Asteroids

Godot 4 - 2D project

Preparation

- Install Godot 4.x
- Setup Project
- Assets

Install

Install Godot 4

Project

Open exercise 1

This is a new project using the compatibility renderer

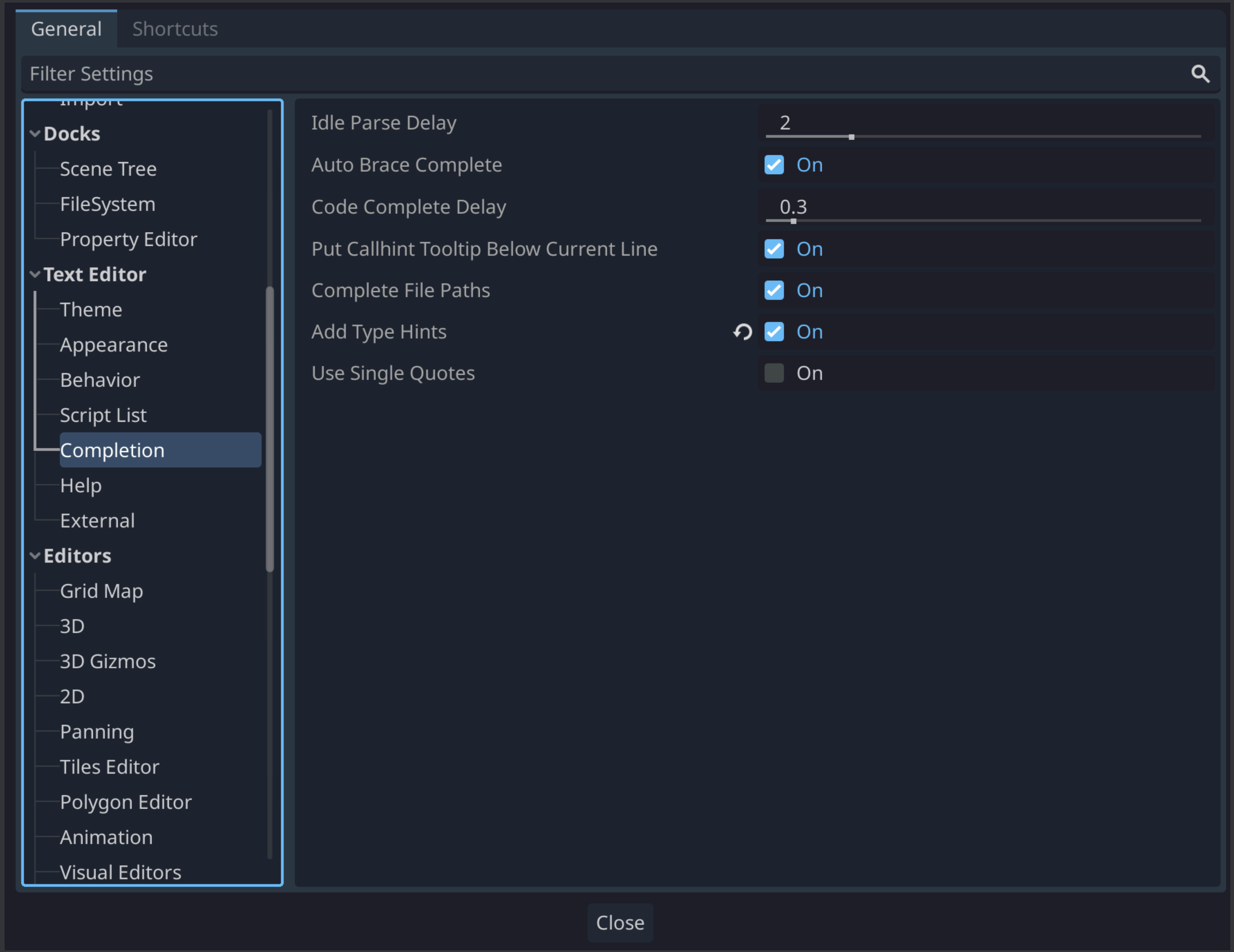
Exercise 1

We want to set an editor setting.*

Under Editor Settings:

- Text Editor
- Completion
- Add Type Hints

* We can only do this with an open project but it sets your local install up for all the following exercises.



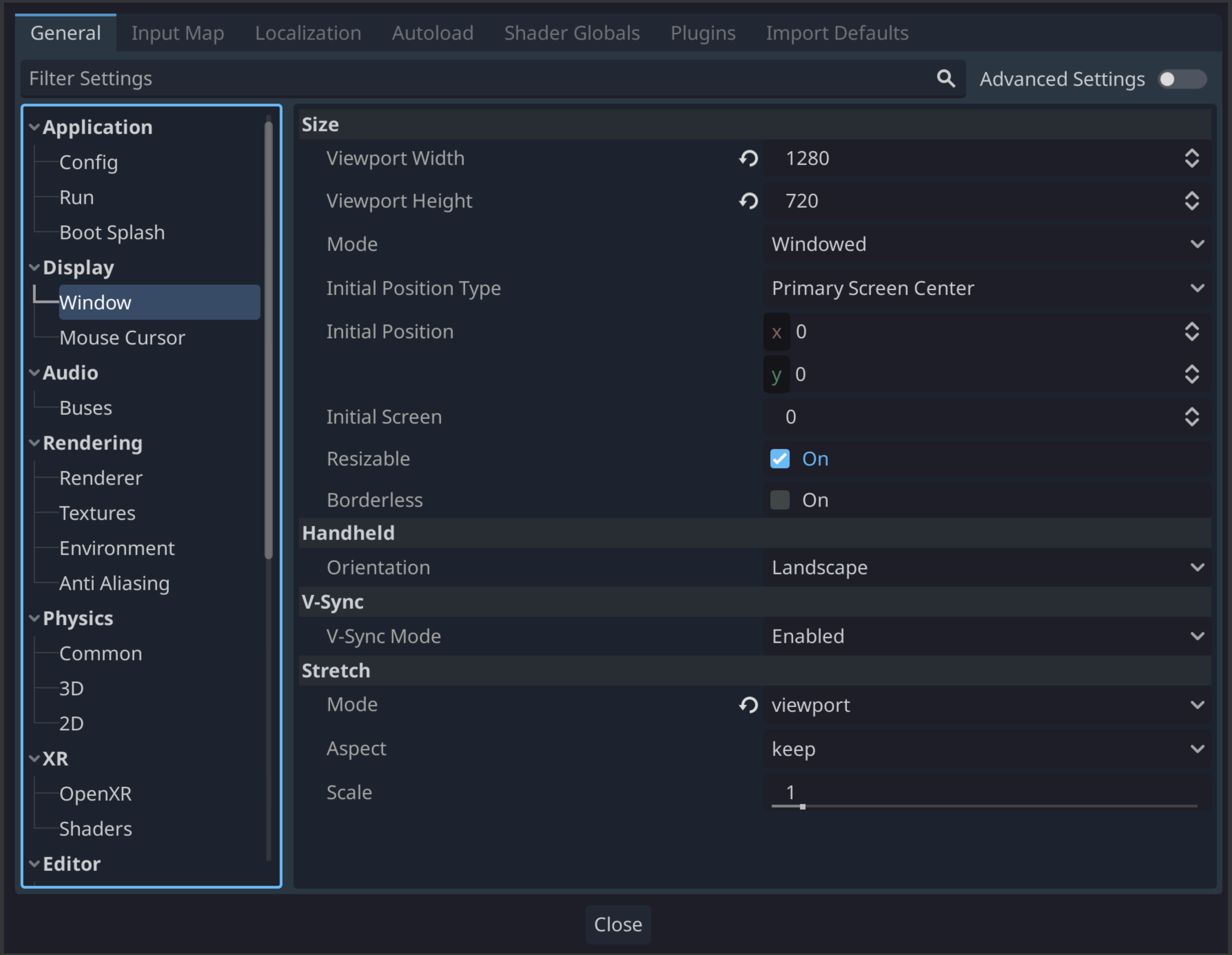
Idle Parse Delay	2
Auto Brace Complete	☑ On
Code Complete Delay	0.3
Put Callhint Tooltip Below Current Line	☑ On
Complete File Paths	☑ On
Add Type Hints	☑ On
Use Single Quotes	□ On

Project settings

These are the project settings that are changed from the defaults in the exercise files.

Display - Window

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



Size

Viewport Width 1280

Viewport Height 720

Mode Windowed

Initial Position Type Primary Screen Center

Initial Position x 0 y 0

Initial Screen 0

Resizable ☑ On

Borderless □ On

Handheld

Orientation Landscape

V-Sync

V-Sync Mode Enabled

Stretch

Mode viewport

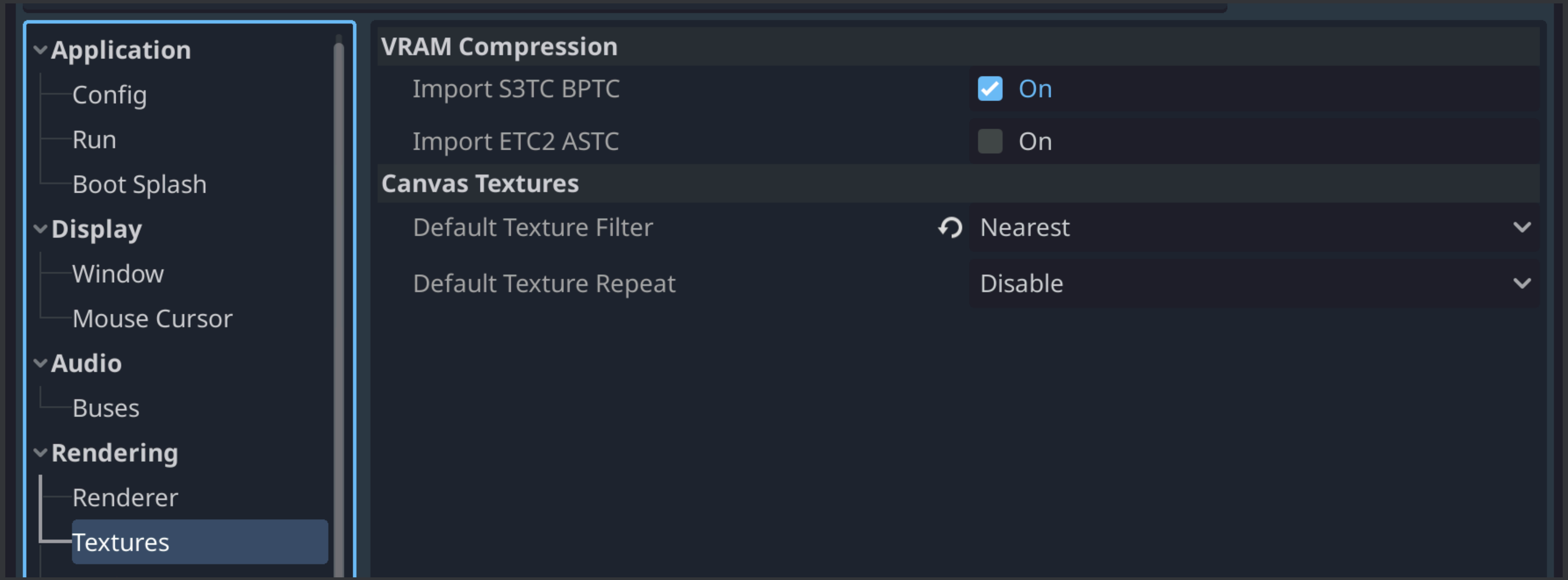
Aspect keep

Scale 1

Project settings

Rendering

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



VRAM Compression

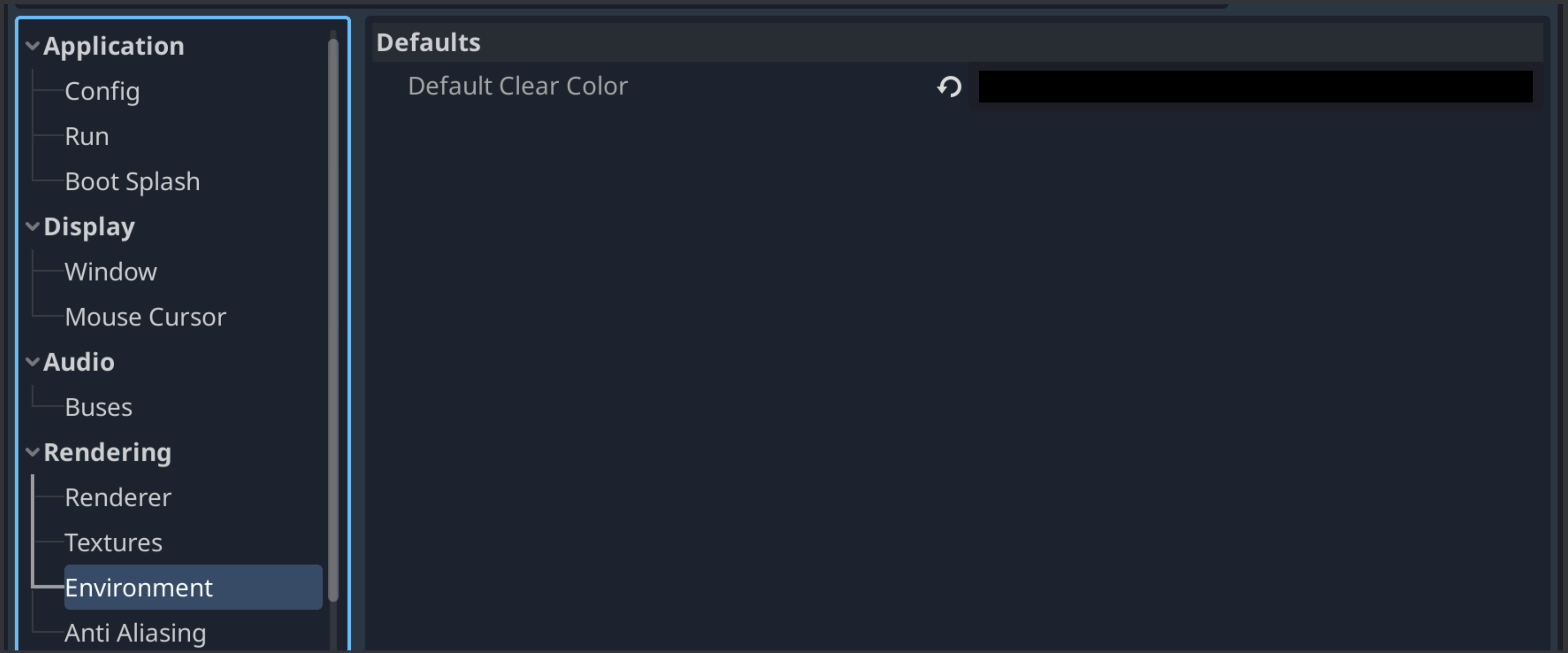
Import S3TC BPTC ☑ On

Import ETC2 ASTC □ On

Canvas Textures

Default Texture Filter Nearest

Default Texture Repeat Disable



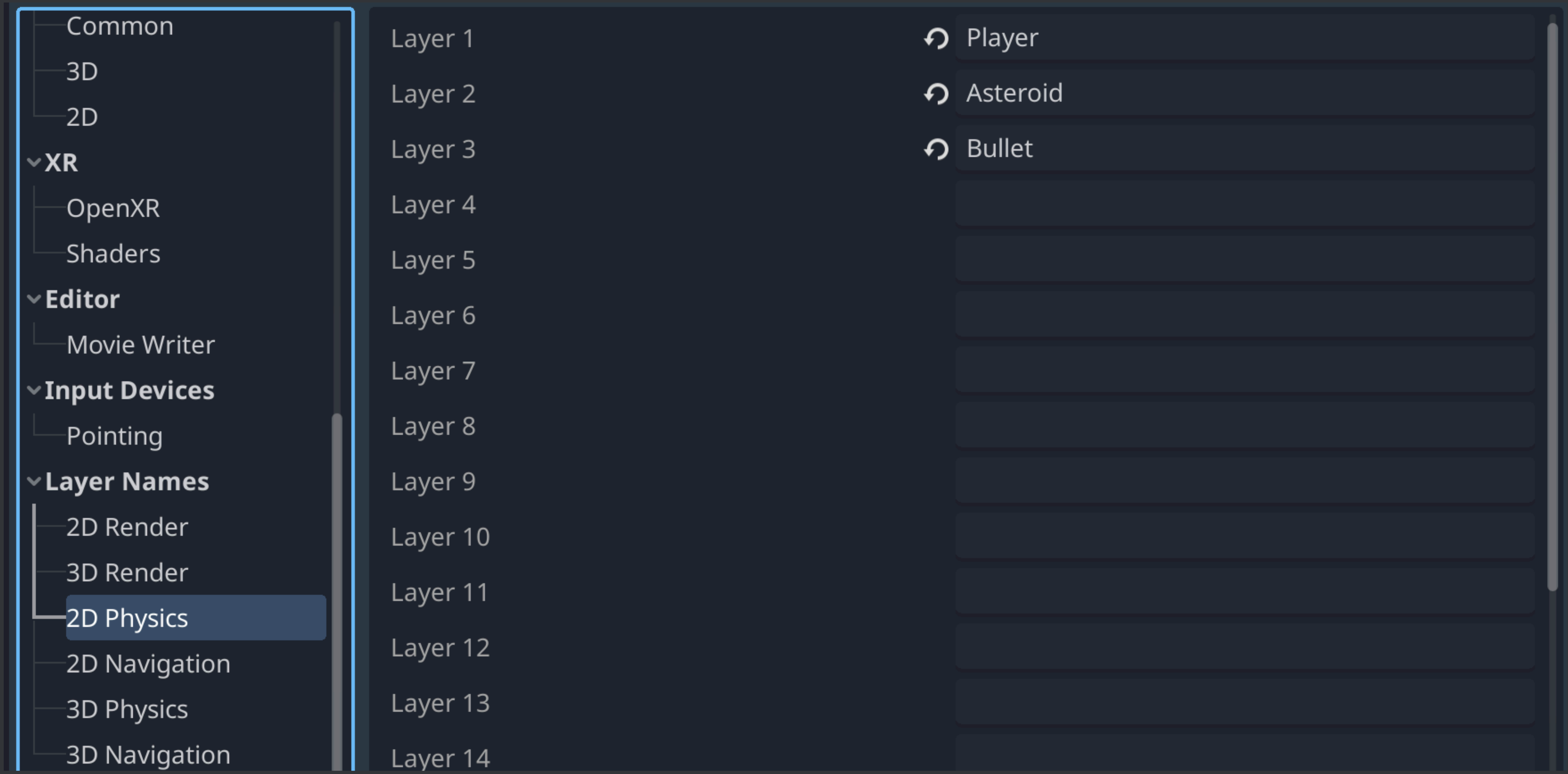
Defaults

Default Clear Color

Project settings

Layer Names - 2D Physics

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



Layer 1	Player
Layer 2	Asteroid
Layer 3	Bullet
Layer 4	
Layer 5	
Layer 6	
Layer 7	
Layer 8	
Layer 9	
Layer 10	
Layer 11	
Layer 12	
Layer 13	
Layer 14	

Provided files

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

Tilesheet

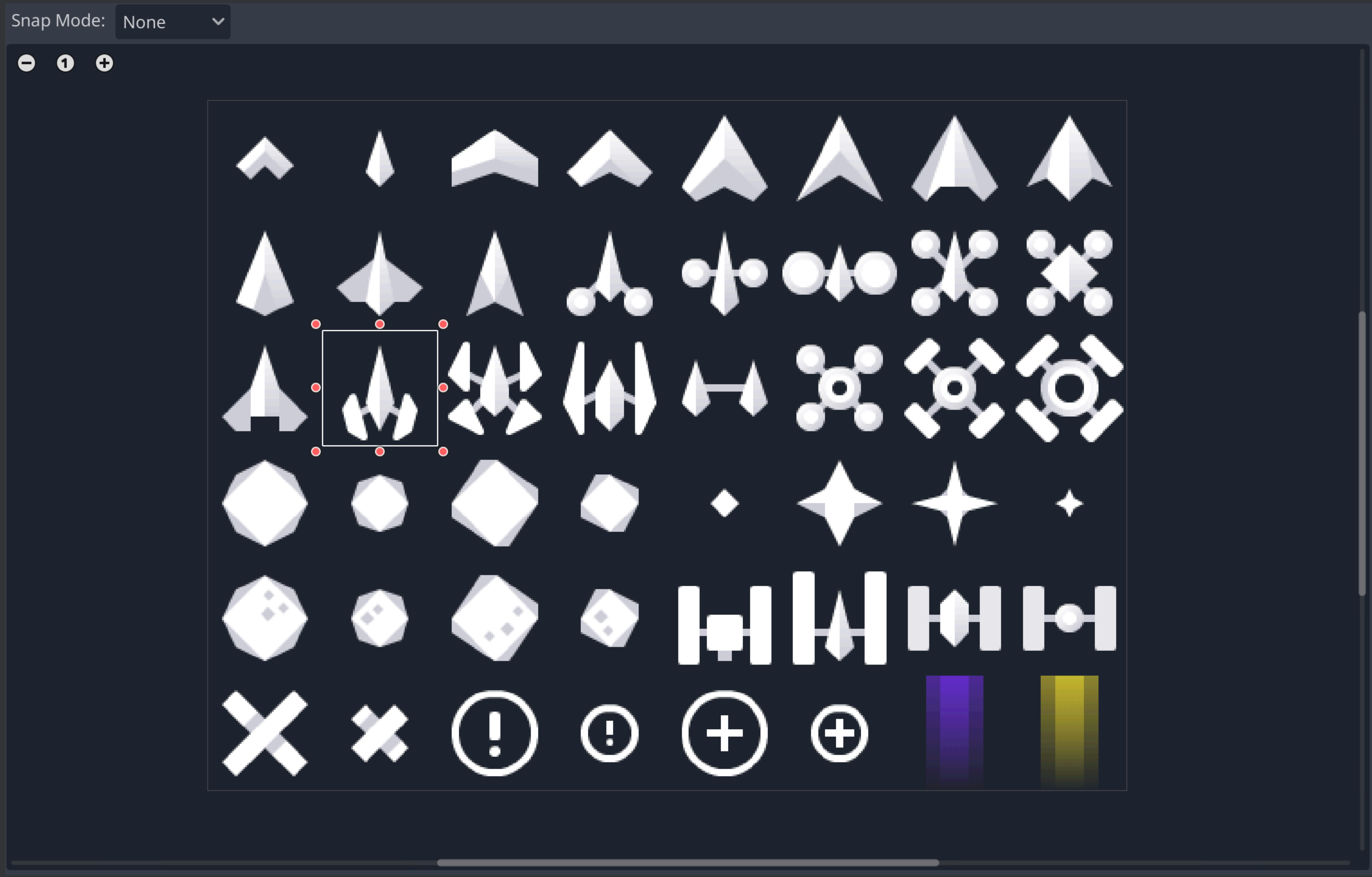
The tilesheet is a set of 64x64 pixel images.

We will select the ones we want by using

`Region`

- X/Y - top left corner - measured from top left of image
- W/H - size

E.g. Player - x: 64 y:128 w:64 h:64



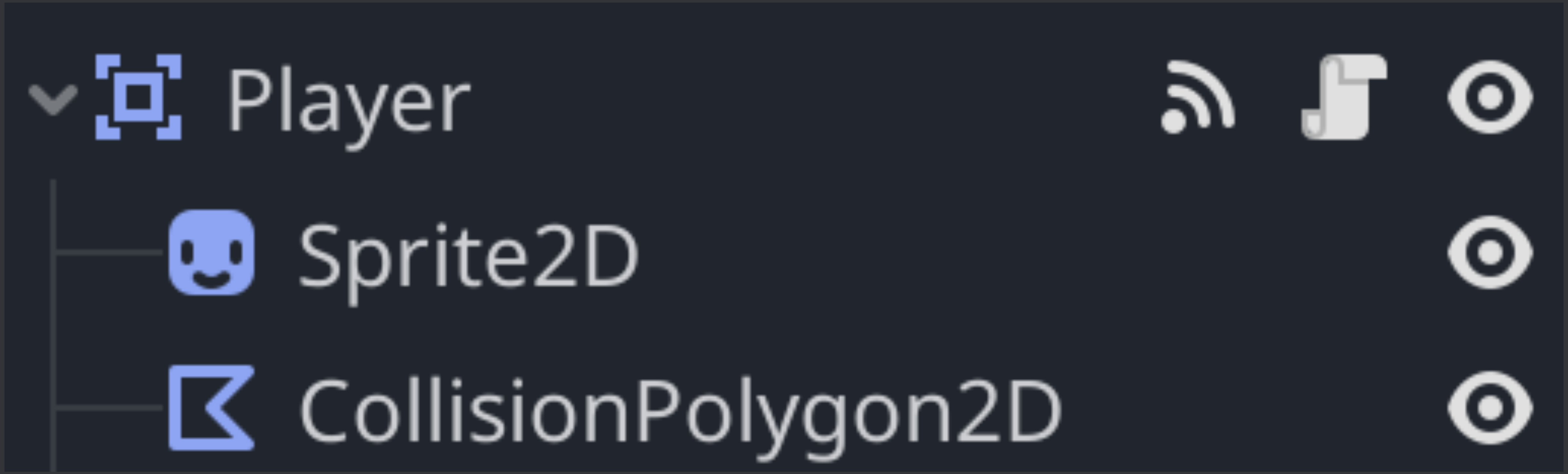
So - let's get started

Player Scene

We'll need a Player scene (Area2D)

It will need a sprite child node.

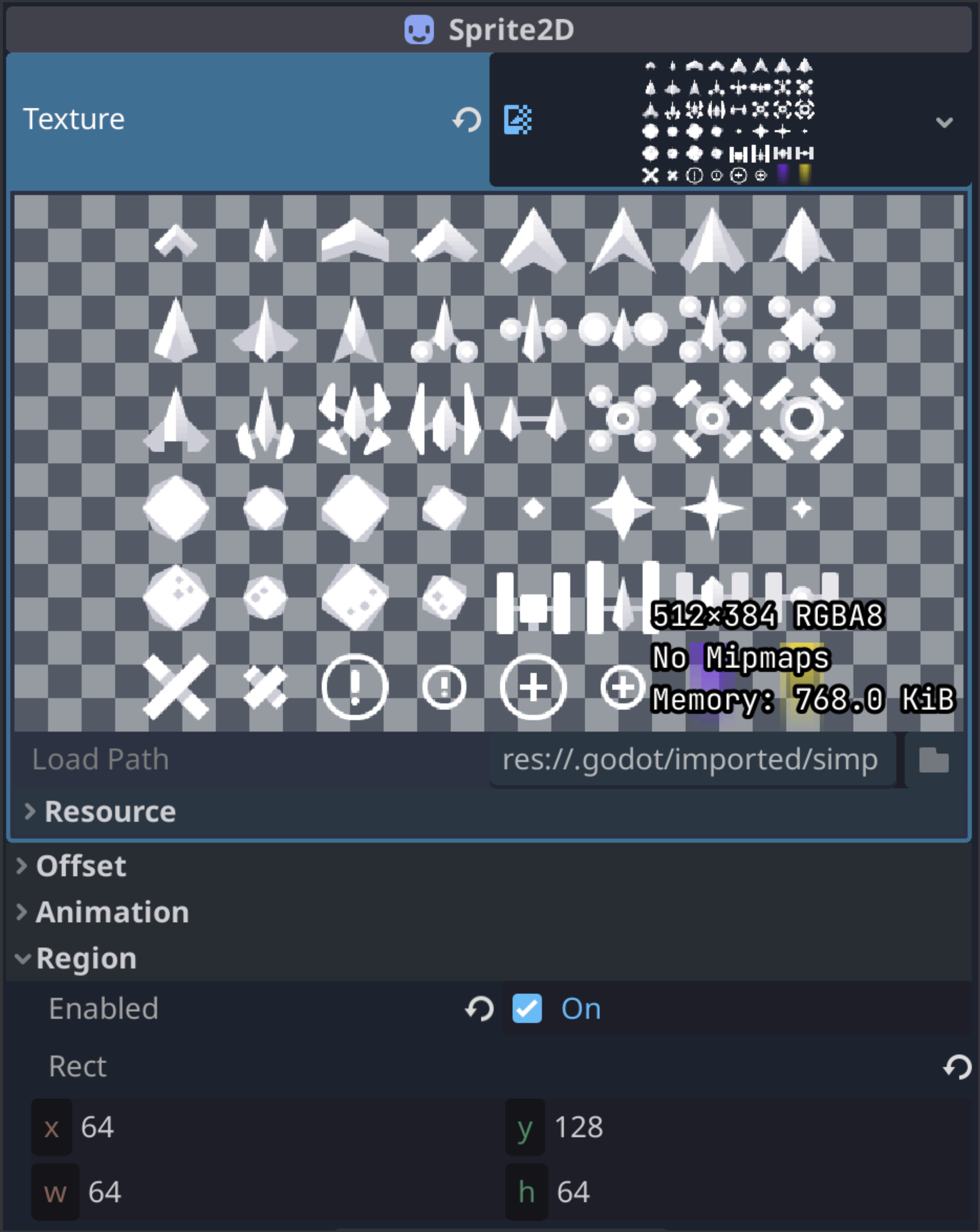
It will also need a collision child node.



Player
Sprite2D
CollisionPolygon2D

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



Sprite2D

Texture

512×384 RGBA8
No Mipmaps
Memory: 768.0 KiB

Load Path res://.godot/imported/simp

Resource

Offset

Animation

Region

Enabled ☑ On

Rect

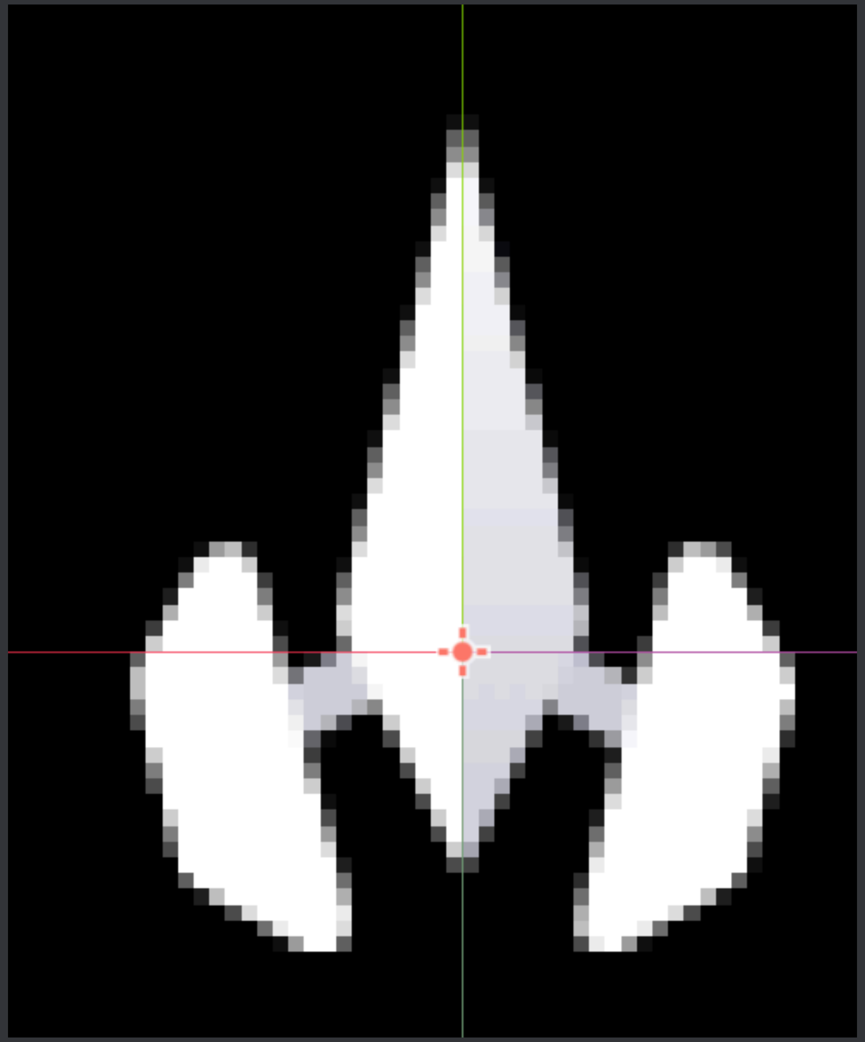
x 64 y 128

w 64 h 64

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

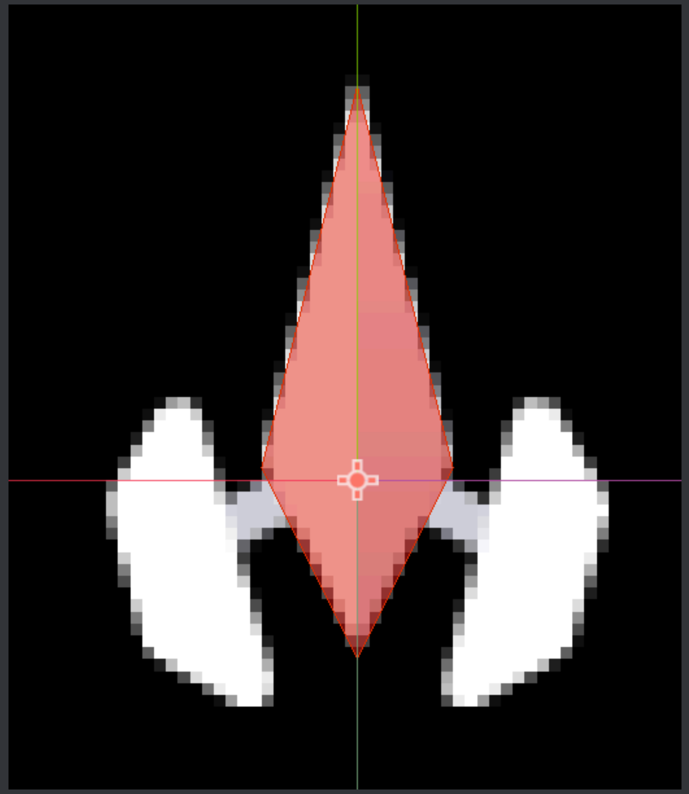
We want it about the widest point of the body.

- 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)
```

Exercise 2

Open exercise 2.

The player scene is set up - but we need to get rotation working.

Complete the player script.

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
```

Exercise 3

Open exercise 3.

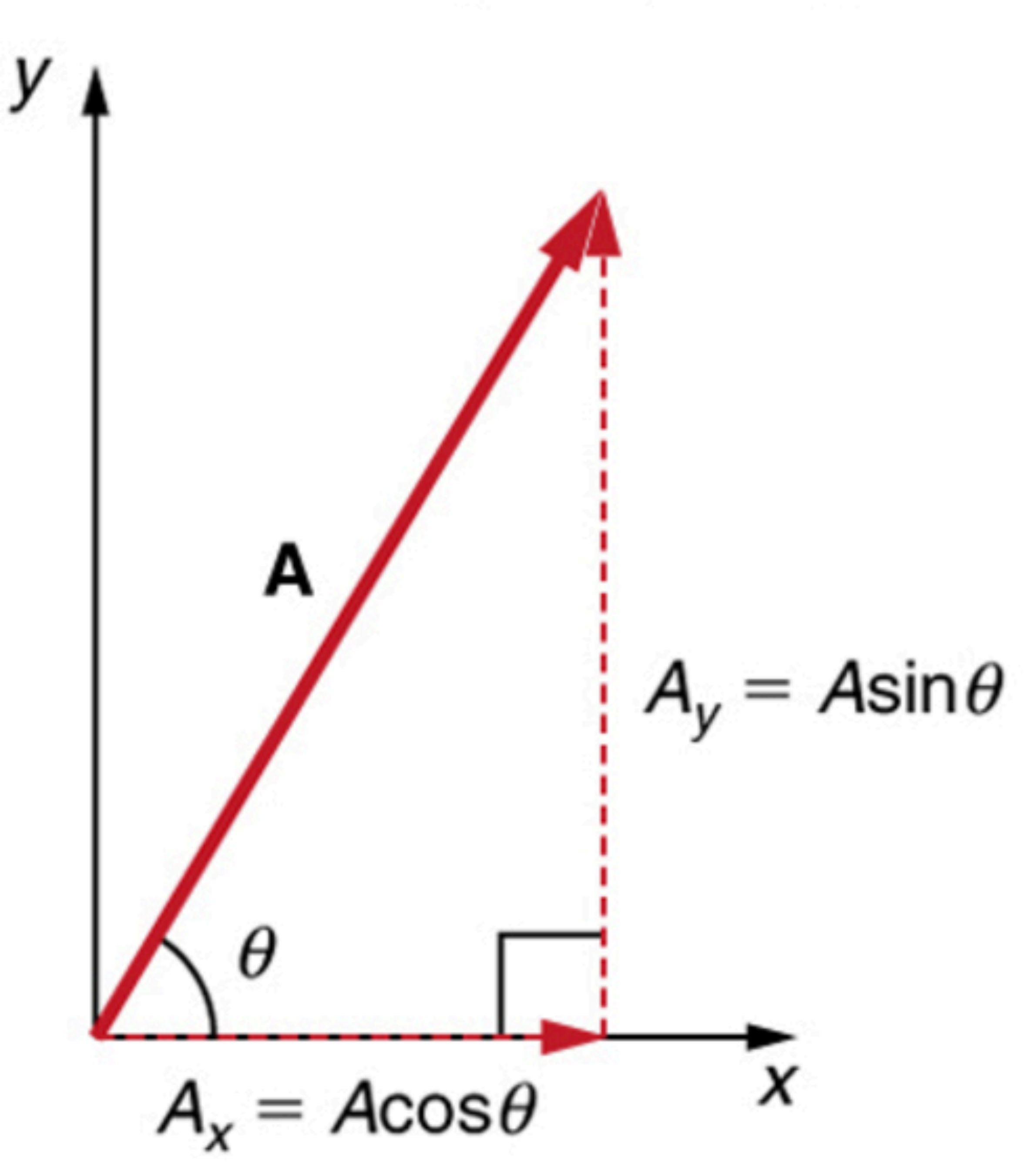
- Create the world scene (new Node2D scene)
- Add the player to it (drag the player scene to the node tree)
- Add script
- Copy in the starter script code from exercise3/exercise_starter.gd
- Complete the script

Player Movement

OK - let's get it to move too

Extend the player script

A little trigonometry



```
@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
```

Exercise 4

Open exercise 4.

- Complete the player script
- Run the game
- Fly around :)

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)
```

Exercise 5

Open exercise 5.

Very simple exercise - test out the screen_wrap function.

Do you understand what it is doing?

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

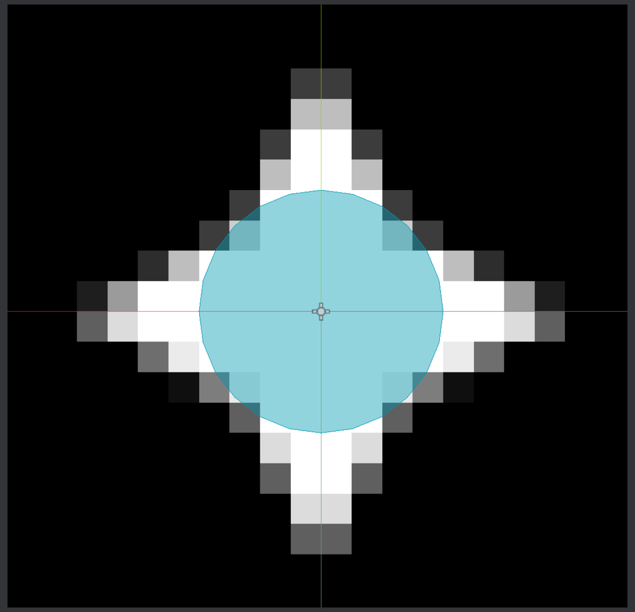
- load same texture as before
- enable region
- select region x:448 y:192 w:64 h:64

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[1]

[1] 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 (player)

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 bulet 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
- …
  ````
  ````