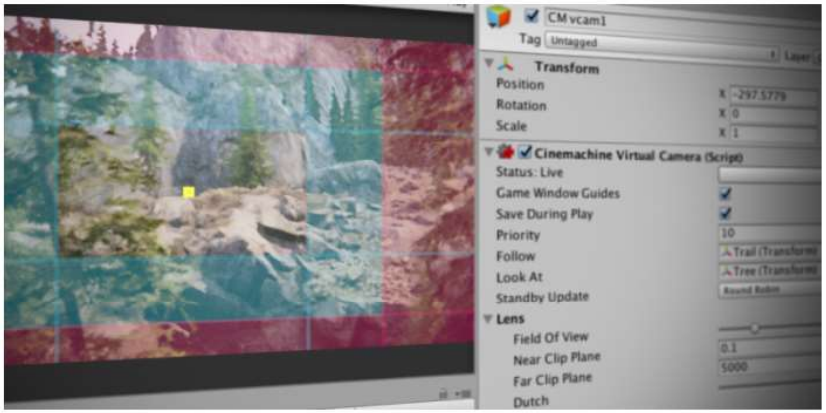
Cinemachine

About Cinemachine



Cinemachine is a suite of modules for operating the Unity camera. Cinemachine solves the complex mathematics and logic of tracking targets, composing, blending, and cutting between shots. It is designed to significantly reduce the number of time-consuming manual manipulations and script revisions that take place during development.

The procedural nature of these modules makes Cinemachine bug-resistant. When you make adjustments—for example, change an animation, vehicle speed, terrain, or other GameObjects in your Scene—Cinemachine dynamically adjusts its behavior to make the best shot. There is no need, for example, to re-write camera scripts just because a character turns left instead of right.

Cinemachine works in real time across all genres including FPS, third person, 2D, side-scroller, top down, and RTS. It supports as many shots in your Scene as you need. Its modular system lets you compose sophisticated behaviors.

Cinemachine works well with other Unity tools, acting as a powerful complement to Timeline, animation, and post-processing assets. Create your own extensions or integrate it with your custom camera scripts.

Installing Cinemachine

Cinemachine is a free package, available for any project. You install Cinemachine like any other package.

After you install Cinemachine, a new *Cinemachine* folder appears in the Gizmos folder of your Project window, and a new **GameObject > Cinemachine** menu is available. You can also access this menu when you right click inside the Hierarchy view.

Create Main Scene

What Is Cinemachine?

Cinemachine is a suite of modules for operating the Unity camera.

Cinemachine solves the complex mathematics and logic of tracking targets, composing, blending, and cutting between shots.

It doesn't replace the Unity camera, but is used to control the Unity camera in an easily configurable way.

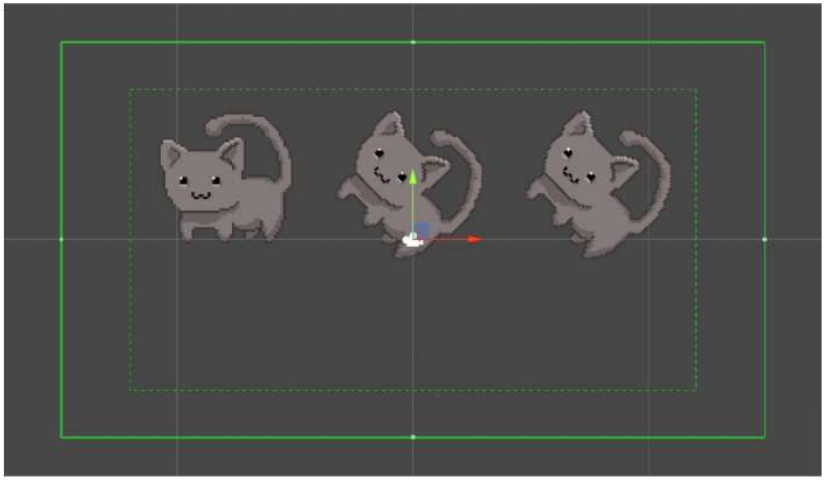
We'll use Cinemachine to get the camera to follow our player character, and to also create a 'target group' to move our camera to keep the player and the aim mouse position in view together (more later!)

Pixel Perfect

2D Pixel Perfect

The **2D Pixel Perfect** package contains the **Pixel Perfect Camera** component, which ensures your pixel art remains crisp and clear at different resolutions, and stable in motion.

It is a single component that makes all the calculations Unity needs to scale the viewport with resolution changes, so that you don't need to do it manually. You can use the component settings to adjust the definition of the rendered pixel art within the camera viewport, and you can use the **Run in Edit Mode** feature to preview any changes immediately in the Game view.



Attach the **Pixel Perfect Camera** component to the main Camera GameObject in the Scene, it is represented by two green bounding boxes centered on the **Camera** gizmo in the Scene view. The solid green bounding box shows the visible area in Game view, while the dotted bounding box shows the **Reference Resolution.**

The **Reference Resolution** is the original resolution your Assets are designed for, its effect on the component's functions is detailed further in the documentation.

Before using the component, first ensure your Sprites are prepared correctly for best results with the the following steps.

Preparing Your Sprites

1. After importing your textures into the project as Sprites, set all Sprites to the same **Pixels Per Unit** value

Create Main Scene

What Is Pixel Perfect?

It’s a camera component for a style of 2D graphics.

It helps your pixel art to remain crisp and clear at different resolutions, and stable in motion.

It does the calculations to scale the viewport with resolution changes, so that you don’t need to do it manually.

Create Main Scene

Implementing Pixel Perfect

The Unity Pixel Perfect component that's found in the package manager doesn't work with the Universal Render Pipeline.

There is another Pixel Perfect component supplied in the Universal Render Pipeline that should be used.

When using Pixel Perfect you also need to add the Cinemachine Pixel Perfect extension – both pixel perfect and Cinemachine modify the camera's orthographic size, and the extension allows both to operate with each other.

When using Pixel Perfect you don't want your sprite textures 'compressed' – so there are some sprite settings that you need to make when importing textures as sprites.

Orthographic Size

Create Main Scene

What Is Orthographic Size?

Orthographic camera mode ignores 'depth' (the z-axis position) when rendering things to the screen. With 2D we are controlling depth using sorting layers so we'll use the orthographic camera mode.

Orthographic size is basically how 'zoomed' out the camera is. A high orthographic size value means the camera is 'zoomed out' so things will appear smaller on the screen. A lower orthographic size value means the camera is 'zoomed in' so things will appear bigger on the screen.

So setting the orthographic size value in the camera is all about how big you want your sprites to appear, and Unity 'units' is a key value in this calculation. Unity units is a measure unity uses to determine screen size and how big sprites should be rendered.

When you import a texture as a sprite you set a value for 'Pixels Per Unity Unit' for the sprite. This is a key 'decision' to set the relative size of all your sprites in unity units.

You then set the orthographic size in the camera to control how many unity units the camera will fit in the screen. Orthographic size is how many unity units you can fit vertically in half the screen. So if you set an orthographic size of 5 that means your screen will be 10 unity units in vertical height. So if you set a sprite to be 16 pixels per unity unit, then you'll be able to fit in 10 sprites vertically.

Create Main Scene

Orthographic Size