Space Shooter

Project Repo: https://gitlab.com/gamedev-cuni-cz/gpp/space-shooter

Starting point

All the necessary GUI is already prepared (main menu, end screen, HUD). Also the game's state is ready and globally accessible with all the necessary functions. The game scene is prepared with a scrolling background.

Steps

Reverse engineering to reverse movement

Look throughout the solution and make sure you understand how everything works. Inspect the nodes themselves and the associated GDScript.

Let's see if you understood how it works by reversing the way the parallax background scrolls!

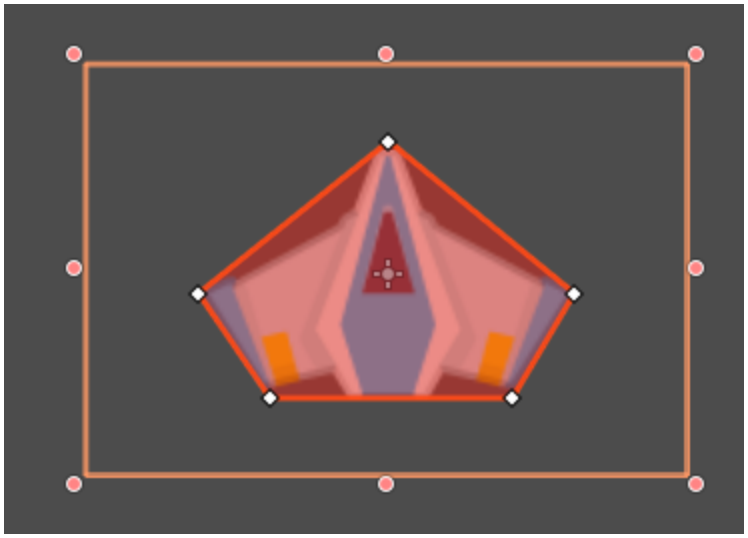
Player's movement

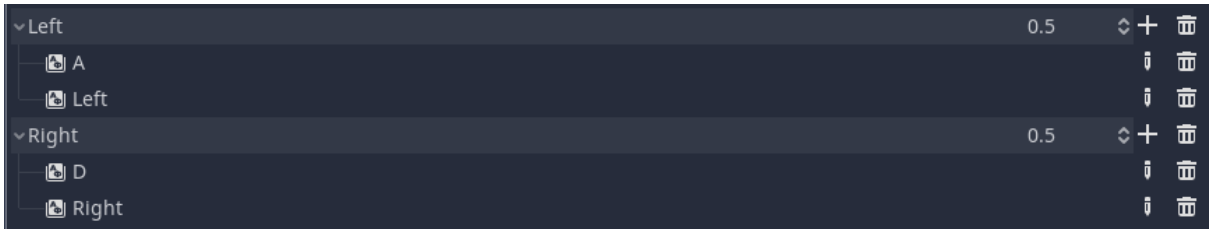
The player is able to move to the sides using A/D or Left/Right.

Task 1: Create a **Player** scene, which can move to the sides using A/D or Left/Right. Create it as an Area2D (so it can detect overlaps), containing a Sprite and CollisionPolygon2D.

CollisionPolygon2D — Godot Engine (stable) documentation in English
Input examples — Godot Engine (stable) documentation in English
CanvasItem — Godot Engine (stable) documentation in English (`get_viewport_rect()`)

WIth the Select Mode (Q) selected, we can create the collision polygon using the tools on the right.





The input map was set up.

Shooting

Task 2: Allow the player to shoot (create bullets which travel upwards) using the spacebar.

To instantiate objects, you need to have a reference to a PackedScene (obtained generally by the *preload* function), call it's *instance()* method and add it as a child of some Node (*node.add_child(obj)*).
The main game scene — Godot Engine (stable) documentation in English (instancing example)

*Optional: Create an array **bullets** in the GameState. Use it to keep track of all bullets on the screen.*

Destroy the bullets when they go out of the screen. Instead of Area2D Nodes on all sides of the screen, use the VisibleOnScreenNotifier2D node (which has a signal it emits when the bullet exits the screen).

Nodes and scene instances — Godot Engine (stable) documentation in English
VisibleOnScreenNotifier2D — Godot Engine (stable) documentation in English
Node — Godot Engine (stable) documentation in English (`queue_free()` method)

The input map was set up.

Spawning meteoroids

Task 3: Create a Meteoroid scene representing an indestructible meteoroid. Add an Obstacle Spawner node (as a container) with a Timer child, which will spawn a meteoroid every second.

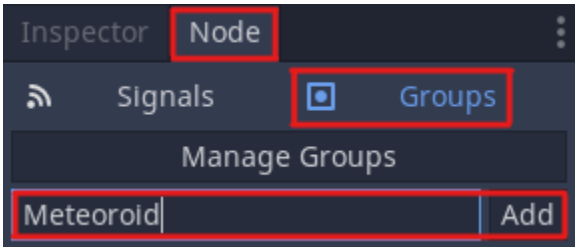
*Optional: Create an array **meteoroids** in the GameState. Use it to keep track of all meteoroids on the screen.*

Meteoroids moving in a random direction are spawned in regular intervals (and registered in the GameState.meteoroids array).

Random number generation — Godot Engine (stable) documentation in English
Timer — Godot Engine (stable) documentation in English

Collisions with meteoroids

Task 4: The player should lose a life on collision with the meteoroids and the meteoroid is destroyed.
Groups — Godot Engine (stable) documentation in English
Node — Godot Engine (stable) documentation in English (`is_in_group()` method)



Adding the selected node to a group.

Bullets destroyed on collision

Task 5: If the bullets collide with the meteoroids, they get destroyed. Use signals.

Enemies

Task 6: Create an **Enemy** scene that bounce from the left to the right, then to the left again while shooting in regular intervals (use the Timer node). They also move slowly down and are destroyed after leaving the screen.

It is recommended to create a new **Enemy Bullet** node instead of using the original **Bullet** for both player's and enemy's.

*Optional: Create an array **enemies** in the GameState. Use it to keep track of all enemies on the screen.*

Spawning enemies

Task 7: Enemies are spawned in regular intervals.
VisibleOnScreenNotifier2D — Godot Engine (stable) documentation in English

Destroying enemies

Task 8: When an enemy is hit, it is destroyed and the bullet as well.

Enemies hurting the player

Task 9: The player loses a life on collision with the enemies' bullets or the enemies themselves.

Score

Task 10: The player gets points for every avoided meteoroid/enemy and every destroyed enemy.

Optional Tasks

Opt. Task 1: Refactor! There's probably a lot of copy-pasted code right now. Try to deal with it by inheriting from your custom base class (which derives from Node2D). (At least go for a common ancestor for the EnemyBullet and the Bullet).
Opt. Task 2: Improve the gameplay of the game and try to make it fun!

Useful resources

Step by step — Godot Engine (stable) documentation in English
GDScript — Godot Engine (stable) documentation in English
Applying object-oriented principles in Godot — Godot Engine (stable) documentation in English

Assets sources

Kenney • Space Shooter Redux