CINEMACHINE

unity

USER MANUAL V1.1

THANK YOU

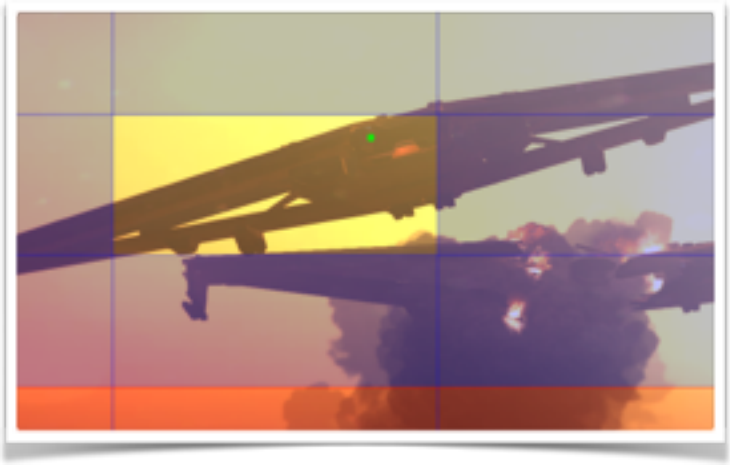
Thanks for purchasing Cinemachine for Unity! We hope it makes your project look amazing

UNIFIED CAMERA SOLUTION

Cinemachine is built to be an entirely unified camera system bridging gameplay, cutscenes, from fully procedural cameras to entirely canned sequences and everything in between.

PROVEN HISTORY

We've been designing camera systems for almost 20 years and have shipped millions of AAA titles across numerous genres. The Cinemachine team has an award winning cinematographer and a senior engineer with heavy math skills. Also, we love this stuff to bits.



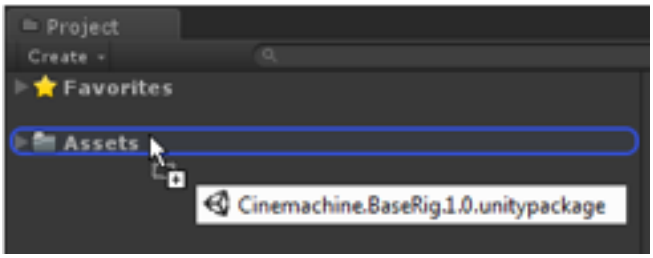
CREATE | INNOVATE | ITERATE

Cinemachine is a suite of 'smart' procedural modules which allow you to define the shot and they'll dynamically follow your direction. Setup shots which track and compose motion in real-time, like AI camera operators. The procedural nature makes them bug-resistant as they always work to make the shot based on your direction. That's great for gameplay, but they're also amazingly fast for cutscenes. Change an animation, a vehicle speed, ground terrain - whatever - and Cinemachine will dynamically make the shot. You can use really telephoto lenses and not have to update the cutscene if things change

TABLE OF CONTENTS

INSTALLING CINEMACHINE

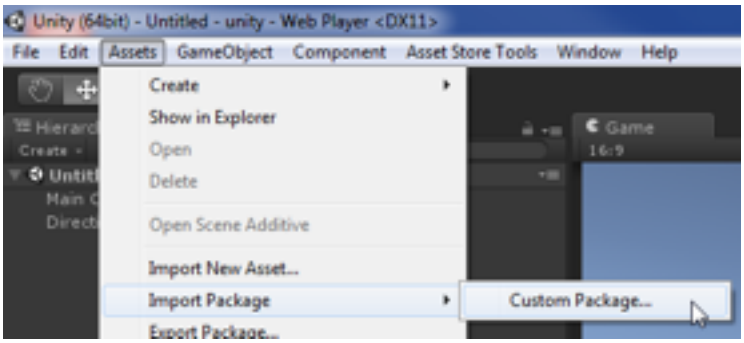
Installing Cinemachine is easy.



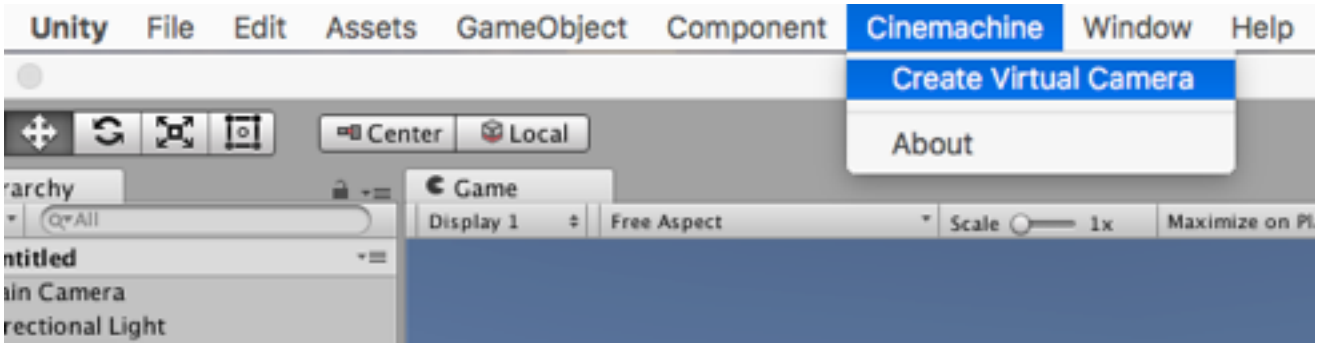
1. Drag the Cinemachine.BaseRig.unitypackage onto your assets folder in Unity.

   OR

   Install it directly:
   Assets->Import Package ->Custom Package… and point to the Cinemachine.BaseRig.unitypackage



2. Setup the Cinemachine Virtual Camera environment
   Cinemachine-> Create Virtual Camera. This is also the way to create additional Cinemachine shots.



3. That's it! You now have Cinemachine installed on your machine.

You will notice that two new components were created: *CinemachineVirtualCamera* which is a single camera or shot, and *AUTOGEN_CinemachineRuntime* which is the Cinemachine core system and where the camera blending information is defined.

There's more info below on how to use them and don't forget to checkout the website at www.cinemachineimagery.com/documentation, and www.cinemachineimagery.com/tutorials

VIRTUAL CAMERA OVERVIEW

This is the property panel for a Cinemachine Virtual Camera. Each virtual camera is a 'shot', or a single camera. You can animate them directly or blend two together to create camera moves.

The reason they are 'virtual' cameras is to allow for blending, camera selection based on priorities and the ability to have Cinemachine hold multiple cameras in memory with the final product being presented to the current active Unity camera(s) which are rendering.

CinemachineVirtualCamera contains:

Priority: The priority of this shot. Equal or Higher value camera priorities compared to the current camera priority will be activated. The way the same/higher priority camera blends into the previous one is defined under the asset called **AUTOGEN_CinemachineRuntime** (more on this below)**.** The default is a 2 second smooth blend. You can set up as many custom blends between cameras as you like. To turn a blend into a cut, delete the right key in the curve (again, see below on the Blending section).

Priority and Virtual Cameras allow for powerful yet easy to set up camera state machine systems where cameras are called based on trigger volumes, animations, health states, etc. Important cameras will trigger over lower priority ones. Build complex multi-camera setups easily.

Noise: A procedural Perlin multi-layered noise system for handheld behaviours, shakes, and vibrations.

Composer: A procedural way to track and compose any object using screen-based compositional controls. Bug-proof your cinematics, have your gameplay cameras follow the action with weight, put the subject exactly where you want it on screen.

Transposer: A way to 'mount' your camera to any object with controls for how it follows the motion

OK let's get into the details!

BASIC CINEMACHINE SETUP

Auto Add To Priority Stack This keep the camera alive and in the priority stack able to be turned on and blended (or cut) to if it's a same or higher priority. If it's set to false, the camera needs to be added to runtime programmatically. Just leave it on in most cases.

Priority the priority setting for that shot. Equal or higher priority cameras will be blended to. An example, say you're currently running a priority 2 camera. If a priority 1 camera is called it will be ignored. If a priority 2 or higher camera is called, it will blend to that new camera using the Default Blend or a specific blend if you've defined that in the Blend Settings set under the *AUTOGEN_CinemachineRuntime* object. This is really powerful in state machine type setups. You can have any number of cameras running and say you've defined a camera for a victory sequence or a specific location in the world, you can have that camera either trump whatever is going on or be ignored.

Composer Camera target: drag any game object into this area to define what the camera will look at. For more info see the Composer section below.

Transposer Camera Target: drag any game object into this area to define what the camera will follow / be 'mounted' to. For more info see the Transposer section below.

Offsets: This is where you can input animation from Unity Animator, or plugins like the fantastic **Cinema Director**, or with the upcoming Unity Sequencer, which will be in Unity v5.5

Noise Amplitude / Speed Scalar: This allows you to scale the camera noise amount and frequency. You can programmatically input values here for nearby explosions, or animate a curve and pipe that in to make the camera shake more when a tank drives by, etc. Tons of uses!

Noise Settings this is the asset which contains all the Noise settings. See Noise below.

Composer Settings this is the asset which contains all the Composer settings. See Composer below

Transposer Settings this is the asset which contains all the Transposer settings. See Transposer below

Show Camera Guides this draws an overlay in screen-space which shows you the current composer settings, where you'd like the target to be composed on the screen and the Soft and Hard composer controls. See Composer below.

COMPOSER

Targets : Composer Camera Target

Common Composer targets: include bones on a character like the upper spine or head bone, vehicles, or dummy objects which are controlled or animated programmatically. See 'Advanced Setups'.

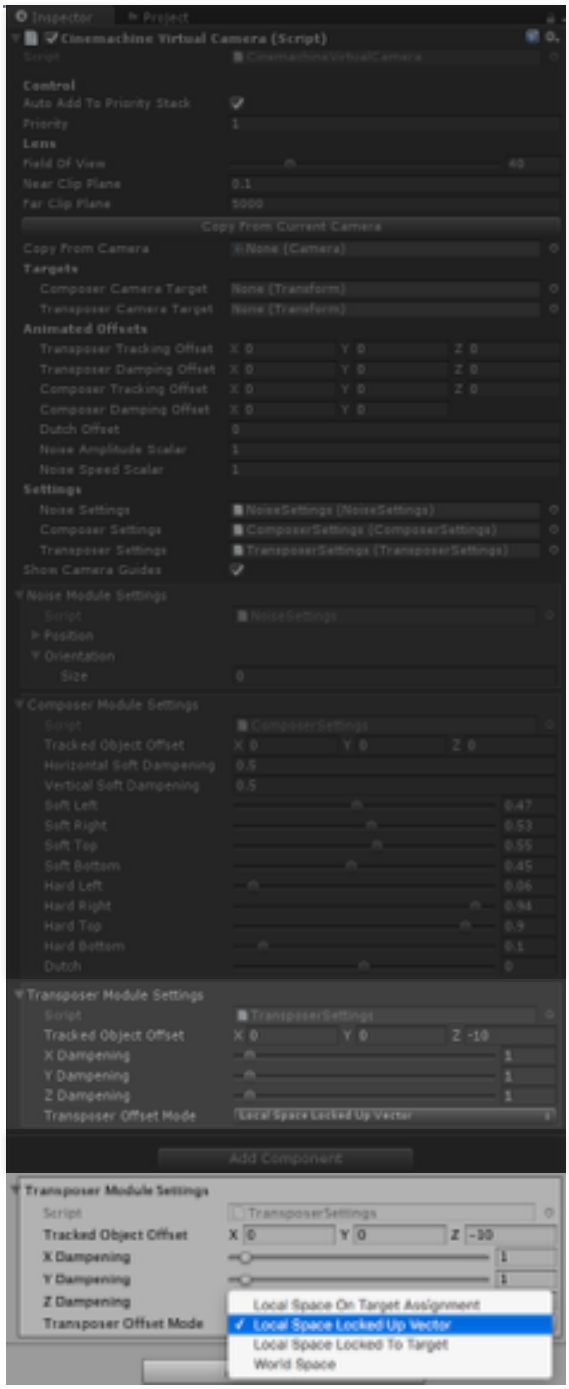
Now that you're tracking something, you need to define where you'd like it to be on the screen and how aggressively you'd like to camera to track it. The degree of lag, or how tightly the camera follows the subject is defined by the two controls:

Horizontal/Vertical Soft Dampening: Setting these values to zero means the camera will hard track the subject and the blue regions above will act as rigid barriers locking the camera movement to keep the target inside those values. Setting the larger will allow the target to 'squish' into the blue regions, giving you some really nice camera weight and lag. This is what real camera operators do!. The vertical and horizontal values are separated so you can let the camera squish more left to right or follow tighter up and down, etc.

Soft/Hard Left/Right: Tune these to define where you want the subject to be onscreen. These controls are *incredibly* versatile. The blue areas are the 'squishy' areas based on how much horizontal/vertical dampening you have, and the red areas define the 'no pass' limits where the camera will always track the subject. Opening up the soft areas to create a 'dead zone' in the middle allows you to make areas of the screen immune to target motion, handy for things like animation cycles where you don't want the camera to track the target if it moves just a little.

Dutch: Camera tilt, or z-roll. Go easy on it !

TRANSPOSER



Transposer is a component which mounts a camera to any object. It has a number of advantages over just putting the camera under the object you want to have the camera follow. Hood Cam, Door Cam, POV Cam, Missile Cam - if you want to mount the camera to something, use Transposer.

Using Transposer for your camera body motion has advantages: The position tuning is kept even if the game is running. Tune a regular camera in game mode and the position is lost once you quit, not with Transposer. Tune it up while the game is running and you're done. Add dampening to the camera - if your camera follows something, the dampening values can give you some 'weight' to the camera so they feel smoother. Organization: Put all your cameras in one place, instead of having them hidden under all sorts of assets in your project - you pick their targets VS put the camera under them in the hierarchy.

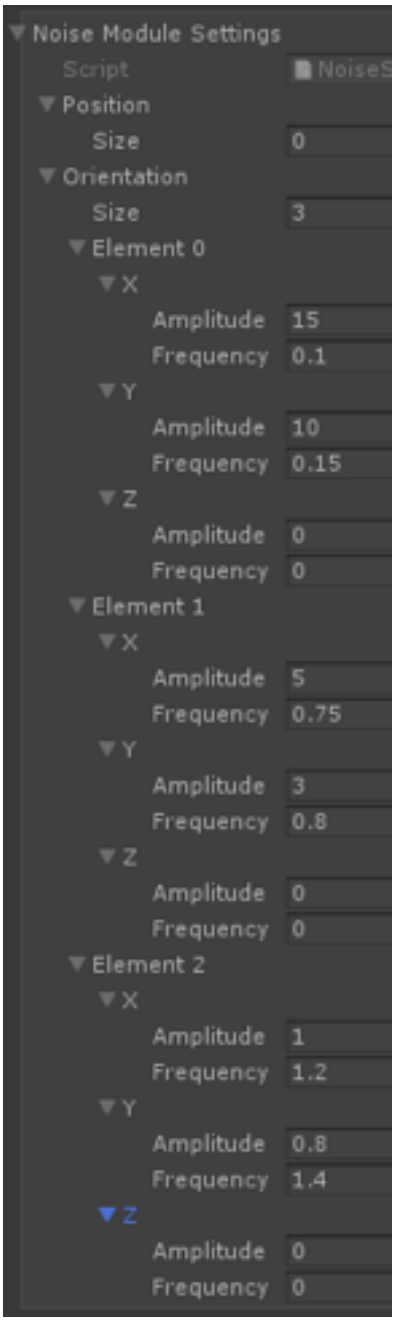
Tracked Object Offset: The camera will go to the centre of the object you're targeting in the Transposer Camera Target, so if you want the camera to be further behind, put in some offsets. We default at -10 so the camera is behind whatever you're targeting.

Dampening: Per channel dampening which will cause the camera to lag behind the target.

Transposer Offset Mode: There's options in the relationship between the camera and the target object.
Local Space On Target Assignment mounts the camera relative to the local coordinate system at the time the camera is initialized.
Local Space Locked Up Vector is the same as the above but it keeps the camera pointing up. Handy if your target flips over.
Local Space Locked To Target is full local coordinates for the camera offset space. **World Space** offsets move the camera relative to the world no matter what direction the target object is facing. These different modes do incredibly different things, so try them out and one should work well for whatever your requirements are.

The 'Door Cam' or 'Hood Cam' ideas would use **Local Space Locked To Target.**

NOISE



The Noise module is a multi-layered Perlin noise function which is applied after the Composer and adds additional transforms. It has controls for Position and Orientation. You can add as many layers as you want by increasing the Size value.

Procedural noise is a complex thing to make look real. Convincing hand-held motion is a mixture of low, medium and high frequency wobbles which together combine to create something believable.

Position / Orientation Size: This is how many noise functions you'd like to blend together for the Position or Orientation or both. Mix at least 3 Orientation channels together for some realistic hand-held motion.

Amplitude defines the amount of noise in degrees. Wider lenses will need larger degree values in order to 'see' the shake. Telephoto lenses use smaller values as that small setting appears amplified through narrower FOV lenses.

Frequency defines the speed of the noise in Hz. Typically a 'low' frequency value might be around 0.1. Consider that your game is running at 30 or 60hz, so settings higher than that will be 'on the other side' of the Nyquest frequency meaning that they will not be directly tracked. A setting of 100 will be higher than what the camera can 'follow' as your game is only running at say 60hz. It can look kind of choppy since the camera can't track something which is sampling faster than what the game is running at. It can also look kind of cool, but rarely. Experiment. Typically, for most hand-held setups, the low is around 0.1-0.5, the mid maybe .8-1.5 and the high around 3-4. That's 3-4 shakes back and forth per second.

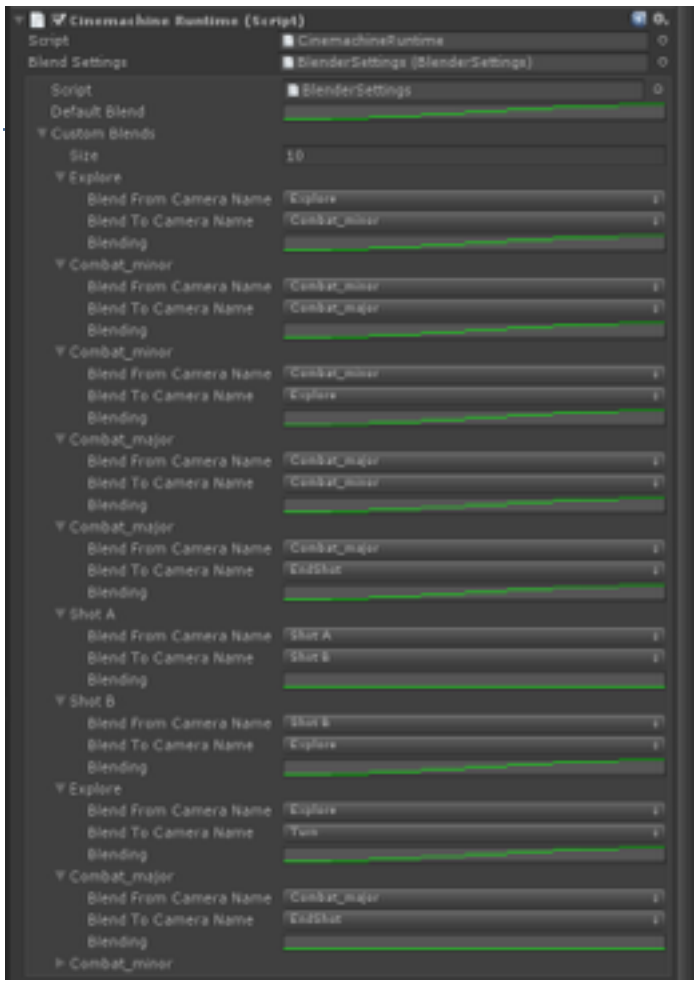
The most convincing camera shakes are typically done with Orientation noise as that's where the camera is aiming. Handheld camera operators tend shake more rotationally than they do positionally, but of course feel free to mix in some Position noise, just remember it's probably best to start with the Orientation.

We've included a number of presets to get you going, under Assets/Cinemachine/Noise and of course you can add as many of your own as you wish, just right click in the Asset window Create->Cinemachine->Noise, and drag that asset into the Noise Settings window under that VirtualCamera.

You can also animate the Noise through the Noise Amplitude Scalar, and Noise Speed Scalar to ramp the effect up and down.

BLENDER

AUTOGEN_CinemachineRuntime



AUTOGEN_CinemachineRuntime contains all the settings for how any camera blends into the next. A **Priority** is on each Cinemachine Virtual Camera and if a same or higher camera is triggered, Cinemachine will check here to see how the cameras will blend.

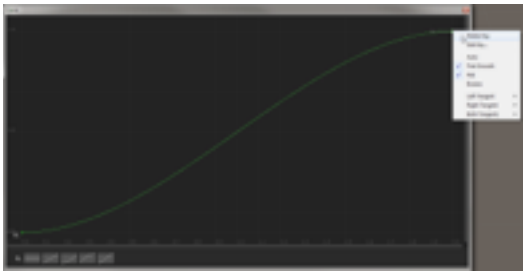
If a specific camera to camera blend is defined, it will use that one. If not ***Cinemachine will use the default blend*** which is a smooth 2 second blend.

Complex, powerful camera state machine setups can be quickly and easily created here.

Multi-camera local-motion system setup a number of cameras and have them triggered during different animations. Have the Run camera blend from the Walk camera in 2 seconds and blend it back from the Run to the Walk over 4 seconds.

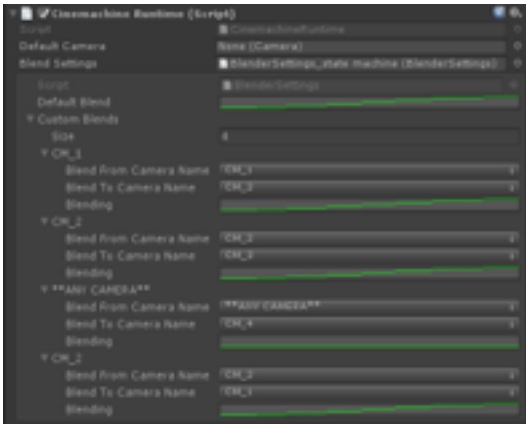
Use trigger volumes in your game to call different cameras. Call the death sequence camera anytime your hero dies, and have it set to a higher priority so it will blend to that camera no matter what else is happening.

The default 2 second smooth blend
Delete the key to make the blend a



With Cinemachine Game Rig, you can make any of the cameras 'Free-Look' or 'Cloud' cameras, which work amazingly well for 3rd person Action Adventure games or for replays and procedural eSports cinematic

If you're doing a lot of cutscenes, you will probably want to set the default blend to be a cut. Just delete a key to make any blend a cut.



Blender supports wildcards called **ANY CAMERA**, which you can use on the input or output of blends so that no matter what camera you're in, you only have to specify one blend to a particular shot if you always want to blend into that shot the same way.

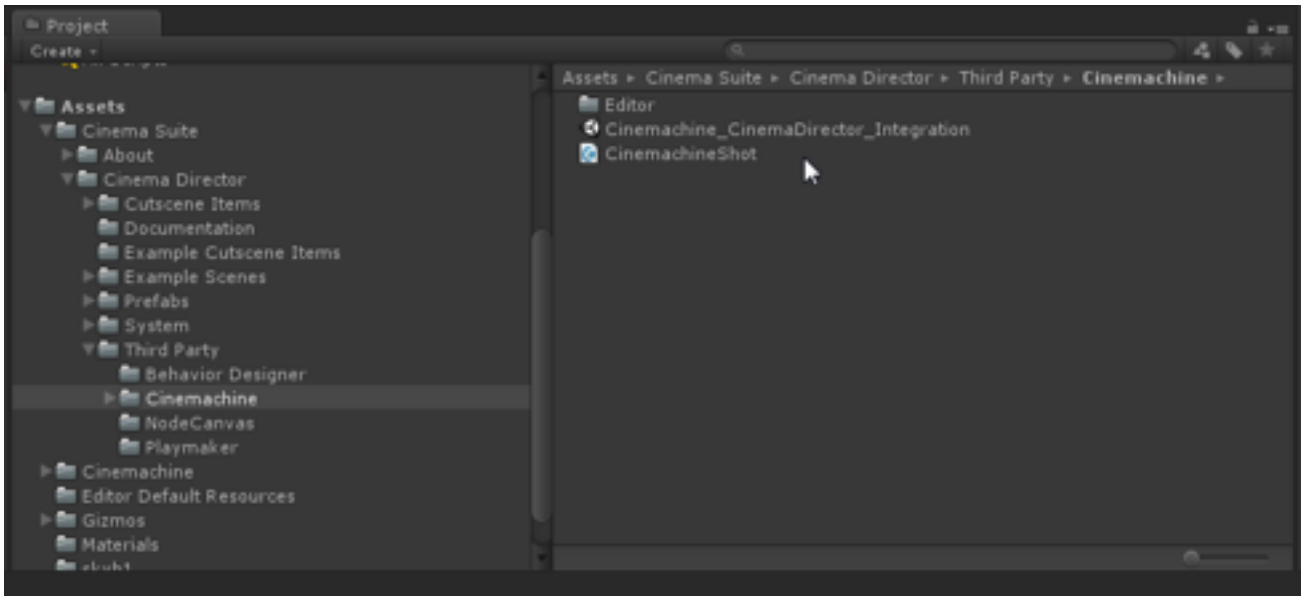
Blending priority: #1 explicit to explicit, #2 from wildcard to explicit, #3 explicit to wildcard, #4 default blend.

WORKING WITH CINEMA DIRECTOR

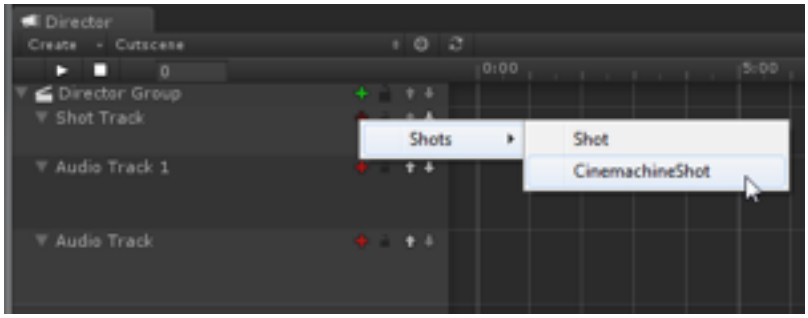
Cinemachine and Cinema Director work fantastically well together but **you need to be on the latest versions**

Cinemachine v1.1.1 or greater
Cinema Director v1.4.5.1 or greater
Cinema Suite v1.2.1.1 or greater

Once you've installed both Cinemachine and Cinema Director, go to the Cinema Director folder in your project assets: *Assets/Cinema Suite/Cinema Director/Third Party/Cinemachine*
Double click the: **Cinemachine_CinemaDirector_Integration**



Now when you make a 'Shot' in Cinema Director you have two options: A regular camera shot and a CinemachineShot.



Note! Cinemachine uses a default 2 second blend between cameras. This will cause your 'cuts' to be blends when using Cinema Director, which is probably not what you want. To make the default blend in Cinemachine be a cut instead of a blend, inspect **AUTOGEN_CinemachineRuntime** open the default blend curve and delete the last key making the curve into a single key - which is a cut. **See Page 9** for more info.

ADVANCED SETUPS I

There are so many different way to use Cinemachine. Make sure to check out our example scenes and tutorial videos for more ideas. Here are a couple of high-level ideas which might work great for your project.

'Marionette' Cinemachine. Sure you can target the hero character with the Composer and that's probably the perfect way to do your cinematic setups, but there's a lot of other ways to use Composer! Create a dummy object, say a small sphere and call it 'Camera Target'. Have the game position this object based on gameplay variables like 'the average position of all the enemies onscreen'. The sphere will now erratically bounce around based on the # enemies but that's ok because the Composer has the dampening controls. Blend between different cameras based on gameplay events and you can now zoom out when there's more guys or zoom in when there's less and you'll have complete control of where you want them onscreen.

Blending cameras with all the same properties except for zoom / Composer composition allows for easy to broadcast events to trigger cameras giving you exactly the control you want.

Pointing Cinemachine to dummy objects which are driven by gameplay variables provides incredible behaviour options. You won't need to filter that math or do any complicated smoothing because Cinemachine gives you blending/dampening math which has been fine tuned to feel like a weighted camera. On one project we had Composer look at an object which was positioned at the centre of the enemies and another object which was raised off the ground in Y based on the number of enemies.

When it was just you and one bad guy, the camera was low and targeted the middle position between you and them. When a bunch more bad guys showed up, the camera raised off the ground and targeted the middle of everyone.

State Machine local-motion camera system. Using Cinemachine Base Rig with GameRig's 'Free-Look' it's possible to set up world-class 3rd person action adventure camera systems. Create a unique Free-Look camera for events like this: Stand / Walk, Run, Sprint, etc. Link the appropriate Cinemachine Free-Look camera up to that animation state, so when the animation state is triggered it turns on that Cinemachine camera. There's a number of ways to do this in Unity or with different Asset Store plugins - we can't help you there! - but the idea is to simply turn on the right Free-Look camera for each animation state.

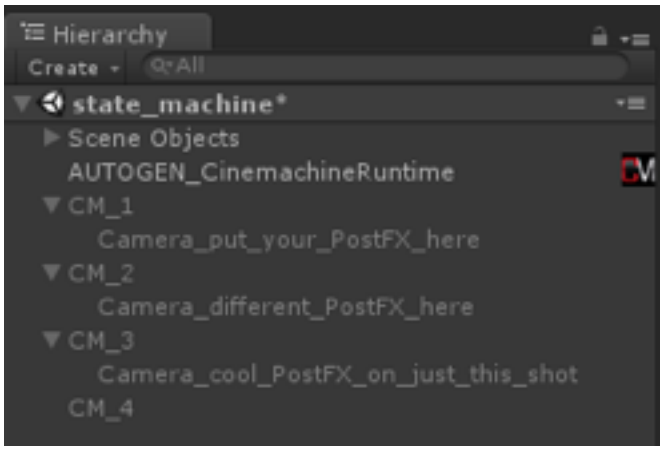
Your 'Sprint' Free-Look gets in closer, the lens is wider and there is a ton of handheld noise. Boom, you now have a 'Roadie Run' camera.

On a previous camera system we designed which is currently in use in Frostbite, we had over 40 Free-Look cameras for a character's local-motion and combat system, all setup so quickly and all tuneable in real-time.

ADVANCED SETUPS 2

Post FX switching / custom components on a camera.

It's easy to set custom components on your shots even though Cinemachine uses virtual cameras. Here's how it works: If there's a real camera under a Cinemachine virtual camera, it will use whatever postFX are on that real camera.



CM_1 is a virtual camera and *Camera_put_your_PostFX_here* is an actual Unity camera underneath.

Cinemachine will detect any real Unity camera underneath on of its virtual cameras and it will inherit those postFX for just that shot.

More advanced setups will be shared on our website in the Tutorials section

http://www.cinemachineimagery.com/tutorials/

Feel free to reach out with any questions you may have on the Unity Forums, we are here to help, thanks so much.

- Adam Myhill, Designer, Cinemachine