

3D WAVE SHOOTER

Welcome to the readme file! Hopefully this can explain to you what the project contains, how it functions and what you can do to get the most out of it.

Contents

Game Features	3
What does the project contain?	3
How the game works	3
File Layout	4
Customizable Aspects	6
Create a Weapon	6
Create a projectile	9
Create a Level	11
Create an Effect	14
Scripts	16
Camera	16
Enemy	16
Managers	16
Other	17
Pickup	17
Player	18
ScriptableObjects	18
UI	18
Weapon	19
Animators	20
Player	20
Enemy	20
Level Required Prefabs	21
Mobile Support	22
How to Enable Mobile Controls	22
FAQ	23

Contact	23
Changes	24
v1.1	24
v1.2	24
v1.2.1	24
v1.2.2	24

Game Features

3D Wave Shooter, is complete project made in the Unity game engine. The player fights of waves of enemies with various different weapons on a range of maps.

- Waves of varied enemies.
- A range of different weapons, each unique.
- Destructible world objects (crates, explosive barrels).
- In-game shop to purchase weapons and upgrades between rounds.
- Mobile controls.

What does the project contain?

The project contains all the assets you need to create a top-down, 3D wave shooter game. More specifically, the project contains:

- Fully documented code.
- Scriptable objects to create custom weapons, projectiles levels and effects.
- Object pooling.
- Basic 3D environment models.
- Sound effects.
- 2 pre-made levels.
- And a fully working game!

How the game works

3D Wave Shooter, is a wave based, top-down shooter. At the start of a game a countdown begins and then enemies start to spawn. Kill them off and at the end of the round you can purchase weapons and various other things from the shop.

- New weapons.
- Weapon upgrades.
- Player upgrades.

Once ready, you can click “Next Wave” to progress onto the next wave. As the game proceeds, it gets harder and harder, with maybe even a boss or two entering the arena. If you manage to succeed and kill off all the final enemies, you win.

- **Controls:** WASD to move, Mouse to rotate, Left Mouse Button to shoot, Scroll Wheel to rotate through equipped weapons.

File Layout

I know that when entering an existing project, figuring out how the folders are setup can be a confusing and time consuming process.

- ❑ Animations
 - ❑ Enemy
 - ❑ *All enemy animations and animation controller.*
 - ❑ Player
 - ❑ *All player animations and animation controller.*
- ❑ Audio
 - ❑ Music
 - ❑ *All the game's music.*
 - ❑ SFX
 - ❑ *All the game's SFX, divided up into categorical folders.*
- ❑ DecorateFoldout
 - ❑ *Used for scriptable objects to add foldouts to the inspector.*
- ❑ Effects
 - ❑ *Custom made effects.*
- ❑ Fonts
 - ❑ *All the game's fonts.*
- ❑ Level Data
 - ❑ *Custom made level data.*
- ❑ Materials
 - ❑ *All the materials used for the 3D models.*
- ❑ Models
 - ❑ *All the 3D models used in the game.*
- ❑ Prefabs
 - ❑ Damageable
 - ❑ *All destructible world objects. Explosive barrels, crates, etc.*
 - ❑ Effects
 - ❑ *Visual elements.*
 - ❑ Enemies
 - ❑ *All the spawnable enemies.*
 - ❑ Environment
 - ❑ *All 3D environmental assets.*
 - ❑ Particles
 - ❑ *All particles.*
 - ❑ Pickups
 - ❑ *All spawnable pickups.*
 - ❑ Projectiles
 - ❑ *All physical projectiles.*
 - ❑ Requirements
 - ❑ *Prefabs that every level in the game must have in the scene.*

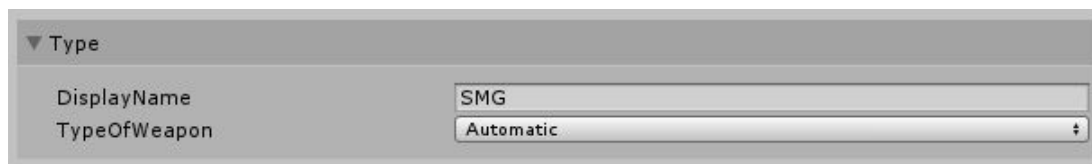
- ❑ Templates
 - ❑ *Template prefabs you can use to easily make custom content.*
- ❑ Weapons
 - ❑ *Weapon visuals that the player holds when that weapon is equipped.*
- ❑ Projectiles
 - ❑ *Custom made projectiles.*
- ❑ Scenes
 - ❑ *All the scenes in the game.*
- ❑ Scripts
 - ❑ Camera
 - ❑ *Scripts that affect the camera.*
 - ❑ Enemy
 - ❑ *Scripts related to enemies (movement, ai, etc).*
 - ❑ Managers
 - ❑ *Audio manager, enemy spawner, particle manager, etc.*
 - ❑ Mobile
 - ❑ *Mobile controls.*
 - ❑ Other
 - ❑ *Non-categorical scripts (effect, pickup highlight, explosion, etc).*
 - ❑ Pickup
 - ❑ *Scripts related to pickup entities.*
 - ❑ Player
 - ❑ *Scripts related to the player (movement, attacking, etc).*
 - ❑ ScriptableObjects
 - ❑ *Scripts for all customizable elements (weapon, projectiles, effects, etc).*
 - ❑ UI
 - ❑ *Scripts that display/interact with UI elements.*
 - ❑ Weapon
 - ❑ *Scripts related to weapon objects (projectiles, etc).*
- ❑ Textures
 - ❑ *All the game's textures (not many as most materials are just flat colours).*
- ❑ UI
 - ❑ *Sprites used as UI elements.*
- ❑ Weapons
 - ❑ *Custom made weapons.*

Customizable Aspects

This project is very customizable as it was built up to be a starting point for developers. Weapons, projectiles, effects and level layouts (waves, timing, etc) are all scriptable objects that you can create.

Create a Weapon

1. Weapons are stored in the *3DWaveShooter/Weapons* folder. There, right click and select Create > *Weapon* to create a new *WeaponScriptableObject*.

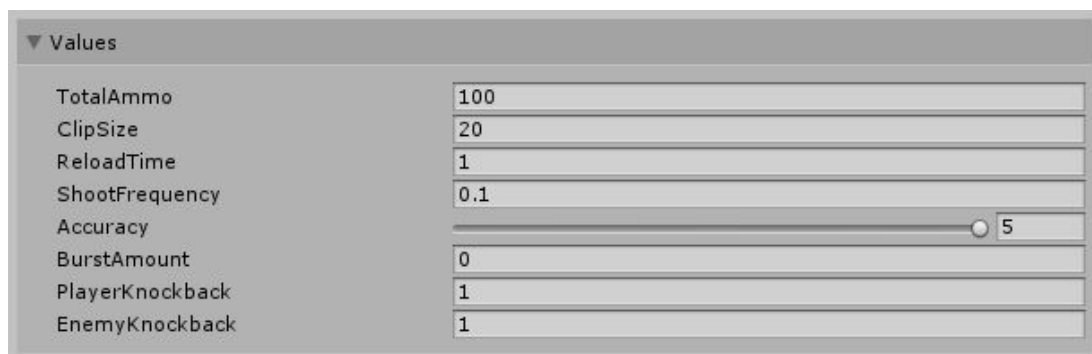


The screenshot shows the 'Type' section of the *WeaponScriptableObject* inspector. It contains two fields: 'DisplayName' with the value 'SMG' and 'TypeOfWeapon' with a dropdown menu set to 'Automatic'.

▼ Type	
DisplayName	SMG
TypeOfWeapon	Automatic

Display Name: Weapon name which is displayed on UI elements.

Type of Weapon: Firing mode. Single Shot, Automatic or Burst.



The screenshot shows the 'Values' section of the *WeaponScriptableObject* inspector. It contains several fields for weapon parameters: TotalAmmo (100), ClipSize (20), ReloadTime (1), ShootFrequency (0.1), Accuracy (a slider set to 5), BurstAmount (0), PlayerKnockback (1), and EnemyKnockback (1).

▼ Values	
TotalAmmo	100
ClipSize	20
ReloadTime	1
ShootFrequency	0.1
Accuracy	5
BurstAmount	0
PlayerKnockback	1
EnemyKnockback	1

Total Ammo: Total amount of ammo the weapon will have.

Clip Size: Amount of ammo per clip.

Reload Time: Duration in seconds to reload.

Shoot Frequency: Minimum amount of time in seconds between shots.

Accuracy: Random projectile spread. Higher it is the more random it is.

Burst Amount: Amount of projectiles shot in a burst.

Player Knockback: Force applied to the player backwards when firing.

Enemy Knockback: Force applied to the enemy backwards when hit.

▼ Prefabs

Projectile	PistolBullet (ProjectileScriptableObject)	⊙
ShootSFX	SMG	⊙
Visual	SMG	⊙
DroppedPickup	SMGPickup	⊙
UiIcon	SMGIcon	⊙

Projectile: ProjectileScriptableObject of the projectile that the weapon will shoot.

ShootSFX: Sound effect that will be played every time a projectile is shot.

Visual: GameObject that is spawned in the player's hands.

Dropped Pickup: Dropped object (pickup) of the gun. More about this below.

UI Icon: Icon of the weapon displayed on UI. White, 500 x 500 pixels.

▼ Offsets

▼ Offsets

Position Offset	X 0	Y 0.1	Z 0
Rotation Offset	X 0	Y 0	Z 0

Position Offset: Offset applied to the Visual's position when spawned in the player's hands.

Rotation Offset: Offset applied to the Visual's rotation when spawned in the player's hands.

▼ Upgrades / Purchase

How much does it cost to buy this gun in the store?

PurchaseCost 500

▼ Upgrades

Size 2

▼ Element 0

Cost to Upgrade

Cost 500

Stat Change

(Only changes is value isn't 0)

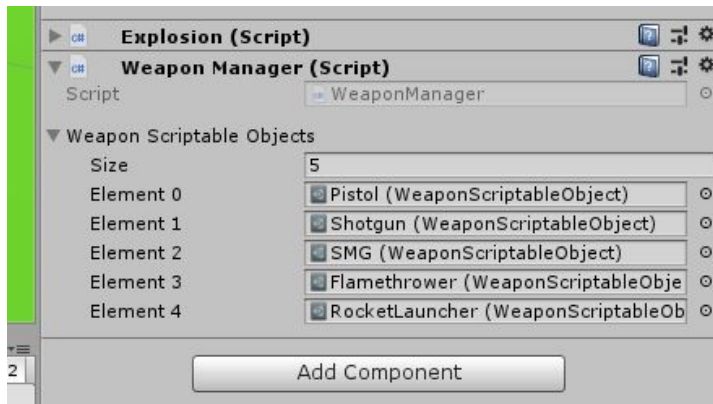
Reload Time	0.8
Shoot Frequency	0.08
Accuracy	5
Burst Amount	0
Player Knockback	0
Enemy Knockback	0

► Element 1

Purchase Cost: Price to purchase the weapon in the shop. Not needed if starting weapon.

Upgrades: Each array element is a new upgrade that can be purchased in the store. They are sequential, so the player upgrades to Element 0, then 1, 2, etc. As many as you have.

Stat Change: Change to weapon stat when upgraded. 0 means no change to stat.



2. Now, you want to go to the *_GameManager* object and find the *Weapon Manager* script. Add the new weapon to the array of *Weapon Scriptable Objects*.



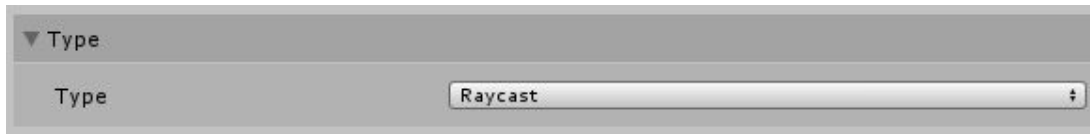
3. Next, you need to create an object for the player to hold. Either make a new object or import a model and place it in the player's *WeaponHoldPos* transform as a child. Note down the offsets of the position and rotation and put them in the weapon file.

Save this prefab in the *3DWaveShooter/Prefabs/Weapons* folder.

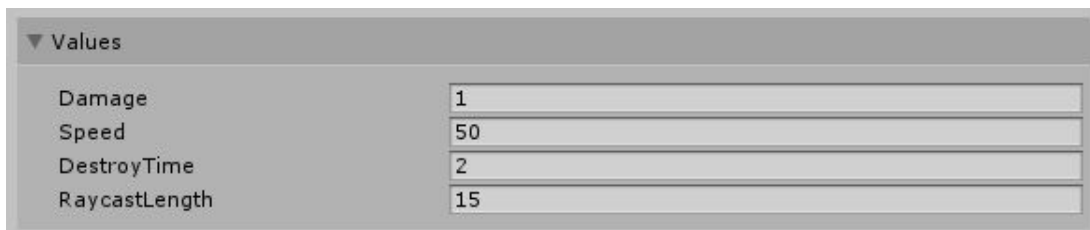
Now you have a new weapon which you can purchase in the shop!

Create a projectile

1. Projectiles are stored in the *3DWaveShooter/Projectiles* folder. There, right click and select *Create > Projectile* to create a new *ProjectileScriptableObject*.



Type: Type of projectile. Raycast shoots out a raycast, Projectile spawns a physical object.

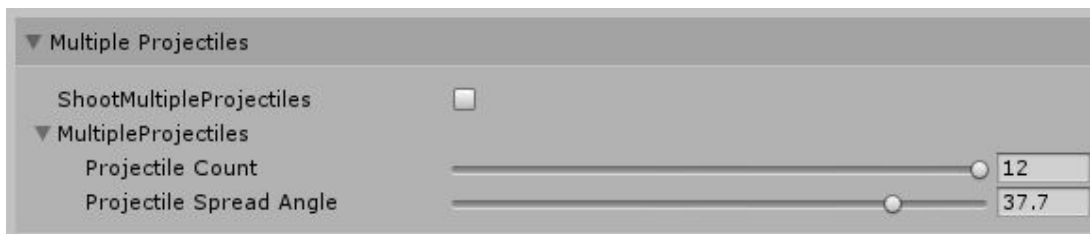


Damage: Damage given to an enemy upon impact.

Speed: Speed of the projectile. Not needed if raycast.

Destroy Time: Projectile destroys itself after this amount of time. Not needed if raycast.

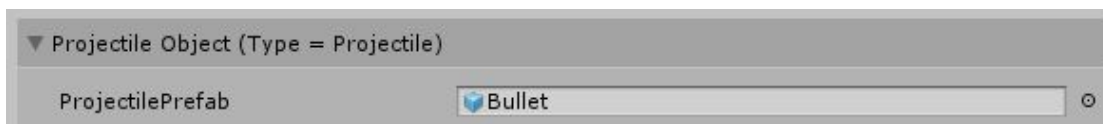
Raycast Length: How far the raycast will look for enemies. Not needed if projectile.



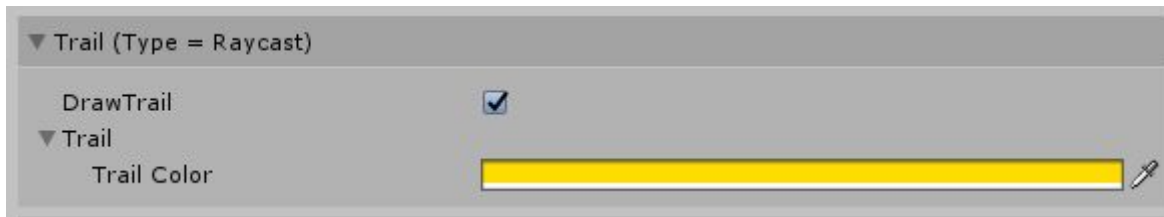
Shoot Multiple Projectiles: Some weapons like shotguns fire of a number of pellets. Enabling this will mean the weapon shoots out multiple of this projectile every fire.

Projectile Count: Number of projectiles to shoot.

Projectile Spread Angle: Angle from the player that these projectiles will spread out from.

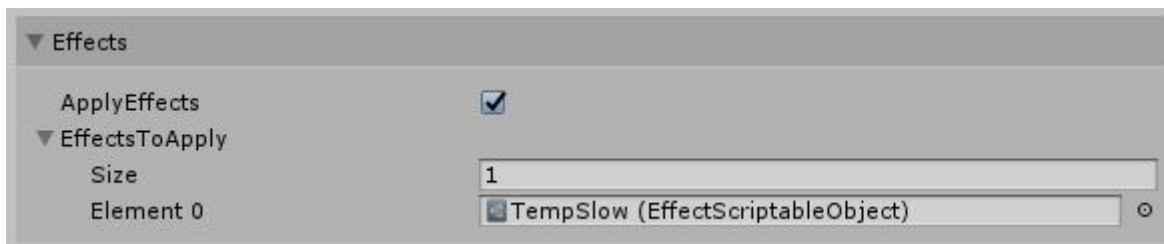


Projectile Prefab: Prefab that will spawn as the projectile. Not needed if raycast.



Draw Trail: Raycast projectiles can have a trail so the player can easily see where and how far it went.

Trail Color: Color of the trail.



Apply Effects: When enemies get hit by a raycast or physical projectile, an effect can be applied to them. This can be a temporary slow, or something like fire damage or an explosion for a rocket launcher.

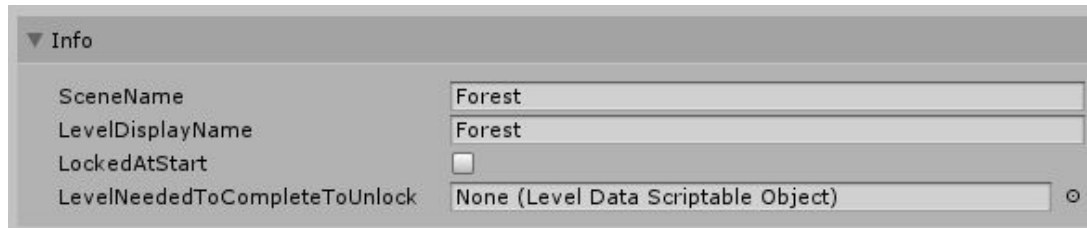
Effects to Apply: An array holding all of the EffectScriptableObjects to apply to the hit enemy.

2. If your projectile type is *Projectile*, you will need a prefab for it to shoot. Go to the *3DWaveShooter/Prefabs/Templates* folder and find the *ProjectileTemplate*. Use this to apply your own projectile model and save it to the *3DWaveShooter/Prefabs/Projectiles* folder with all the existing ones. Don't worry about filling in any of the values in the *Projectile* script attached as this is done automatically.
3. Finally, go to the *WeaponScriptableObject* file that you want to use this projectile with and make sure it's linked.

Create a Level

Each level has a `EnemyLayoutScriptableObject` file to define the waves that will spawn.

1. Enemy Layouts are stored in the `3DWaveShooter/Level Data` folder. There, right click and select `Create > Enemy Layout` to create a new `EnemyLayoutScriptableObject`. I'd name this the name of your level like "Ruins", or something like "Level 2".



▼ Info	
SceneName	Forest
LevelDisplayName	Forest
LockedAtStart	<input type="checkbox"/>
LevelNeededToCompleteToUnlock	None (Level Data Scriptable Object) ⓘ

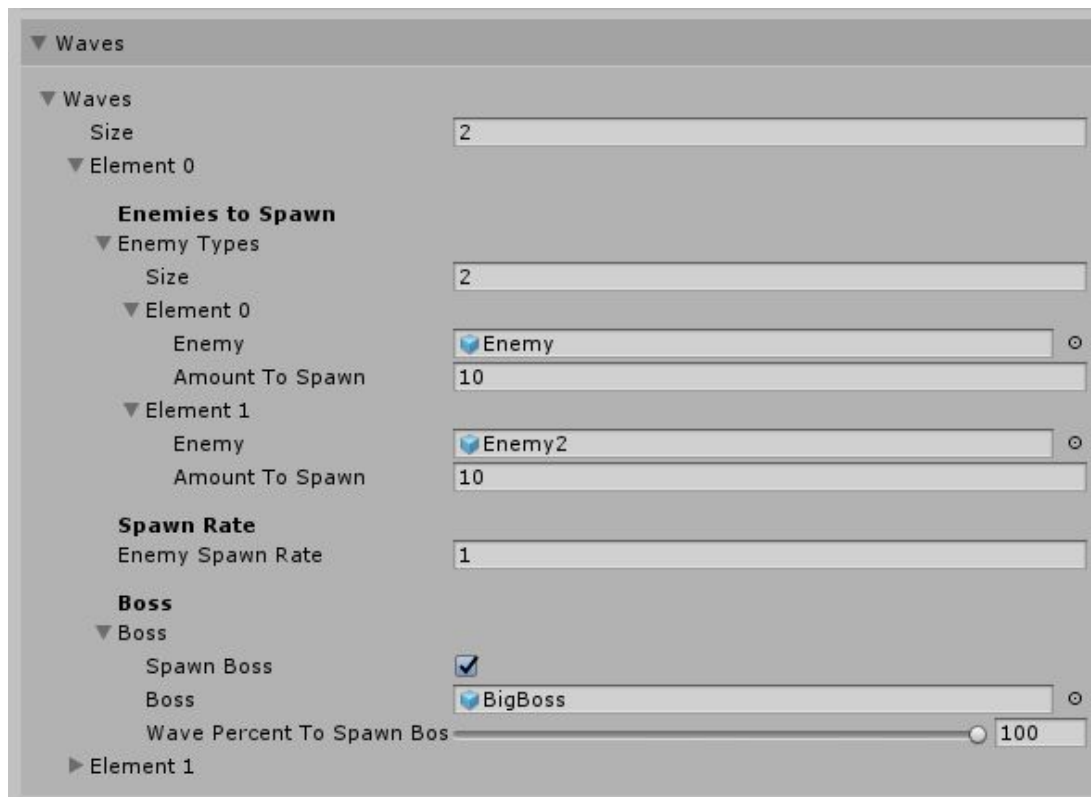
Scene Name: Name of the scene that contains the level.

Level Display Name: Name of the level that will be displayed on UI elements.

Locked at Start: Is the level locked at the start of the game? Does it require another level to be completed in order to unlock?

Level Needed to Complete to Unlock: Level Data file of level that is required to be completed in order to play this one.

More on the next page...



Waves: Each element is a new wave that will happen one after the other.

Enemy Types: Each wave can have multiple types of enemies, you can choose the prefab for each enemy and how many of them you want to spawn in that wave. Their order will be randomised upon playing.

Spawn Rate: How often will an enemy spawn in seconds, during the wave.

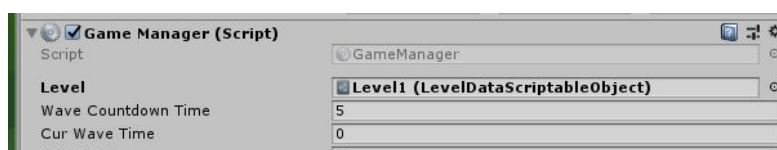
Boss: You can choose to have a boss spawn in a wave.

Spawn Boss: Will a boss spawn this round?

Boss: Boss prefab that will be spawned. Use the enemy template to create this one.

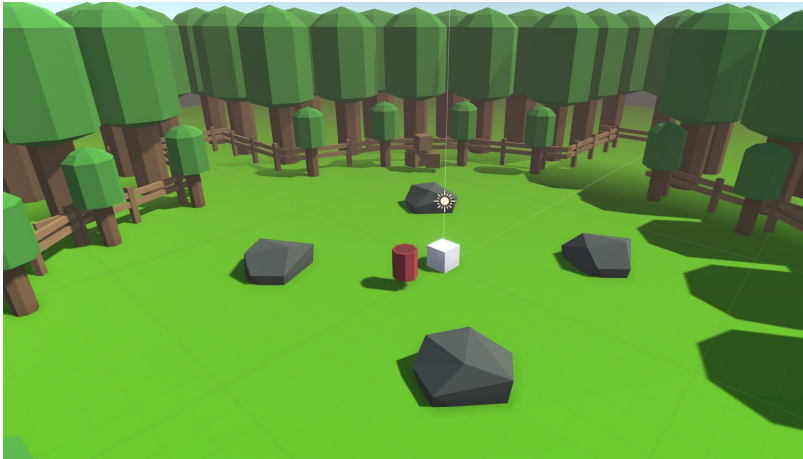
Wave Percent to Spawn Boss: During the process of spawning enemies, the boss will spawn this much (%) through the wave. 100% = end of wave. 0% = start of wave.

2. Now you need to create a scene for the new level.
3DWaveShooter/Prefabs/Templates has a *NewLevelTemplate* scene which has everything you need to create your level.



Make sure that the Game Manager script has your level in it.

3. Finally, go to the *Menu* scene, located at *3DWaveShooter/Scenes* and find the *_MenuManager* gameobject. Add your level to the array of existing levels in the Menu UI script. Now it will display in the menu and you should be good to go.



From there, it's pretty much designing your level.

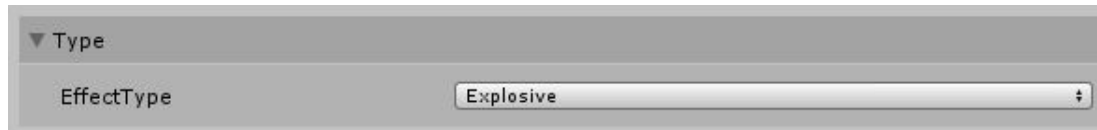
Make sure to move around the spawn points and pickup spawn points to their new areas.

Also make sure to generate your navmeshes!

Create an Effect

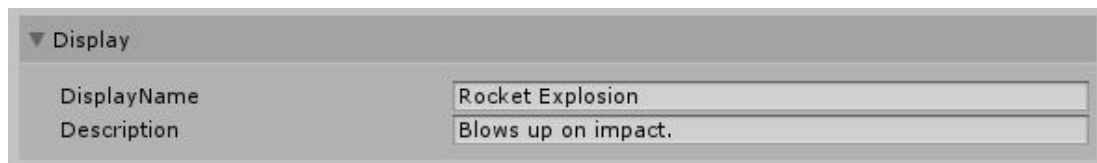
Effects are attributes you can apply to projectiles. When they hit an enemy that effect will take place. This can include damage over time, explosions, temp stat change like reducing move speed.

1. Effects are stored in the *3DWaveShooter/Effects* folder. There, right click and select Create > *Effect* to create a new *EffectScriptableObject*.



The screenshot shows the 'Type' section of the *EffectScriptableObject* inspector. It has a dropdown menu labeled 'EffectType' with the value 'Explosive' selected.

Effect Type: Type of effect to be applied. Explosive, Temp Stat Change, Damage Over Time.



The screenshot shows the 'Display' section of the *EffectScriptableObject* inspector. It contains two text input fields: 'DisplayName' with the value 'Rocket Explosion' and 'Description' with the value 'Blows up on impact.'

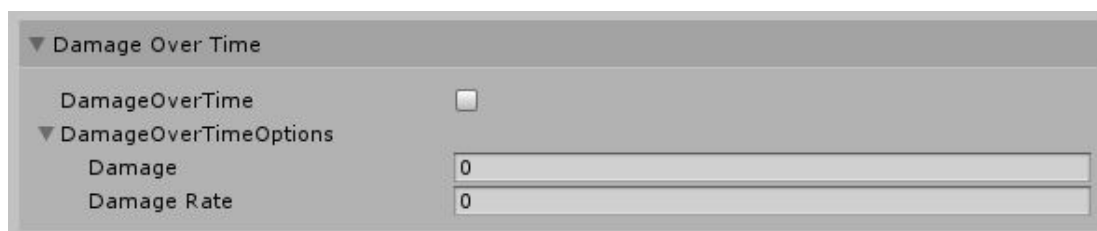
Display Name: Name displayed on UI elements (currently not in use).

Description: Short description about effect to display on UI elements (currently not in use).



The screenshot shows the 'Values' section of the *EffectScriptableObject* inspector. It contains a single numeric input field labeled 'Duration' with the value '0'.

Duration: The length of time in seconds the effect will take place for.



The screenshot shows the 'Damage Over Time' section of the *EffectScriptableObject* inspector. It includes a checkbox for 'DamageOverTime' which is currently unchecked. Below it is a collapsed section 'DamageOverTimeOptions' which, when expanded, shows two numeric input fields: 'Damage' with the value '0' and 'Damage Rate' with the value '0'.

If type is Damage Over Time...

Damage Over Time: Will this effect cause damage over time?

Damage: Amount of damage applied.

Damage Rate: Rate at which damage is applied in seconds.

▼ Explosive

Explosive ☒

▼ ExplosiveOptions

Explosive Damage 8

Explosive Range 10

Explosive Force 20

Explosive Damage Drop Off ☐

If type is Explosive...

Explosive: Will this effect cause an explosion upon impact?

Explosive Damage: Damage applied to all entities inside of the explosion.

Explosive Range: Range of explosion in units.

Explosive Force: Force applied outwards to all rigidbodies in the explosion range.

Explosive Damage Drop Off: Will the damage be less the further away an entity is from the center of the explosion?

▼ Stat Change

TempStatChange ☐

▼ TempStatChangeOptions

Stat To Change Move Speed

Stat Modifier 1

If type is Stat Change...

Temp Stat Change: Will this effect temporarily change the stat of the hit enemy?

Stat to Change: Stat that will be affected. Currently only move speed.

Stat Modifier: Multiplier added to the stat for the duration of the effect. E.g. 0.5 mod onto move speed will half the enemy's move speed.

► Projectile Object (Type = Projectile)

► Trail (Type = Raycast)

▼ Effects

ApplyEffects ☒

▼ EffectsToApply

Size 1

Element 0 TempSI

2. Now you just need to locate the Projectile you want to have this effect, and add it into the Effects tab.

Scripts

Camera

CameraEffects.cs

Manages camera effects such as screen shake.

CameraFollow.cs

Moves the camera to follow a target (player), with a rotation, offset, etc.

MenuCameraRotator.cs

Rotates around Vector3.zero on the Y axis, to add some flavour to the menu scene.

Enemy

Enemy.cs

Manages enemy health, death, stats, damage taken, etc.

EnemyAI.cs

Manages enemy movement, attacking, distances, etc.

Managers

AudioManager.cs

Manages all the audio in the game and all sounds that are played, go through it.

EnemySpawner.cs

Spawns enemies when told by the *GameManager* based of the level's enemy layout.

GameManager.cs

Manages the basic states of the game. When waves start, end, game over, etc.

ParticleManager.cs

Basically a container used to store particle prefabs that are commonly used.

PickupSpawner.cs

Manages the spawning of pickups.

Pool.cs

Global pooling system. Pool.Spawn() to instantiate, Pool.Destroy() to destroy.

ShopData.cs

Holds data for some shop items like health cost, ammo cost and player upgrades.

WeaponManager.cs

Contains the base weapons that other scripts can grab.

Mobile

MobileButton.cs

Shoot button UI control.

MobileControls.cs

Manages mobile controls, enable/disable, joystick and button.

MobileJoystick.cs

Movement/aim joystick ui control.

Other

Damageable.cs

World objects that can be interacted with and destroyed. Explosive barrels, crates, etc.

Effect.cs

Base class for an effect, which others use to determine what is affected.

Explosion.cs

Global class which all explosions go through to maintain consistency.

ExplosionSphere.cs

Explosion particle sphere which fades out. Purely visual.

MeshSetter.cs

Changes mesh material/color of enemies and player.

ParticleDestroy.cs

Destroys itself after a certain period of time. Could be used for things other than particles.

PickupHighlight.cs

Blue ring around pickups that moves. Purely visual.

ScreenFade.cs

Fades the screen in and out during scene changes.

Pickup

Pickup.cs

Base class for pickups. Weapons, health packs, ammo packs, etc.

Player

Player.cs

Base/core class for the player. Manages state, stats, damage taken, etc.

PlayerAttack.cs

Manages firing of weapons.

PlayerMovement.cs

Manages player movement.

Weapon.cs

Base class for a weapon.

ScriptableObjects

EffectScriptableObject.cs

Scriptable object class for an effect.

LevelDataScriptableObject.cs

Scriptable object class for a level, scene, waves, locked status, etc.

ProjectileScriptableObject.cs

Scriptable object class for a projectile.

WeaponScriptableObject.cs

Scriptable object class for a weapon.

UI

GameUI.cs

Manages all HUD UI, like ammo, health, wave counter.

MenuUI.cs

Manages all the UI in the menu scene. Choosing a level, etc.

ShopUI.cs

Manages all the UI in the shop.

UIButton.cs

Applied to every button. Hover effects, etc.

Weapon

Projectile.cs

Base class for physical projectiles. OnTriggerEnter, etc.

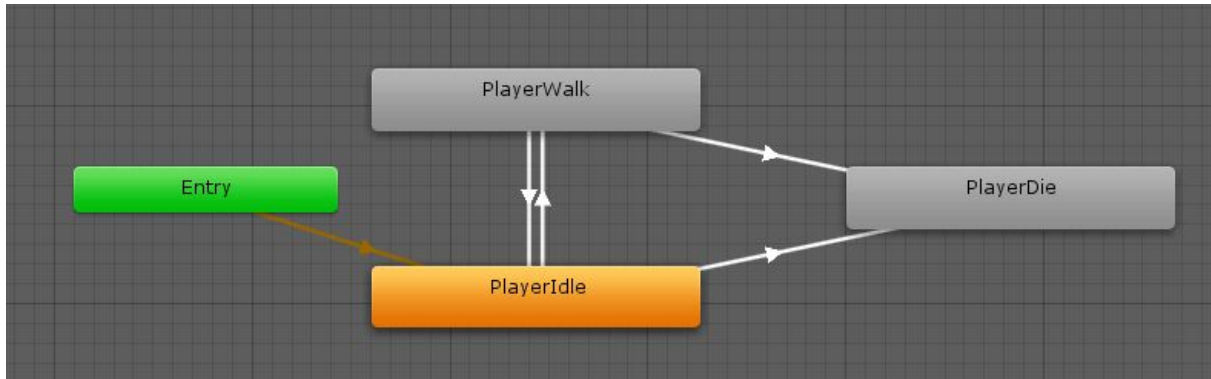
ProjectileTrailFade.cs

Raycast projectiles have a trail which fades after it's created. Purely visual.

Animators

At the moment, only the player and enemies have animations.

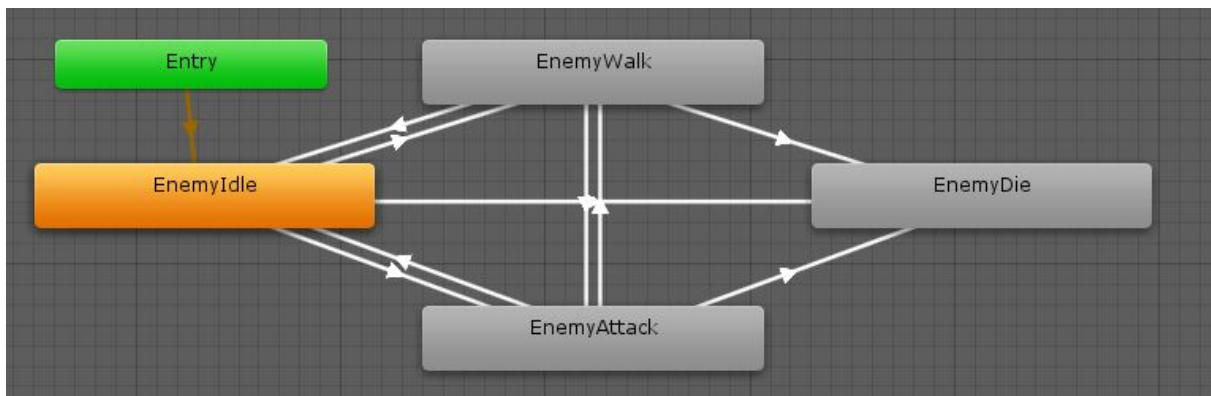
Player



Parameters:

- Moving (bool): If true, play PlayerWalk. Otherwise, play PlayerIdle.
- Die (trigger): On trigger, play PlayerDie.

Enemy



Parameters:

- Moving (bool): If true, play EnemyWalk. Otherwise, play EnemyIdle.
- Attack (trigger): On trigger, play EnemyAttack.
- Die (trigger): On trigger, play EnemyDie.

Level Required Prefabs

Each level requires a set of prefabs (located *3DWaveShooter/Prefabs/Requirements*) to make that level functional and consistent. This is the structure and layout of these 3 objects:

➤ **_GameManager**

Object that holds scripts which manage elements of the game.

➤ ParticleManager

Holds script which contains spawnable particle effects.

➤ Pool

Holds pooling script.

➤ ShopData

Holds script containing some shop prices.

➤ EnemySpawnPoints

Contains empty game objects which relate to spawn points.

➤ PickupSpawnPoints

Contains empty game objects which relate to spawn points.

➤ **CameraController**

Holds the camera (done this way so camera shake can work).

➤ Camera

■ Canvas

Holds all in game UI elements.

➤ **Player**

Player object.

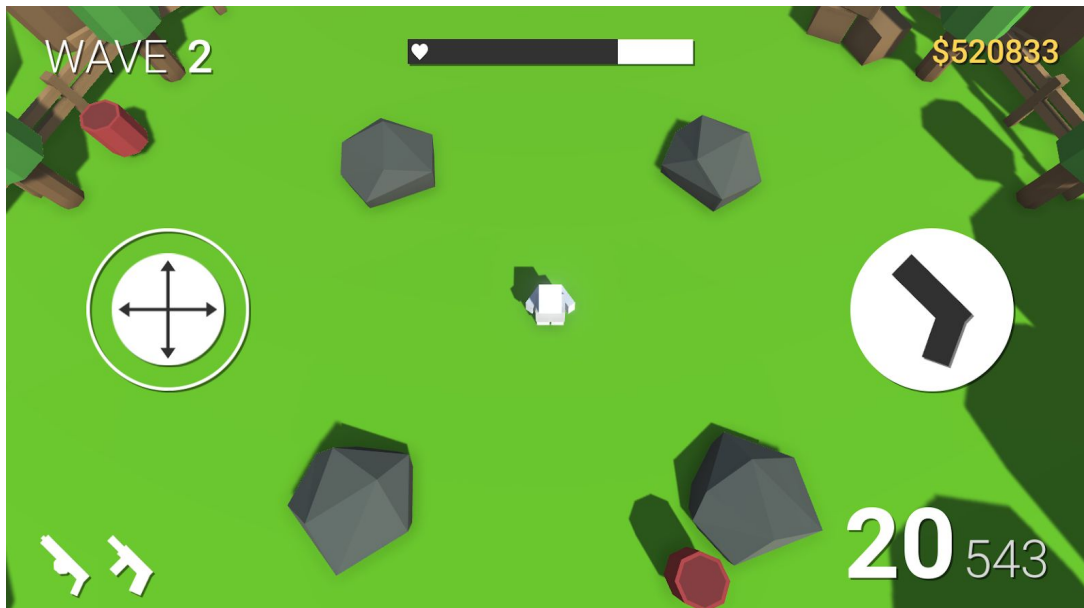
➤ PlayerModel

Contains the player model with an Animator component.

➤ WeaponHoldPos

Empty game object where weapon objects will spawn.

Mobile Support



This is what the mobile layout looks like (added movement joystick and shoot button).

- Move the joystick on the left to control the player's look direction and movement.
- Press the shoot button on the right to shoot your weapon.
- Press the weapon icons at the bottom to change your equipped weapon.

How to Enable Mobile Controls



1. First, go to the *CameraContainer* / *Camera* / *Canvas* object in your game scene. Find the *_MobileControls* object and enable it to enable the on-screen mobile controls.

2. Then in the *_GameManager* object in the game scene, make sure that the "Enable Mobile Controls" is set to true under the Mobile Controls script.



FAQ

I can't shoot my weapon sometimes / when hovering over UI elements.

When firing, it checks if your cursor is over UI elements. To fix this, make sure that all UI elements have "Raycast Target" unchecked. We do this because pressing the "Next Wave Button" causes you to shoot.

My new weapon isn't appearing in the shop.

Make sure the weapon is included in the "Weapon Manager" (on the _GameManager object) in all scenes. The shop only has a limited amount of weapon buttons. You can add more yet that may clunk things up so a scroll bar might need to be added. Make sure all new weapon buttons are included in the Shop UI script (on the Camera object).

My scene is dark when I choose a level on the menu.

This is a lighting issue when changing scenes inside of the editor. If it's bothering you, just open the scene you want to test. When you build the game, this problem won't occur.

None of the UI buttons are working in my scene.

Make sure you have an EventSystem object in your scene.

Contact

If you have any questions or suggestions for *3D Wave Shooter*, feel free to contact me at any of the following:

- Email: buckleydaniel101@gmail.com
- Twitter: [@dbuckleydev](https://twitter.com/dbuckleydev)

Thank you for purchasing the asset, and I hope you can use it well!

- *Daniel*

Changes

v1.1

- Added enemy and player model.
- Added enemy and player animations.
- Added MeshSetter.cs script.
- Updated documentation PDF.

v1.2

- Added mobile support.
- Added MobileControls.cs script.
- Added MobileJoystick.cs script.
- Added MobileButton.cs script.
- Added joystick icons.

v1.2.1

- Fixed spinning player issue on mobile.
- Added ScreenFade.cs script (fading between scenes).
- Changed MobileButton.cs script so shooting works properly now on mobile.
- Increased size of menu buttons.
- Included PostProcessing asset.

v1.2.2

- Added "Open Shop" button at the end of a round.
- Added Options screen to the menu.