

# ASTEROIDS

## GODDOT 4 - 2D PROJECT

# PREPARATION

- Install Godot
- Setup Project
- Assets

# INSTALL

## Install Godot 4

Then under preference:

- Text Editor
- Completion
- Add Type Hints

# PROJECT

Create a new project - choose compatibility  
renderer

# PROJECT SETTINGS

## Display - Window

- Viewport Width: 1280
- Viewport Height: 720
- Stretch - Mode: viewport
- Stretch - Aspect: keep

# PROJECT SETTINGS

## Rendering

- **Default Texture Filter - Nearest**
- **Default Clear Color - Black**

# PROJECT SETTINGS

## Layer Names - 2D Physics

- **1: Player**
- **2: Asteroid**
- **3: Bullet**

# PROVIDED FILES

- Sounds were generated using **chiptone**
- Font from **fontspace**
- Artwork from **kenney.nl**



SO - LET'S GET STARTED

# PLAYER SCENE

**We'll need a Player scene (Area2D)**

**It will need a sprite child node.**

**It will also need a collision child node.**

# PLAYER SPRITE

- Add Sprite2D child node
- Set texture to the provided tilesheet
- Enable region
- Set region rect to x: 64, y: 128, w: 64, h: 64

But - the center (will be used for rotation) is too far forward.

We want it about the widest point of the body.

- Set the sprite transform to y: -10

# PLAYER COLLISION

- Add CollisionPolygon2D child node
- Draw round main body of ship
- On the player node - set collision layer for player to layer 1, no mask

# PLAYER ROTATION

Let's get it rotating

Add script to the player node

```
extends Area2D
```

```
@export var rotation_max: = 3
```

```
func _process(delta: float) -> void:
```

```
    var rotate_input = Input.get_axis("ui_left", "ui_right")
```

```
    rotation += rotation_max * rotate_input * delta
```

```
    rotation = fmod(rotation, TAU)
```

# WORLD SCENE

It's hard to see the player in the corner.

Let's put it middle of screen.

This means we'll need a world.



- Create world scene (Node2d)
- Add player node as child
- Add script to world

```
extends Node2D
```

```
@onready var screen_size: Vector2i = get_viewport().size
```

```
@onready var player: = $Player
```

```
func _ready() -> void:  
    # Put player in middle of screen  
    player.position = screen_size / 2
```

# PLAYER MOVEMENT

OK - let's get it to move too

Extend the player script

```
@export var speed_max = 200

func _process(delta: float) -> void:
    ...

    var acceleration = Input.get_action_strength("ui_up")

    if acceleration > 0:
        var y = -speed_max * cos(rotation)
        var x = speed_max * sin(rotation)

        position += Vector2(x, y) * delta
```

WRAP AROUND

Hmm

It goes off screen

Let's get wraparound

Extend the player script

# We could do the math

```
if position.x > screen_size.x:
    position.x = 0
if position.y > screen_size.y:
    position.y = 0
if position.x < 0:
    position.x = screen_size.x
if position.y < 0:
    position.y = screen_size.y
```

## But - we can use `wrapf` to make that simpler

```
@onready var screen_size: Vector2i = get_viewport().size

func _process(delta: float) -> void:
    ...
    screen_wrap()

func screen_wrap() -> void:
    position.x = wrapf(position.x, 0, screen_size.x)
    position.y = wrapf(position.y, 0, screen_size.y)
```

# SHOOTING - WE'LL NEED AMMO

**We need to be able to shoot stuff.**

**Add a bullet scene (Area2D)**

**Add a Sprite2D and CollisionShape2D**

# BULLET SPRITE

- load same texture as before
- enable region
- select region x:448 y:192 w:64 h:64



# BULLET COLLISION

Add a circular collision shape

Size it just inside the bullet

On the bullet node - set it on collision layer 3  
(bullet) and mask layer 2 (asteroid)

# BULLET MOVEMENT

**To make the bullet move - add a script so that we can both set the initial direction and also move it.**

```
var direction: = Vector2.ZERO
var moved: = 0
@export var max_move: = 240
@export var speed_max: = 210

func set_direction(rads: float) -> void:
    rotation = rads
    direction = Vector2.UP.rotated(rotation)

func _process(delta: float) -> void:
    moved += 1

    if moved > max_move:
        queue_free()

    position += direction * delta * speed_max

    # Make the bullet spin too
    rotation += 10 * delta
    rotation = fmod(rotation, TAU)
```

# SHOOT!

We actually need to be able to shoot it from the player.

To do this - we'll add a point on the player for the bullets to come from.

Then - each time the trigger is pulled - we'll create a new bullet instance and place it there - aligned with the ship.

- Add child Node2D
- Move it with transform - about y: -32 to place at front of ship
- We also want to refer to it in the script - so rename it to Tip

Now - we will need to load the bullet for every shot.

That's not efficient - so - we'll preload the scene so that we can use it multiple times.

```
const bullet = preload("res://Bullet/Bullet.tscn")
```

Then in process - we want to detect the shot -  
when this happens

- create a new bullet instance
- place it at the tip of the ship
- rotate it to match the ship
- add it to the world scene
- we'll also use the world scene to play the sound

we use the world scene for display and sound so  
that they continue even if the player dies

```
if Input.is_action_just_pressed("ui_select"):
    var bullet_instance = bullet.instantiate()
    bullet_instance.global_position = tip.global_position
    bullet_instance.set_direction(rotation)
    get_parent().add_child(bullet_instance)
    get_parent().bullet_fired()
```



Now - this won't work until we add `bullet_fired` to the world scene.

To do this - in the world node:

- add an `AudioStreamPlayer` (not 2d or 3d) as a child node
- Call it `BulletSoundPlayer`
- Add a stream (quick load - choose `shoot.wav`)

## Then in the world script:

```
@onready var bulletSound: = $BulletSoundPlayer
```

```
func bullet_fired() -> void:  
    bulletSound.play()
```

If you want bullets to wrap then you can also add the same wrap function to the bullet script.

I won't but it is possible.

# TARGETS!

**We've nothing to shoot at**

**Let's add some asteroids**

# ASTEROID SCENE

- Add an Area2D scene for Asteroid
- Add a Sprite2D
- Add a CollisionShape2D
- Sprite region: 0 256 64 64
- Collision - circle
- Layers - place on 2 (asteroid) and mask on 1

# ASTEROID MOVEMENT

**Add a script to the asteroid scene.**

**When ready - we want the asteroid to spin randomly and to move in a random direction.**

```
@export var rotation_max: = 3.0
@export var speed_max: = 100

var direction: = Vector2.ZERO
var rotation_speed: = 0

func _ready() -> void:
    randomize()

    rotation = randf() * TAU

    direction = Vector2(build_random_direction(), build_random_direction())

    rotation_speed = (2 * rotation_max * randf()) - rotation_max

func _process(delta: float) -> void:
    position += direction * delta
    rotation += rotation_speed * delta

func build_random_direction() -> float:
    return (1.0 - randf() * 2) * speed_max * (1.0 + randf())
```

# ADDING ASTEROIDS

**We need to add asteroids to the world.**

**We will also need to keep track of how many  
there are**

**Expand on the world script**

```
const asteroid = preload("res://Asteroid/Asteroid.tscn")

@export var start_count = 7

var asteroid_count: = 0

func _ready() -> void:
    ...

    for _i in range(start_count):
        build_asteroid()

func build_asteroid() -> void:
    var asteroid_instance = asteroid.instantiate()
    asteroid_count += 1

    add_child(asteroid_instance)

    asteroid_instance.global_position = Vector2(screen_size.x * randf(), screen_size.y * randf())
```



ADD WRAP AROUND HERE TOO

**We'll use the same code as we did for player**

# HITTING STUFF

**We need to work on collisions:**

- **Player can be hit by asteroids**
- **Asteroid can be hit by bullets**

**We've already set up the collision layers for this -  
but we need to actually detect the collisions and  
do something when they happen.**

# DYING

The asteroid needs to react if it hits the player.

Select the asteroid node then in the node menu - connect the "area entered" event.

We'll send a signal when this happens that will trigger changes in the world.

Expand the asteroid script

```
signal kill
```

```
func _on_area_entered(area: Area2D) -> void:  
    emit_signal("kill")
```

## Then set up the World

- Add a new `AudioStreamPlayer`
- call it `KillSoundPlayer`
- add the `die.wav` stream

Expand the world script

```
@onready var killSound: = $KillSoundPlayer

var alive: = true

func kill_player():
    killSound.play()
    player.queue_free()
    alive = false

func build_asteroid() -> void:
    ...
    asteroid_instance.kill.connect(kill_player)
```

# HITTING ASTEROIDS

The bullet needs to react if it hits an asteroid.

Select the bullet node then in the node menu - connect the "area entered" event.

This time we'll also have to send which asteroid (area) was hit when we signal.

Expand the bullet script

signal hit

```
func _on_area_entered(area: Area2D) -> void:  
    emit_signal("hit", area)
```



Only the player knows about the bullet - so this needs to pass the event on.

In the player script:

```
func _process(delta: float) -> void:
    ...

    if Input.is_action_just_pressed("ui_select"):
        ...
        bullet_instance.hit.connect(bullet_hit)

func bullet_hit(area: Area2D) -> void:
    get_parent().hit(area)
```

## Then set up the World

- Add a new `AudioStreamPlayer`
- call it `HitSoundPlayer`
- add the `boom.wav` stream

Expand the world script

```
@onready var hitSound: = $HitSoundPlayer
```

```
func hit(area):  
    asteroid_count -= 1  
    hitSound.play()  
    area.queue_free()
```

# END GAME

We have one end game already - the player dies.  
We need to support the other option - the player wins.

For now - we'll just remove the player.

```
func hit(area):  
    ...  
  
    if asteroid_count <= 0:  
        player.queue_free()
```

# RESTART

After end of game - we need to be able to restart.

Simplest is just to reload the app

```
func _process(delta: float) -> void:  
    if (asteroid_count <= 0 or not alive) and Input.is_action_pressed("ui_accept"):  
        get_tree().reload_current_scene()
```

# POSSIBLE IMPROVEMENTS?

- Scoring
- Start/Died/Won screens
- Engine effects (sound, particles)
- Different asteroids
- Asteroid breakup to smaller rocks
- ...