SPACE SHOOTER ART PACK 02

Documentation version 1.0

Questions about this product? Send an email to support@playniax.com

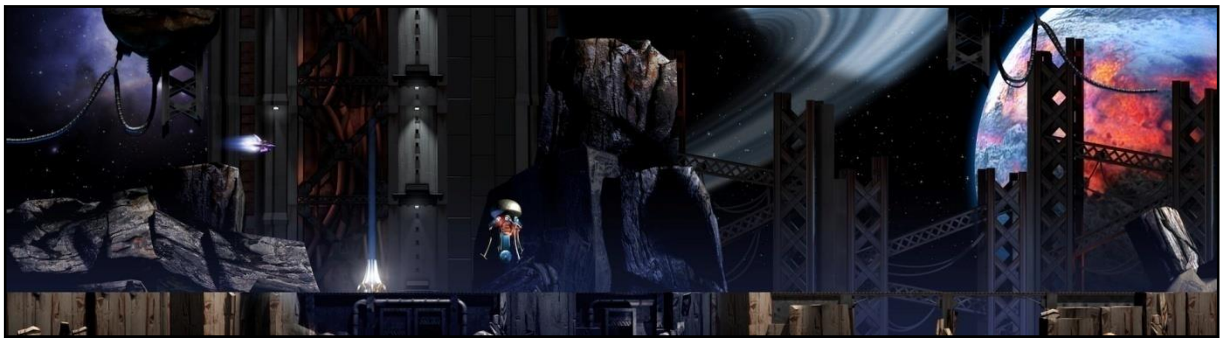


Table of contents

1. INTRODUCTION

WANT TO MAKE A HORIZONTAL SPACE SHOOTER BUT YOU'VE GOT NO ART?

As a coder or game designer you sometimes run into this problem of having no graphics for your game. We know the problem all too well so we released this set of graphics that look cool and are very helpful in bringing your Shoot 'Em Up come to life!

ART WORK FEATURES

- Background images for making beautiful scenes
- 1 animated player ship.
- 12 different types of animated attackers
- 2 mid games animated bosses
- 1 end game animated big boss
- 5 different pickups

BONUS

- Sprites overview scene with all basic behavior scripts attached
- Ignition, a small framework to bring the sprites to life (Bullet Spawner, Object Pooler, Sprite Collider and more).
- Optimized 2D Particle System + particle effects (explosions, rocket exhaust and nuke)

2. USAGE

The Sprites Overview Scenes can be found in the "Playniax / Space Shooter Art Pack 02 / Demo / Scenes" folder.

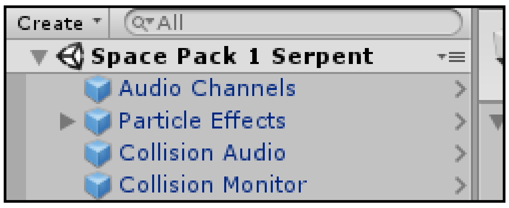
Basic behavior is pre-coded and the scripts for this can be found in the "Playniax / Space Shooter Art Pack 02 / Demo / Scripts" folder.

In addition to the basic behaviors there is also a little framework called Ignition that takes care of bullets, collisions, effects and other handy stuff. For more Ignition examples and explanation have a look at the Examples folder in the Playniax/Ignition Folder.

You can discard the Ignition scripts and replace them with your own but the way this pack is setup you just need to copy and paste the sprites into your own scene, add your own AI scripts and you are good to go as long as certain prefabs (Audio Channels, Collision Audio, Collision Monitor and Particle Effects) are in your scene as well.

The prefabs

The sprite overview scenes contain 4 prefabs i.e. Audio Channels, Collision Audio, Collision Monitor and Particle Effects and these take care of the collisions, visual effects (explosions and smoke for example) and sound effects (these 4 prefabs replace the 1 Engine prefab in previous versions of Ignition).



Create
All
Space Pack 1 Serpent
Audio Channels
Particle Effects
Collision Audio
Collision Monitor

Make sure for each scene you make to drag the prefabs to your scene and all copied sprites should work or (inter) act like they are intended. Additional sprite behaviors or AI are up to you.

Note that the Ignition framework is a work in progress and that the docs are in its infancy but usage should be intuitive.

The player

The player script and pickups also really work. Scripts for making this happen are WeaponsSystem.cs and Pickup.cs.

Pickups from left to right are : Automatic aiming weapon, nuke the place, laser, more bullets for the front gun and shield.

Player can be controlled by the cursor keys or WASD.

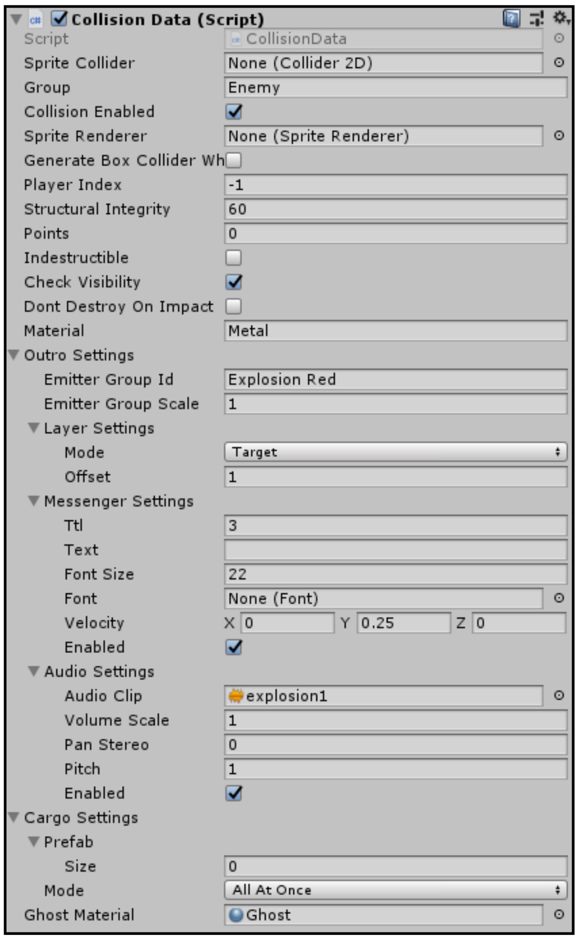
This is handled by the PlayerControls.cs script that has more options like Swipe or Mouse.

The Ignition Collision system

The collision system basically uses 3 scripts and depend on the 4 prefabs:

CollisionMonitor.cs - Takes care of the collision detection between 2 groups.
CollisionAudio.cs - Takes care of the sound effects to be played when sprites collide.
CollisionData.cs - This is a custom data script and you attach them to a sprite.

The Collision Data settings determines to what collision group a sprite belongs, how tough it is, what sound it plays on impact, what sounds it makes or what particle effect it shows when it gets killed, what message is displayed etc. CollisionData uses 2D colliders to detect collisions.



Collision Data (Script)	✓
Script	CollisionData
Sprite Collider	None (Collider 2D)
Group	Enemy
Collision Enabled	✓
Sprite Renderer	None (Sprite Renderer)
Generate Box Collider Wh	☐
Player Index	-1
Structural Integrity	60
Points	0
Indestructible	☐
Check Visibility	✓
Dont Destroy On Impact	☐
Material	Metal
Outro Settings	
Emitter Group Id	Explosion Red
Emitter Group Scale	1
Layer Settings	
Mode	Target
Offset	1
Messenger Settings	
Ttl	3
Text	
Font Size	22
Font	None (Font)
Velocity	X 0 Y 0.25 Z 0
Enabled	✓
Audio Settings	
Audio Clip	explosion1
Volume Scale	1
Pan Stereo	0
Pitch	1
Enabled	✓
Cargo Settings	
Prefab	
Size	0
Mode	All At Once
Ghost Material	Ghost

CollisionData (Script)

Group - This identifier is used by the CollisionMonitor script (Engine prefab) to know what objects should collide.

Material - This identifier is used by the CollisionAudio script (Engine prefab) to know what sound to play when sprites collide.

Player Index - Value above -1 tells the script to add the points to the players scoreboard. So if Player Index is 0 the points are added to the scoreboard of player 1. There is not really an example of multiple players so keep it 0 in this case. This can be used when the sprite is a bullit for example.

Structural Integrity - When this reaches zero the sprite will be destroyed and the particles set at **Outro Settings** will be played. Structural Integrity is lowered every time sprites collide.

Basically the strongest ‘wins’.

Points - Points are added to the player scoreboard when you destroy the sprite. If Points is zero it will add the Structural Integrity value multiplied by 10.

Collision Enabled - Detection on or off.

Check Visibility - Only detect collision with onscreen sprites

Targetable - Targetable sprites can be shot down by the autonomous weapons such as the laser.

Outro Settings - Emitter Group Id, scale and offset determine what particle effect is played when the sprite is destroyed (and it’s relative size and layer). Note that the emitter need to be present. Have a look at the engine prefab for example usage.

Messenger Settings - Will show points collected by player or custom text.

Audio Settings - Sound effect that will be played when the sprite is destroyed.

Cargo Settings (Script) – A sprite can have cargo. Cargo will be released when a sprite is destroyed.

Additional Settings – can be ignored and collects data automatically.

3. IMPORTANT VARIABLES AND FUNTIONS

PlayerData.Get(index).scoreboard - Is used to keep track of how many points the player has. Index 0 is player 1, index 1 is player 2 etc.

Timing.Paused() - Returns True if game is paused (TimeScale = 0)

4. ADDITIONAL INFO

LICENSE (this basically sums it up)

- This product can be used to create non-commercial or commercial titles
- Credit / attribution is not mandatory, but appreciated
- The assets itself cannot be shared freely or (re)sold!

CREDITS

- Programming by **T**ony **S**mits
- Graphics by **P**eter van **D**riel