Topdown Weapon Editor Documentation

If you need assistance please email me directly at: alansherba@gmail.com
If you use this editor and publish something with it let me know through my email or @AlanSherba on twitter, I want to see what you make from it.
Thank you, and enjoy the editor :)

This documentation may be out of date, live version here

Table of Contents

Script functions

Essential Scripts:

TweWeapon.cs is the core scriptable object script that controls the weapons.

TweProjectileControl.cs is the main script that controls the projectile after it has been spawned.

TweEnemyStatControl.cs is the default script used to apply damage from projectiles.

TweBurst Spawn.cs is used on burst weapons. The script is added to the weapon firing transform and counts down to spawn each burst projectile.

TweCountdownDestroy.cs is used to delay the final destruction of the projectile objects till after the projectiles trail has finished.

TweWeaponObjectInspector.cs is the custom inspector for the weapon objects that keep everything clean.

Non-Essential Scripts:

TweMouseAim.cs Aims its transform towards the mouse

TwePlayerCombat.cs Shoots the weapon using the set aimpoint transform

TwePlayerMovement.cs Basic movement script, uses basic character controller

TweDissolveControl.cs Creates the dissolve effect on certain particle systems. It manipulates a value in the dissolve shader.

TweParticleDestroy.cs Automatically destroys a particle system once it has finished, this is on all particle systems by default.

Basic Setup and Utilization

1. Create a new weapon object by creating a new asset and selecting “Weapon”
2. Customize the parameters of the new weapon. The most important to change initially will be the collision tags in the last tab. More detailed explanation of each parameter will be in the full documentation.
3. If not using the provided PlayerCombat.cs, inside your combat script you must create a Weapon variable, then use `wepName.FireWeapon(transform)` to fire the weapon how you see fit on your own script.
4. If using the PlayerCombat.cs you only have to drag your new weapon into the public Weapon variable slot.
5. INPUT. The provided PlayerCombat.cs uses the “Fire1” input axis. If you have renamed this you will need to adjust accordingly

The weapon editor has many options and is very versatile, The provided examples given with the editor are extremely basic and barely scratch the surface of what’s possible. For more information view the full documentation Below.

Main

Weapon Name Sets the name of the projectile objects that are instantiated

Tag Sets the tag of the projectile, this has no use internally be default

Layer Sets the layer index of the projectile, this has no use internally by default

Damage The damage passed through to the hit objects with a EnemyStatControl.cs. Uses the damage tags under the collision section

Destroy on Hit When enabled the projectile will destroy when it damages an enemy. Use this if you don't want to mix break and damage together.

Fire Rate The time between shots fired. This does not have any inherent function and must be implemented inside your combat script. See the PlayerCombat.cs for an example.

Bullets How many bullets are fired per shot

Lifetime How long the bullets lasts before automatically being destroyed. Triggers fizzle particle on time-out

Screen Shake Screen shake for each weapon. This does not have any inherent function and must be implemented inside your combat script. See PlayerCombat.cs for an example

Inaccuracy The Inaccuracy of each axis.

Random Velocity Multiplier The max and min random multiplyer of the velocity on each projectile. Good for creating weapons like a shotgun.

Even Innacuracy When selected inaccuracy will be evenly distributed, Think like a triple machinegun

Bounce when selected projectile will bounce off objects with the proper bounce tags under the collision tag.

Bounces how many times the projectile will bounce before breaking on hit

Bounce Friction friction force applied to the velocity of the projectile when it bounces

Random Bounce Angle Random rotation applied to the projectile when it bounces

Burst when enabled turns the weapon into a burst weapon using the BurstSpawn.cs script.

Burst Fire Rate Time between each bullet within the burst

Bullets Per Burst How many bullets are in a burst

Reverse Even Sweep with burst enabled and even innacuracy enabled, the burst will sweep along the innacuracy provided. This will reverse the direction of the sweep.

Show Original Inspector shows the full default inspector for the object. Only used for debug purposes.

Motion

Initial Forward Speed Initial forward speed for the projectile

Forward Acceleration Forward acceleration for the projectile, negative values can be used.

Limit Speed Enables the following two parameters

Max Forward Speed Maximum forward speed of the projectile

Min Forward Speed Minimum forward speed of the projectile

Rotational Speed the constant rotation speed for the projectile.

Random Rotational Speed When enabled, rotational speed becomes a random value functioning similar to Innacuracy

Angular Drag Drag force applied to the projectiles rotational speed

Angular Turbulence Adds random rotational speed overtime.

Projectile Stick When enabled projectiles will stick to objects using the following parameters

Stop Stick On Bounce When enabled projectiles will stop sticking once they have bounced. This helps prevent particles glitching when the bounce gives them an x rotation

Projectile Height the height above the ground in units that the projectile sticks to

Interpolate Stick if enabled projectiles smoothly stick scaling with the following stick speed value

Stick Speed How fast the projectile will lerp to its desired location

Projectile Stick Tags The object tags that the projectile sticks to

Enable Homing toggles homing for projectiles

Check for target on start only When selected the projectile only checks for a homing target on start

Initial Check Range Check range for the first check on start

Interval Check Range Check range for every check after the first

Check Interval How often the projectile checks for a target. Setting this to very low values will have a performance impact.

Homing Speed Speed at which the projectile turns toward the target

Homing Acceleration acceleration of homing speed.

Homing Tags The object tags that the projectile homes to

Show Debug Gizmo Shows a gizmo in the editor when enabled

Destroy on Zero Speed When enabled the projectile will die when its velocity is <= zero

Gravity constant downward motion applied to the projectile.

Look & Feel

Mesh The mesh of the projectile

Material The material for the mesh of the projectile

Receive Shadows Modifies the receive shadows variable in the mesh renderer of the projectile

Projectile Scale Scale applied to the projectile game object

Fire Sound Audio Clip played when the weapon is fired

Fire Particle Particle spawned when the weapon is fired

Particle Trail Adds a particle system trail to the projectile. Make sure the particle system has looping enabled and simulates in world space.

Unity Trail Renderer Enables the use of a unity trail renderer on the projectile weapon.

Material Material of the Trail

Time Defines the length of the trail, measured in seconds

Min Vertex Distance The minimum distance between anchor points of the trail

Autodestruct Enable this to destroy the GameObject once it has been idle for Time seconds. Be careful using this option, you will want this DISABLED in most cases

Emitting Toggles whether the trail is emitting or not

Width A width value and a curve to control the width of your trail at various points between its start and end. The curve is applied from the beginning to the end of the trail, and sampled at each vertex. The overall width of the curve is controlled by the width value

Color A gradient to control the color of the trail along its length

Alignment Set to View to make the Trail face the camera, or Local to align it based on the orientation of its Transform component

Texture Mode Control how the Texture is applied to the Trail. Use Stretch to apply the Texture map along the entire length of the trail, or use Wrap to repeat the Texture along the length of the Trail. Use the Tilingparameters in the Material to control the repeat rate

_____ Sound, Particle, and Spawn The audio clip, particle system, and game objects that get spawned when the projectile damages, breaks, bounces, or fizzles

*__Fizzle__ happens when a projectile reaches its max life, or reaches 0 speed with destroy on zero speed enabled

Collision

Hitbox Radius The Radius of the projectiles hitbox

Show Collision Debug Shows collision gizmos

Damage, Break, and Bounce Tags The tags used in collision detection, if the object should damage, break, or bounce when colliding.

Pooling

Use Object Pooling Toggles object pooling

Start Pool Size The initial pool size

Pool Auto Expansion How much to expand the pool by each time the pool reaches its limit. HIGHLY recommend keeping this value greater than bullets

Custom Player Control Adaptation

If you want to throw away the provided player/combat controls and adapt it to your own that's great, follow the checklist below to make sure you keep all functionality.

- ❏ Inside your playerCombat create a reference to a weapon
- ❏ Determine how you are handling camera shake,
    - ❏ Somewhere you need a script with a method that takes a float in and shakes the camera.
    - ❏ Inside TweWeapon.cs find Line 121 and change the class name of cameraControlReference to your class.
    - ❏ Do the same to the class name of "cc" on line 124
    - ❏ Inside TweWeapon.cs find the last line in the ScreenShake() method (its the very last line of the script), and change-cameraControlReference.Screenshake(screenShake); to use your method name.
    - ❏ You must now give the weapon the camera control reference during play, you can use the second argument in TweWeapon.FireWeapon() like is done in the provided playerCombat script
- ❏ (Optional) call WepName.CreatePool() somewhere like start to prevent it from happening in the middle of gameplay. Ignore if not using pooling

- ❏ Create a transform that points towards where you want bullets to fire (the Z axis is forward) I refer to this as an “AimPoint”
- ❏ Fire the weapon using wepName.FireWeapon(AimPoint, Camera Control)
    - ❏ You must create your own fireRate controller, check TwePlayerCombat.cs for an example

It’s mostly very simple. Adapting the camera shake can be confusing, If you are having problems or don’t know why it’s done like this feel free to contact me and I will help you out.

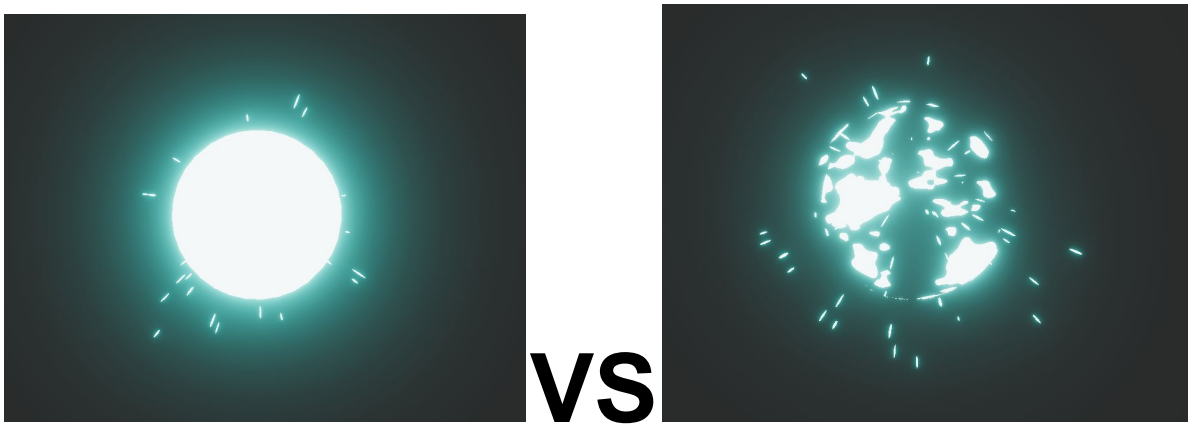
Particles

The package comes with a lot of particle effects, this section will explain the differences between the different types and things you should keep in mind when using them.

Hits vs **Regular**
These are impact effects that are split into two categories. **Hits** being effects with a 180d arc and **Regular** being effects with a 360d arc. Regular effects can be used in pretty much any situation, but hits are very useful when you want to convey a specific direction in the effect like sparks on a wall, or a muzzle flash.

Trails
These are looping systems that simulate in world space. See the HomingRocketLauncher weapon

Particles Using Dissolve Shader
Some particle effects use a custom dissolve shader to give them an explosion effect. The shader is controlled by the TweDissolveControl.cs script which is attached to them. Since these effects rely on this script they look different in edit mode compared to play mode. See below:



Edit Mode **Play Mode**