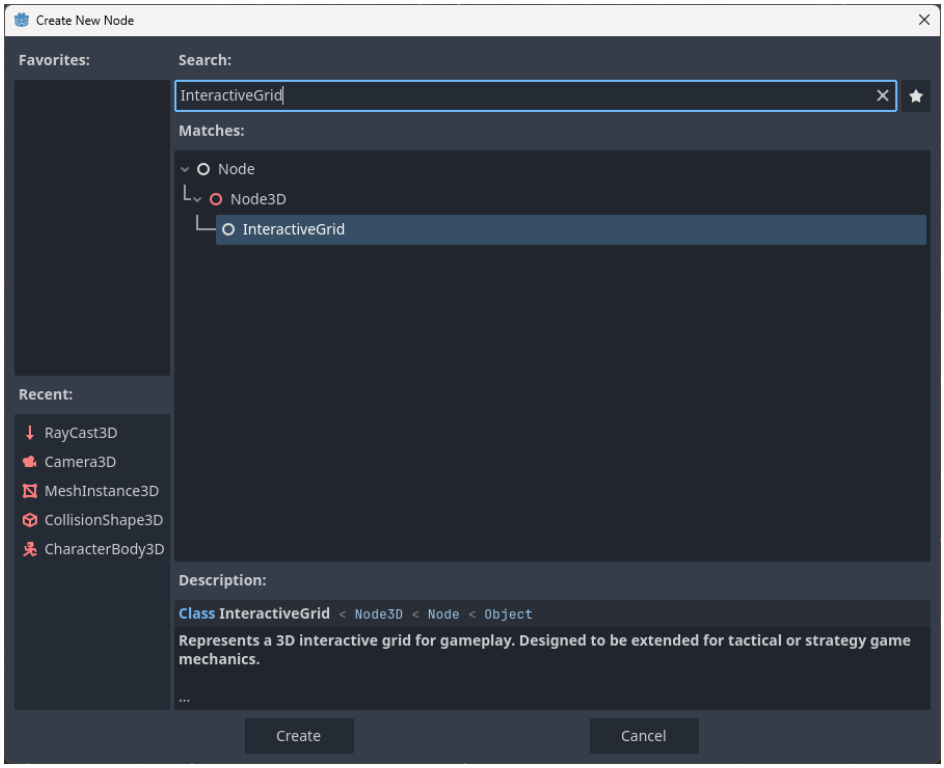
4 - Install interactive grid addons

- In Godot, click AssetLib.
  - Search for Interactive Grid GDExtension by antoinecharruel.
  - Download and install.



5 - Setup interactive grid addons

- Open the PawnPlayer scene.
- Select CharacterBody3D (Pawn), 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 the size to 0.8, 0.1, 0.8.

6 - Interactive grid scripting

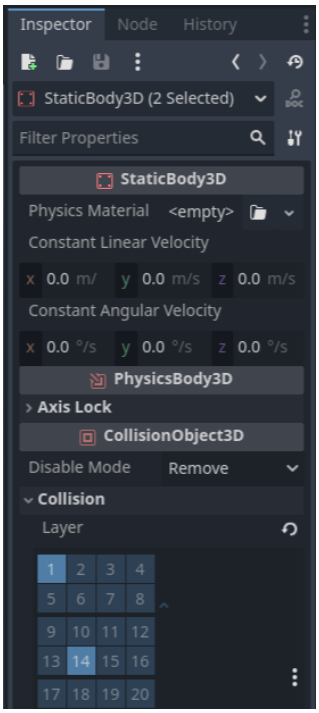
- Attach a script
    - Select the InteractiveGrid node.
    - Click Attach Script.
    - Choose or create the script interactive_grid.gd.
    - Fill in the script.

```gdscript
extends InteractiveGrid

@onready var pawn: CharacterBody3D = $".."
@onready var ray_cast_from_mouse: RayCast3D = $"../../RayCastFromMouse"
@onready var camera_3d: Camera3D = $"../Camera3D"

func _ready() -> void:
    pass

func _process(delta: float) -> void:
    if pawn && ray_cast_from_mouse:
        # Highlight the cell under the mouse.
        if self.get_selected_cells().is_empty():
            self.highlight_on_hover(ray_cast_from_mouse.get_ray_intersection_position())

func _input(event):
    if event is InputEventMouseButton and event.button_index == MOUSE_BUTTON_RIGHT:
    # ------------------------------------------------------------------
    #  RIGHT MOUSE CLICK.
    # ------------------------------------------------------------------
        if event.pressed:
            print("Right button is held down at ", event.position)

            if pawn && ray_cast_from_mouse:
                # Makes the grid visible.
                self.set_visible(true)
                # Centers the grid.
                # ! Info: every time center is called, the state of the cells is reset.
                self.center(pawn.global_position)

                var index_cell_pawn: int = self.get_cell_index_from_global_position(pawn.global_position)

                # Manually set cell as unwalkable.
                # set_cell_walkable(75, false);

                # Check if the cell is walkable
                # print("Cell 75 is walkable ? : ", is_cell_walkable(75))

                # Hides distant cells.
                self.hide_distant_cells(index_cell_pawn, 6)
                self.compute_inaccessible_cells(index_cell_pawn)

                # Manually set cell color.
                # var color_cell = Color(0.3, 0.4, 0.9)
                # self.set_cell_color(65, color_cell)
        else:
            print("Right button was released")


    if event is InputEventMouseButton and event.button_index == MOUSE_BUTTON_LEFT:
    # ------------------------------------------------------------------
    #  LEFT MOUSE CLICK.
    # ------------------------------------------------------------------
        if event.pressed:
            print("Left button is held down at ", event.position)

```

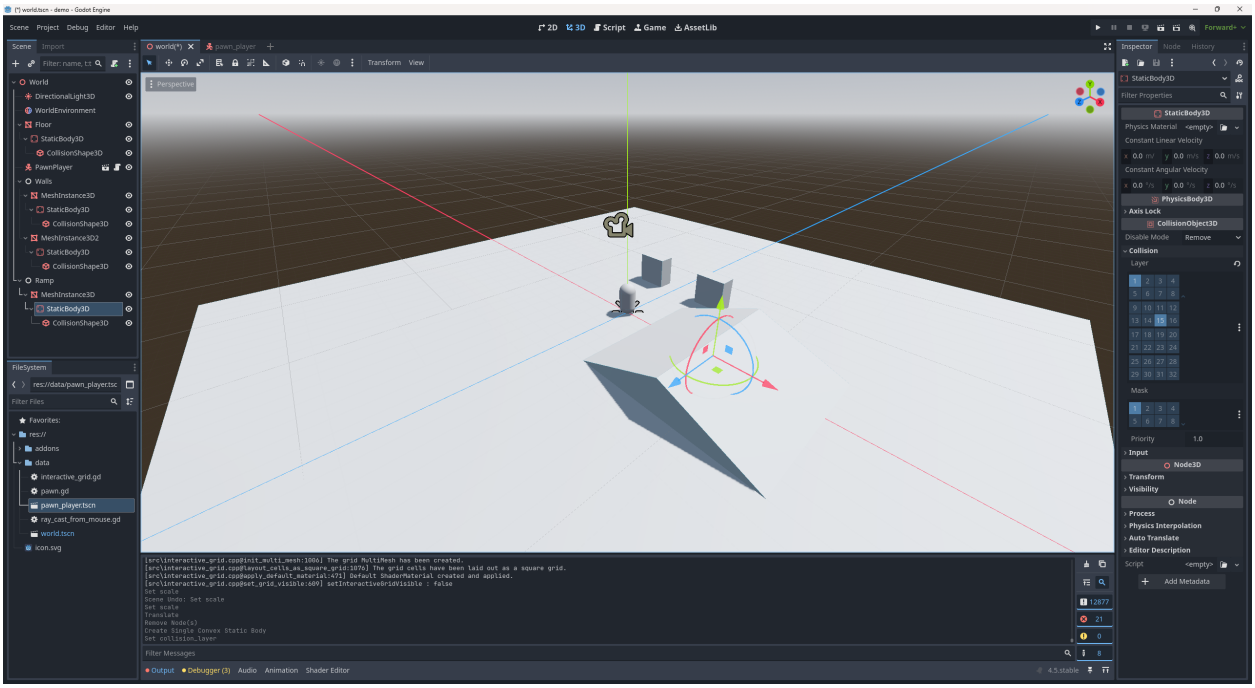
```
            if pawn && ray_cast_from_mouse:
                # Select a cell.
                if self.get_selected_cells().is_empty():
                    self.select_cell(ray_cast_from_mouse.get_ray_intersection_position())

                # Retrieve the selected cells.
                var selected_cells: Array = self.get_selected_cells()
                if selected_cells.size() > 0:

                    get_cell_golbal_position(selected_cells[0])

                    var index_cell_pawn = self.get_cell_index_from_global_position(self.
    get_grid_center_global_position())
                    print("Pawn index: ", index_cell_pawn)

                    # Retrieve the path.
                    var path: PackedInt64Array
                    path = self.get_path(index_cell_pawn, selected_cells[0]) # only the
    first one.
                    #path = self.get_path(index_cell_pawn, self.get_latest_selected()) # the
     last one.
                    print("Last selected cell:", self.get_latest_selected())
                    print("Path:", path)

                    # Highlight the path.
                    self.highlight_path(path)
        else:
            print("Right button was released")
```

7 - Setup World Scene for interactive grid addons

- Create a wall.
  - Add a parent node for the walls.
    * Click +, select Node3D.
    * Rename it “Walls”.
  - Add the wall mesh.
    * Select Walls, click +, choose MeshInstance3D.
    * In the Mesh property, select CubeMesh.
    * 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 wall to Collision Layer 14.



- 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, 3.
  - Add collision.
    * In the Inspector, check Use Collision.
    * Set the Collision Shape Type to Single Convex.
    * Assign it to Collision Layer 15 (same as the floor).



Here is what the World scene structure looks like after setting up walls, ramps, 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.