Contents

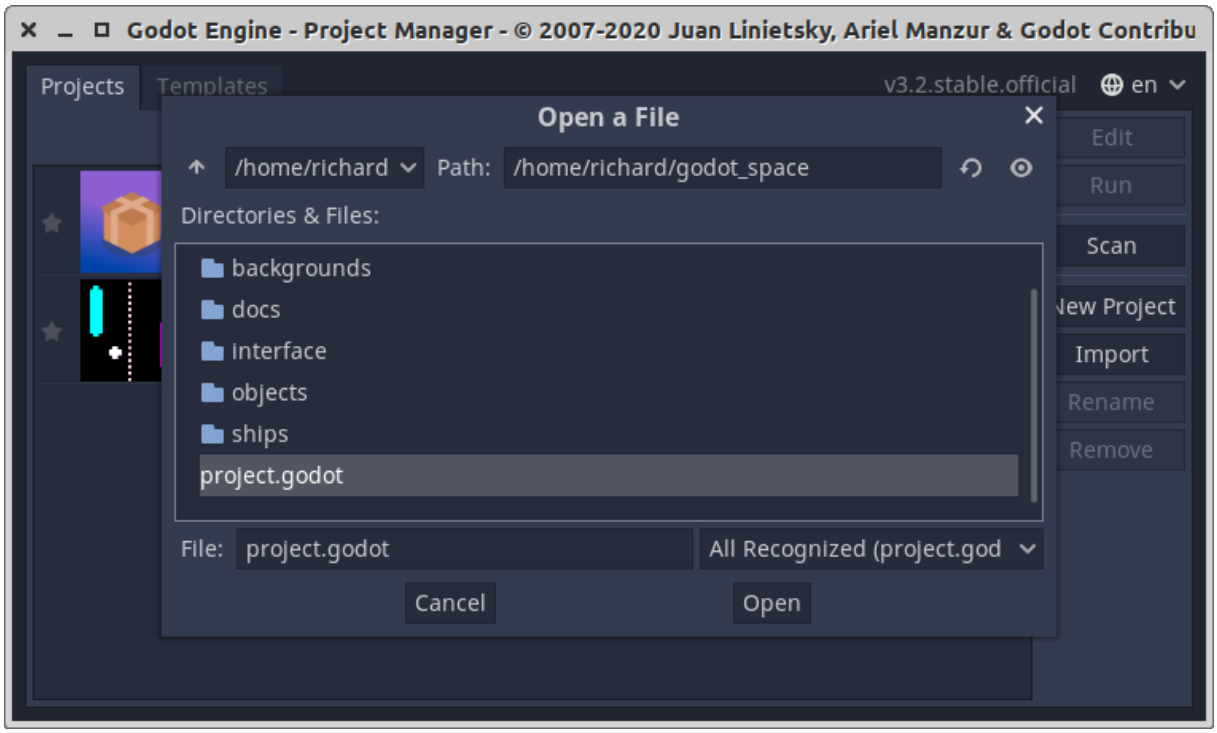
1 Space Shooter Tutorial

1.1 Import project

Download the starter project: https://electronstudio.github.io/godot_space/godot_space1.zip.

Unzip it. Open Godot. `Import` the `project.godot` file.

Helpful video: https://www.youtube.com/watch?v=ovt4hS9iLYY



Run the game.

You should have a spaceship sprite that can turn left and right.

The project contains many spaceship images you can use. There is also lighting and a HUD.

1.2 Player movement

This code is in `player.gd`. **You do not need to type this, it has already been typed for you!** Make sure you understand it before you continue.

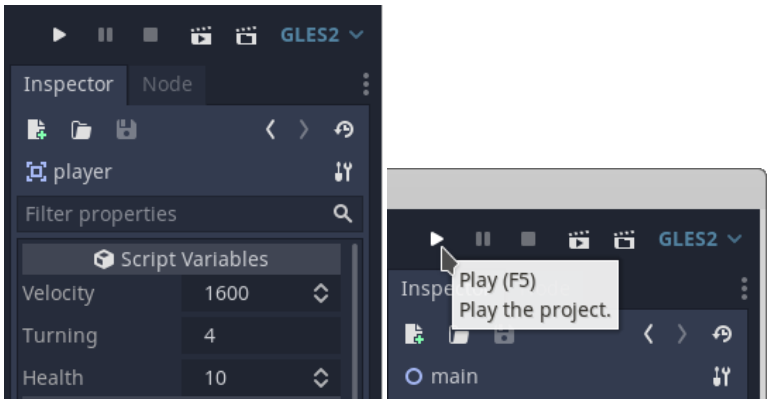
Which line moves the player?

What is `delta`?

How many times per second does the code run?

```
1  extends Area2D
2
3  export var velocity = 0
4  export var turning = 4.0
5  export var health = 10
6
7  var Bullet = preload("res://player_bullet.tscn")
8  var score = 0
9
10 func _process(delta):
11     if Input.is_action_pressed("turn_left"):
12         rotation -= turning * delta
13     if Input.is_action_pressed("turn_right"):
14         rotation += turning * delta
15     if Input.is_action_just_pressed("fire"):
16         Bullet.instance().init(self, 4000)
17
18     position += Vector2.RIGHT.rotated(rotation) * velocity * delta
```

1.3 Velocity

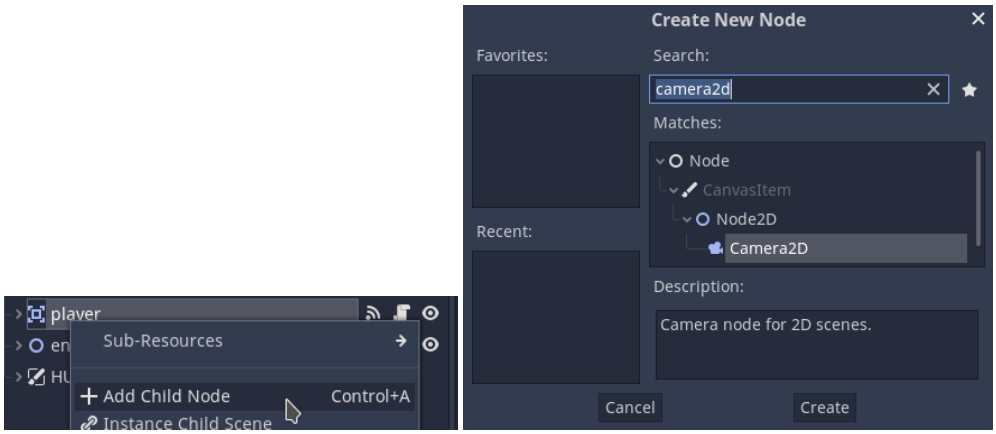


When a script `exports` a variable we can change the value using the Inspector without editing the script.

Change the `velocity` of the `player` to 1600 in the node inspector. Run the game.

1.4 Camera

We need a camera to track the player. Add a `Camera2D` node to the `player` node.



In the node inspector set:

- Current: On
- Zoom x: 6
- Zoom y: 6
- Drag Margin:
  - Left: 0
  - Right: 0
  - Top: 0
  - Bottom: 0

What happens if you change these values?

1.5 Background

We would like to add another layer to the scrolling background.

1. Right-click on the `ParallaxBackground` node and select `Add Child Node`. Choose the type `ParallaxLayer` and press `Create`.
2. Click on the `ParallaxLayer2`.
3. In the Inspector, set:
    - Motion: Mirroring x: 15360
    - Motion: Mirroring y: 15360
4. Find `stars_big_1024.png` in the FileSystem. (Bottom left of screen, inside `backgrounds` folder.)
5. Drag into it the scene in the centre of the screen.
6. Click on the `stars_big_1024` sprite node.

7. In the Inspector, under `Node2D` `Transform` set:

    - Position x: 7680
    - Position y: 7680
    - Scale x: 15
    - Scale y: 15

Run the game to verify it works.

1.6 Particle effect

The player already has a `CPUParticles2D` node made for you. Click on it. In the Inspector, set:

- Emitting: On
- Amount: 50

Experiment with changing these settings. *What do they do?*

- Lifetime
- Spread
- Gravity
- Velocity
- Color
- Anything else you like

1.7 Make enemy move

Open the `enemy.tscn` scene file by double clicking it.

Right click on the `enemy` node and attach a script. Press `Create`. Replace the contents of the script with:

```
extends Area2D

export var VELOCITY = 1000.0
export var TURNING = 0.7
export var FIRE_RATE = 0.01

var Bullet = preload("res://enemy_bullet.tscn")
onready var player = get_node("/root/main/player")

func _process(delta):
    var d = player.position.angle_to_point(position)
    rotation = Util.rotate_toward(rotation, d, TURNING*delta)
    position += Vector2.RIGHT.rotated(rotation) * VELOCITY * delta

    if position.distance_to(player.position) > 7000:
        queue_free()

    if randf()<FIRE_RATE:
        Bullet.instance().init(self, 3000)
```

Run the game again.

1.8 Add more enemies

NOTE: The `Light2D` node under `HUD` covers the whole screen with an invisible object (the light) and that makes it difficult to select other sprites because you always accidently select the light. I suggest you click the eye icon next to `Light2D` to hide it. But don't forget to unhide it once you have finished positioning your sprites! 

1. Go back to the `main.tscn` scene (should be open as a tab).
2. Duplicate (ctrl-D) the enemy node a few times. Drag the duplicates to new positions in the main window.
3. Try changing the exported variables in the Inspector so they move at different velocities. *Can you make a more deadly enemy this way?*
4. Also try changing `Node2D` `Transform` `Rotation`.
5. Test the game again.

1.9 Make bullets move

Try pressing space to shoot bullets. *What happens? Why?*

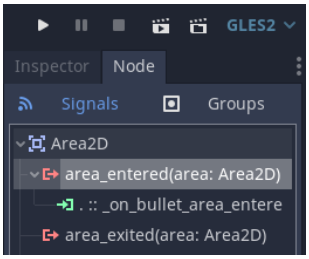
We already have scene files for the bullets: `player_bullet.tscn` and `enemy_bullet.tscn`.

They are both attached to the same script file, `bullet.gd`. Double click the file to edit the script and add this function:

```
func _process(delta):
    position += Vector2.RIGHT.rotated(rotation) * velocity * delta
    if position.distance_to(player.position) > 5000:
        queue_free()
```

Test the game again. *Why are we testing the distance from the player?*

1.10 Make bullets collide

1. Open the `player_bullet.tscn` scene file by double clicking it.
2. Click `bullet` node.
3. In the inspector click `Node` at the top to view the signals. 

4. Double click the `area_entered` signal.
5. Select `bullet` from the list of nodes to connect to.
6. Click `connect`.
7. Godot will create an empty function for you. Replace it with this:

```
func _on_bullet_area_entered(area):
    queue_free()
```

7. Repeat steps 1-5 for `enemy_bullet.tscn` scene. (No need to edit the script and add the function again, since it's the same script and you already did it.)
8. Test the game again. *What does the `area_entered` signal do?*

1.11 Make enemy collide

1. Open the `enemy.tscn` scene file by double clicking it.
2. Click `enemy` node.
3. In the inspector click `Node` at the top to view the signals.
4. Double click the `area_entered` signal.
5. Select `enemy` from the list of nodes to connect to.
6. Click `connect`.
7. Godot will create an empty function for you. Replace it with this:

```
func _on_enemy_area_entered(area):
    $explosion.play()
    $AnimationPlayer.play("fade")
    $CollisionPolygon2D.queue_free()
    $CPUParticles2D.emitting = true
    $CPUParticles2D.show()
    player.score += 1
    get_node("/root/main/HUD/score").text = str(player.score)
    yield(get_tree().create_timer(1.0), "timeout")
    queue_free()
```

Test the game again.

1.12 Make player collide

1. Go back to the `main.tscn` scene (should be open as a tab).
2. Click `player` node.
3. In the Inspector click `Node` at the top to view the signals.
4. Double click the `area_entered` signal.
5. Select `player` from the list of nodes to connect to.
6. Click `connect`.
7. Godot will create an empty function for you. Replace it with this:

```
func _on_player_area_entered(area):
    health -= 1
    get_node("../HUD/health").value = health
    if health <= 0:
        get_tree().reload_current_scene()
    $crash_sound.play()
    modulate = Color(1000, 0, 0, 255)
    yield(get_tree().create_timer(1.0), "timeout")
    modulate = Color(1, 1, 1, 255)
```

Test the game again. _What does the `modulate` value do?

1.13 Gamepad controls (optional)

This is only required if you want to play to play with a gamepad.

Open `player.gd` script. **Delete** the `gamepad(delta)` function and **replace** it with this:

```
var virtual_stick_direction = Vector2.ZERO

func gamepad(delta):
    var input = Vector2(Input.get_joy_axis(0, 0), Input.get_joy_axis(0,
        1)) + virtual_stick_direction
    if input.length() > 0.2:
        var direction = input.angle()
        rotation = Util.rotate_toward(rotation, direction, turning *
            delta)
```

1.14 Touch screen controls (optional)

This requires the gamepad code above to have been added. If you have a mobile phone or tablet this will allow you to play on the touch screen.

Add to `player.gd` script:

```
var virtual_stick_origin = Vector2.ZERO

func _input(event):
    if event is InputEventScreenTouch:
        if event.position.x < get_viewport().size.x/2.0:
            if event.pressed:
                virtual_stick_origin = event.position
        else:
            if event.pressed:
                Input.action_press("fire")
            else:
                Input.action_release("fire")
    elif event is InputEventScreenDrag and event.position.x <
       get_viewport().size.x/2.0:
        virtual_stick_direction =  (event.position -
           virtual_stick_origin).normalized()
```

You can use your mouse (hold down left button) to test the touch controls on your computer.

1.15 More types of enemies (advanced)

We could make a new enemy scene from scratch, but it's probably easier to duplicate the existing one.

1. Duplicate (ctrl-D) `enemy.tscn` to create `cool_enemy.tscn`.
2. Double click `cool_enemy.tscn` to open it.
3. Click `2D` at the top of the screen if you are in the script view.
4. Right-click on the `destroyer` sprite node and delete it.
5. Right-click on the `CollisionPolygon2D` sprite node and delete it.
6. In the FileSystem, find a ship png image, e.g. `ships/Faction10-Spaceships/cargoship.png` and drag it into the scene at the (0,0) point.
7. Click on the node you just created (`cargoship`). Click on the Inspector if it's not visible. You can see the sprite has a texture but does not have a normal map. Without a normal map lighting will not work.
8. Find the appropriate normal map png (e.g. `cargoship_normal.png`) and drag it on to the `Normal Map` space in the Inspector.
9. In the middle-top of the screen, click the word `Sprite`. A menu pops up.
10. Click `Create CollisionPolygon2D Sibling`. A preview opens.

11. Click `Create CollisionPolygon2d`.
12. Save your finished scene (ctrl-S).
13. Go back to the `main.tscn` scene.
14. Right-click on the `enemies` node.
15. Select `instance child scene`. A window opens.
16. Double click `cool_enemy.tscn` in the window.
17. Position your enemy by dragging it in the main window. You could also drag the `cool_enemy.tscn` directly from the FileSystem into the main window.
18. *Can you make more types of enemy, e.g. a big space station?*

1.16 Enemy spawner (advanced)

Rather than creating enemies ourselves, we will create an invisible node to spawn them for us automatically.

1. Add a child node to `enemies` node. The type should be `Timer`.
2. Right-click the `Timer` node and rename it to `enemy_spawner`.
3. This node will "timeout" once every second. You can change the `Wait Time` in the Inspector if you want.
4. In the Inspector set `autostart` to `On`.
5. Right-click the `enemy_spawner` and attach a script. Click `create`. Replace the contents of the script `enemey_spawner.gd` with this code:

```
1  extends Timer
2
3  export (PackedScene) var  Enemy
4  export var MAX_ENEMIES = 10
5  export var MAX_SCORE = 999999
6  export var MIN_SCORE = 0
7  onready var player = get_node("/root/main/player")
8
9  func _on_enemy_spawner_timeout():
10     if get_child_count() < MAX_ENEMIES && player.score <= MAX_SCORE &&
       player.score >= MIN_SCORE:
11         var enemy = Enemy.instance()
12         add_child(enemy)
```

6. Save (ctrl-S).

7. Click `Node` by the Inspector to view Signals.
8. Double-lick the `timeout()` signal.
9. Select `enemy_spawner` from the list and click `connect`.
10. Click on the `enemy_spawner` node.
11. Click on the Inspector. See where is says `Enemy [empty]`. This is the kind of enemy it will spawn.
12. Find `enemy.tscn` in the file system and drag it onto the `[empty]`.
13. You can duplicate this spawner to create more spawners and change the exported variables in the Inspector, including which kind of enemy is spawned.

1.17 Enemy randomize

Add this to `enemy.gd` to randomize the position of the enemies when they spawn.

```
1 func _ready():
2     position = player.position + Vector2.RIGHT.rotated(rand_range(0, PI
          *2)) * 5000
3     rotation = player.position.angle_to_point(position)
```

1.18 Charge laser

Right-click on the `player` node and select `instance child scene`. A window opens.

In the window double-click `laser.tscn`.

Right-click on `laser` node and `Attach Script`. Click `Create`. Replace the contents of the script with this:

```
extends Area2D

var charge = 0.0

enum  {CHARGING, DISCHARGING}
var laser = DISCHARGING

func _process(delta):
    if Input.is_action_just_pressed("fire") and charge < 0.01:
        laser = CHARGING
    if Input.is_action_just_released("fire"):
        laser = DISCHARGING
    if laser == CHARGING:
        charge += delta
    elif laser == DISCHARGING:
        charge -= delta
        if charge > 0.3:
            show()
            monitorable = true
        else:
            hide()
            monitorable = false
    charge = clamp(charge, 0.0, 3.0)
    get_node("/root/main/HUD/charge").value = charge
```

Note this laser only has two states and so could have been done more simply using a boolean, but I wanted to demonstrate use of `enum` because in future you might have more than two states.

1.19 Title screen (challenge)

Can you add a title screen to the game? Here are some hints.

Create a new scene called `title.tscn`. Put your title screen text and graphics here.

Attach this script to the root node to start the game when player presses space.

```
extends Node2D

func _process(delta):
    if Input.is_action_just_pressed("fire"):
        get_tree().change_scene("res://main.tscn")
```

When the player dies, execute this code to switch to the title screen:

```
get_tree().change_scene("res://title.tscn")
```