HOW-TO GUIDE

Cinema-quality lighting in real-time

How to achieve a high-end look in your Unity animation project

Whether you’re an experienced CG lighting pro or a junior 3D artist, you can create rich, nuanced lighting for animation projects with Unity.

This ebook outlines some of the tools and features you’ll find useful as you get started.

Introduction

Real-time production is much more than a different method for rendering. It's actually a way to accelerate your entire animation pipeline.

Today, real-time production platforms built on game engines such as Unity have finally achieved the quality and maturity they need to serve as a production hub for cinematic-quality content creation.

But what makes the real-time rendering process different from offline rendering? How is it possible to go from hours of overnight compute time on a gigantic render farm to instant feedback and final 4K frames in less than a second?

Well, believe it or not, there's no magic involved. If you want polished, high-quality animation as fast as real-time, you can speed ahead with just a few new techniques.

When you use the Unity real-time production platform, the difference is in the many optimizations it makes behind the scenes, crunching every factor that goes into a perfect frame.

In this ebook, we'll discuss those factors, introduce some real-time concepts, and walk you through some common animation lighting configurations in Unity to help you reach that elusive cinematic quality. When you're finished, you'll come away with concrete advice and valuable lighting tips you can bring to your own projects – and take your first step into real-time with an animation demo project you can try for yourself.

Let's get started.

A shot from the animated short film, Sherman – our real-world example of cinematic lighting in Unity

Working differently

Certain tasks make real-time different from traditional CG in your lighting workflow, even before you place your first light. Here are the first four approaches that will help you push your project further when working in Unity.

1. Commit to a rendering pipeline

The first thing to know before you can think about lighting for your project: you need to select your render pipeline, and stick with it.

The pipeline you choose will perform a varying series of operations that may not be applicable if you switch to a different pipeline mid-project. You have to commit to a pipeline right from the beginning.

There are different reasons for choosing each of the available pipelines. But since the goal of this ebook is achieving the highest cinematic quality, the optimal pipeline to work in is Unity's High Definition Render Pipeline (HDRP). HDRP is a hugely customizable, C# scriptable rendering path for high-fidelity graphics used in AAA-quality games and cinematics on high-end hardware. It's also the only pipeline compatible with real-time ray tracing technologies. HDRP uses physically based lighting and materials as well as compute shader technology – so you'll need compatible console-grade GPU hardware to use this pipeline.

Getting HDRP set up in Unity is easily done with a wizard. Once you have it, you open up a slew of sophisticated lighting options. You can also learn more about HDRP by reading this blog post.

2. Organize with volumes

In HDRP, many rendering and post-processing properties are driven by a Volume framework (and stored in the volume's associated profiles). Your settings will be essentially organized in volume categories that you create and add graphics characteristics to, driving the parameters of the whole rendering as viewed from a given position in the world. You give each volume priority and weight values that determine how it's addressed in the rendering of the on-screen image, making volumes a huge help in refining the workflow to work on the scenes at different scales (globally, locally, or per shot).

What's more, volumes are perfect for experimentation, since you can duplicate a volume profile asset, play around with its configuration, and know you can always revert to the original volume without consequence.

Get to know volumes in Unity for maximum flexibility and the ability to fine-tune how your real-time scene appears and performs.

3. Embrace collaboration

One of the most important advantages of working in real-time with Unity is that multiple people who need to handle the same scene can work on it without blocking each others' progress, instead of waiting for the other's task to be completed.

You'll see this capability in action in the concept of multi-scene editing. With **multi-scene editing**, one artist can make a change – for instance, in response to a modification that someone else made in the same shot – without stepping on the other's work. Unity handles each artists' data through version control and merges in any changes behind the scenes.

Multi-scene editing also gives you the ability to store the base art direction lighting (such as global illumination) in one scene, along with a separate scene whose settings are dynamic and shot-specific so that artists can safely modify the shot.

4. Save time with Prefabs

If real-time is new to you, one of the greatest advantages of using Unity is that it gives you quick ways to get started – templates you can use so you're not forced to manually set up every detail in a scene, start all over again in the next scene, and then waste time tracking down elements when you need to change something small.

Using **Prefabs** is one of those shortcuts. These templates are great for grouping objects that need to be reused multiple times. A Prefab lets you modify the composition of your group, and all instances will follow automatically.

Unity Prefab instances also let you easily and selectively revert any derivation you made, or apply it back to the original Prefab and propagate changes instantaneously. This alone can save you hours on a shot.

Now let's get lighting in Unity with these familiar principles: global illumination, reflections, and direct lighting.

The big picture: Global illumination

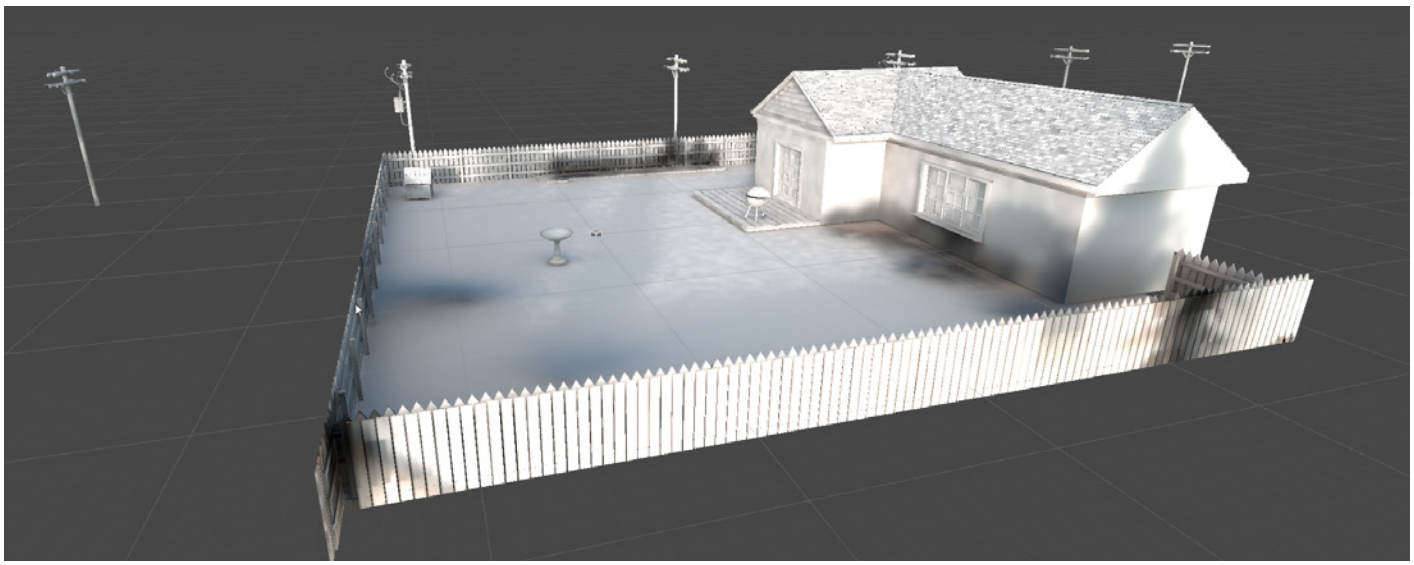
In an offline renderer, light scattering is normally achieved through expensive path tracing. In order to achieve similar results in real-time rendering, the engine instead emulates this effect by building an approximation, performing an initial pass of light bouncing around the static elements of the scene – global illumination.

Unity's baked **Global Illumination** system optimizes lighting by storing the calculations of indirect diffuse lighting into **light maps** for static objects and **light probes** for dynamic objects, then computing all the indirect specular lighting into **Reflection Probes**. This way, you can keep the light calculations simple at a lower resolution with the Baked Indirect lighting mode – which means the **baking** time is faster, and you maintain the flexibility of allowing small shot-by-shot corrections of directional orientation.

Lightmapping

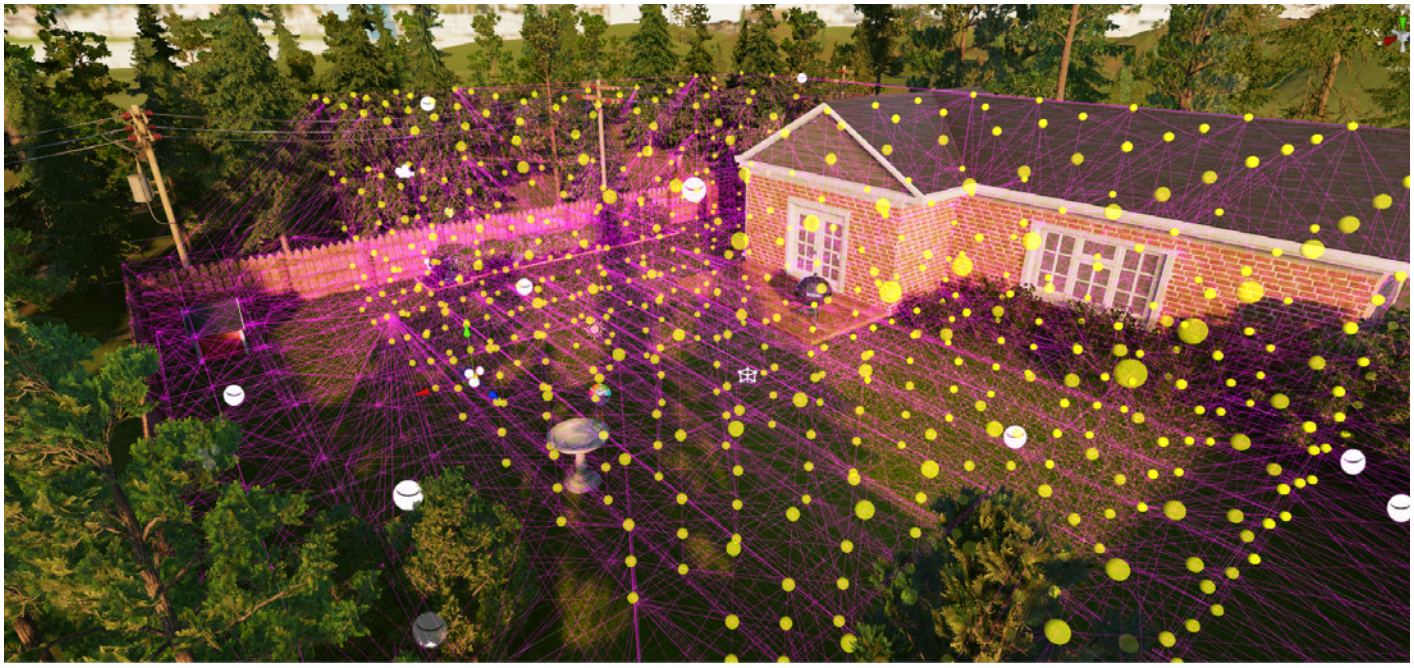
Lightmapping is a form of surface caching in which the illumination of surfaces in a virtual scene is pre-calculated and stored in texture maps for later use. Most traditional DCCs have lightmap-generating tools, so this concept should already be familiar.

In Unity, lightmapping is an integral part of the rendering engine, directly affecting the quality of lighting in any given scene – from the smoothness of shadows to blending between light and dark areas. With lightmapping, you'll configure how many lighting samples you need to retrieve lighting information, as well as the resolution of the map itself.



A view of our baked (indirect diffuse) lightmaps

Light probes



Light probe array with a one-meter spacing

After establishing the ambient lighting and lightmapping settings, you can use a light probe group or array to illuminate the remaining props and dynamic objects. The scene here uses an array of probes in the inner yard with a spacing of one meter between them, following the walls (and making sure no probes end inside an object).

You can experiment with using multiple groups of light probes to see how they affect performance. Larger objects, such as the lawn grass patches, benefit from a more refined probe lighting through the use of a *Light Probe Proxy Volume*. This way, an object can be lit by many localized samples of the probes, rather than by a single result.

Ambient occlusion

Screen Space Ambient Occlusion (SSAO)

You've explored how to generate great localized ambient light. The next concept, ambient occlusion, is used to darken creases, holes, intersections and surfaces that are close to each other. In the real world, such areas tend to block out or occlude ambient light, so they appear darker. You can author ambient occlusion maps for some objects, such as inside the doghouse or the raccoon's mouth, using your preferred DCC.

Next, to "glue" all of your elements together, you can use a post-processing effect called Screen Space Ambient Occlusion (SSAO). This can provide a superior result to setting Ambient Occlusion in your lightmaps, avoiding the over-occlusion that comes from having multiple overlapping objects with their own individual occlusion settings.

With Illumination mode selected and light probes added, it's time to move on to indirect specular lighting – your reflections.

Object-based Baked Ambient Occlusion

Reflections: High-frequency indirect lighting

Real-time reflections

The most processor-intensive attribute to work with is real-time reflections. These can quickly deteriorate your frame rate, making it difficult to work in the Editor, so plan for them carefully.

Indirect Specular Lighting (reflections)

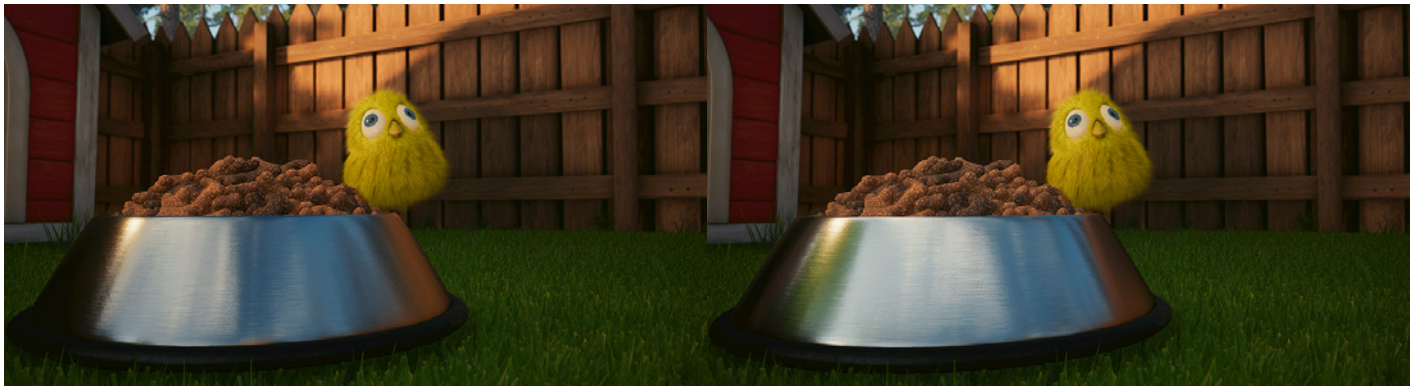
Ambient and baked reflections combined

Planar reflections

Example of a planar reflection effect

Planar reflections are mirrors. These are perfectly suited for flat surfaces, but they can also be used in more complex situations like the one created here to help connect the raccoon and the inflated hose. The more complex the reflection, the more expensive it is – not only in terms of memory, but also increased rendering time, because rendering will be performed an additional time at the resolution of the planar reflection.

Real-time reflection probes



A small real-time spherical probe engulfs the bowl to showcase the inflating hose that's about to rupture.

Real-time reflection probes are another class of element that can be strategically placed to show important details in the reflections and occlusions of light sources. Reflection probes behave like cameras – they capture a view of their surroundings and store it so materials with reflective properties can display it accurately and dynamically when the viewing angle changes. Capturing multiple objects' reflections in probes can improve performance, compared to each object calculating its own reflection texture.

Static (baked) reflections

Baked reflection probes need to fully cover this set.

A good rule of thumb is to place your **capture point** at the average height at which your camera work is captured. If you look closely, you will notice that some reflection probes are very specific to shadowed areas. Using more localized reflection probes will ensure greater precision in the indirect reflections. In these cases, they will provide more accurate sky occlusion in the less open areas.

Real-time direct lighting

Direct lighting casts shadows across the scene.

Now it's time to place your real-time direct lighting. In the HDRP, lights are physically accurate. The larger categories of direct lights – Directional and Punctual – have two properties that are handled in specific ways in real-time: Speculars and Shadows. The following are some of the controls and settings you'll find in Unity to configure them.

Controlling direct specular lighting

Specular lighting emulates the reflection of your light source from any object it appears on, and one important setting has a much greater impact than others: Radius.

The Radius has an impact on the size of specular highlights, diffuse lighting falloff, and the softness of baked, ray traced, and PCSS shadows. Radius affects so many other characteristics (and their performance) that it's important to keep an eye on its setting for every light, especially in very large bounce lights.

Shadows in moderation

Shadows are far from free, and they can be controlled for each light. Unity HDRP's Shadow Atlas system enables you to have a large number of shadow-casting lights at the same time. For the purposes of cinematic lighting in the sample scene, the largest atlas possible is used, but keep in mind that while a large atlas substantially improves quality, it's also more expensive in terms of computing time. The tradeoff is that it can eventually slow down the Editor. And while reducing the atlas size improves performance, it reduces the amount of shadow maps you can put in.

In short, managing shadow quality is a complex process that can be refined to attain balance between performance and quality. There are many parameters you can tweak for real-time, and there's also a very good debug mode in Unity to help you narrow down to the best choice for your scene.

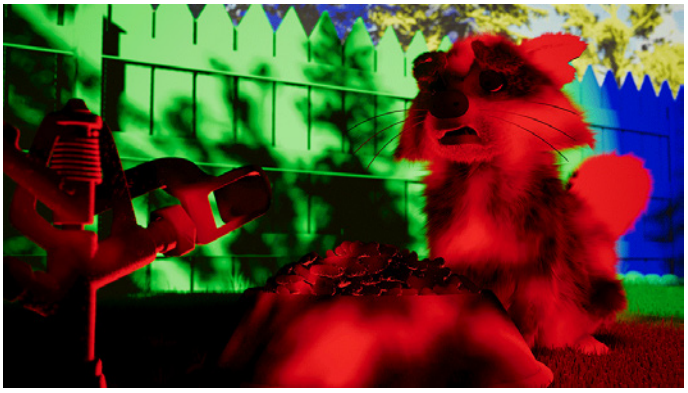
For example, to get the specific look in the sample project, every light source used casts a shadow. This is why area lights were not recommended here. But that's not necessarily the case with other projects. For instance, you can use **fill lights** for lighting adjustments that do not cast shadows. It's all about experimenting with individual settings.

Directional light

HDRP Directional Lights make use of Cascaded Shadow Maps (CSM).

Shadow Cascades help solve a problem called perspective aliasing, where real-time shadows from directional lights appear pixelated when they are near the camera. The more cascades you use, the less your shadows are affected by perspective aliasing.

This naturally increases the rendering overhead – but not as much as if you were to use a high-resolution map across the whole shadow.



Adjusting Percentage Closer Soft Shadow (PCSS) changes the lights' penumbra, or outer shaded region.

Contact Shadows

Contact Shadows are shadows that are **ray marched** in screen space inside the depth buffer. The goal of Contact Shadows is to capture small details that regular shadow maps fail to capture.

They are not filtered, which means they can appear very harsh over blurry shadows and leave dotted artifacts. But in Unity, you can use the Accumulation Motion blur to filter them by using very low/grainy settings and let this noise accumulate.

Mastering volumetric lighting

Atmospheric effects are an important tool in your real-time arsenal when it's time to create depth in your scenes. HDRP high-quality **volumetrics** provide you with a robust solution.

Volumetrics can be enabled in every light's inspector, and Volumetric Fog is used in the volume visual environment's Fog Component. This effect realistically simulates the subtle glows and beams of light that stream and scatter through elements in the scene. You can maximize the effect's quality through the advanced options of the Fog Volume Override.

For the best localized effect, you can use Density Volumes, placed shot by shot to maintain full control over their appearance as in the sample project; alternatively, they can be placed once for all shots at strategic places in your scene. These volumes can also enable variations of local density using 3D textures.

Creating depth using volumetric lighting effects

Light layers

Adding a dedicated catchlight to the eyes

In HDRP, you can assign an entity to specific **light layers**. This technique can be used, for instance, to manually add a **catchlight** to make the raccoon's eyes look more alive and interesting. Light layers let you manage any undesired specular highlights, for instance those coming from multiple **punctual lights** you use for fill and **rim lighting**. You can also use them to prevent punctual lights from interfering with your water effects.

With the scene set up, lights placed, and objects treated with the lighting effects they need, you're ready to tie the entire sequence together, shot by shot.

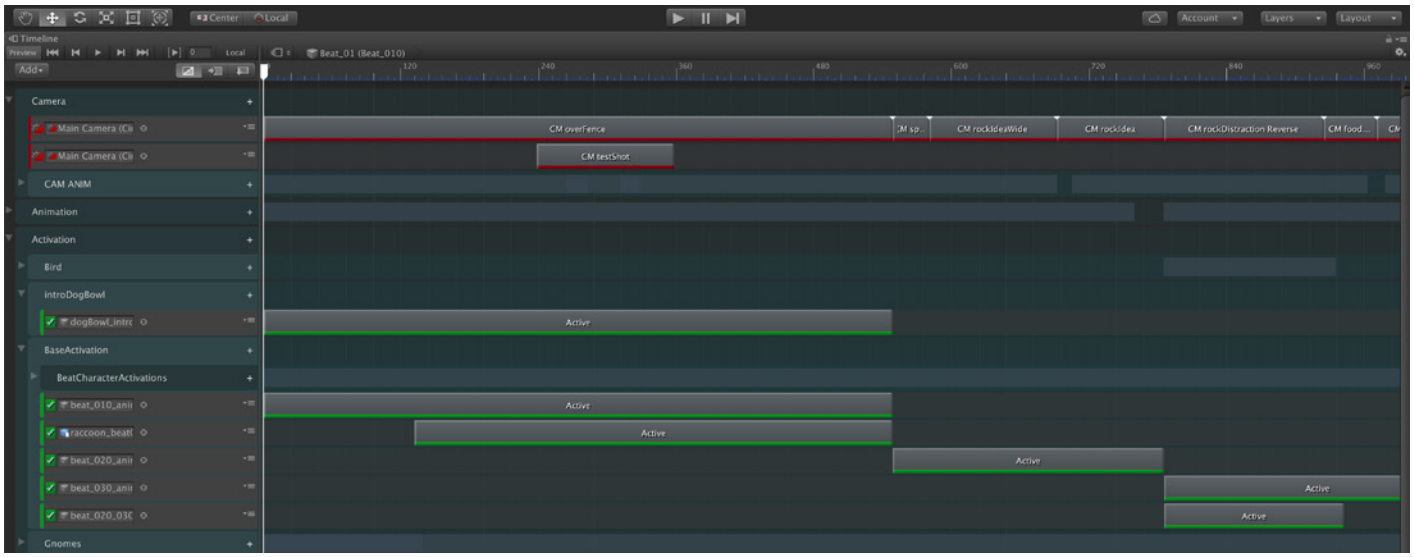
Adding animation

The Unity Timeline

Think of the Timeline as the central hub for coordinating and organizing each shot: from the cameras, to the audio, to light movements, and animations. This is your sequencer, where all the work you've done on the scene you've just lit and all the assets and lights you've placed in it, will come to life.

If you've ever cut two video or audio clips together in a media editor, then the Unity Timeline will look familiar to you. It's where you can orchestrate, rearrange, and determine the timing of how each component in a scene animates in each shot.

To optimize scene animation for real-time, you can use **activation tracks** to determine whether an object is active or inactive during a scene. This can make it easier to manage groups of assets dedicated to a shot, rather than animating every component of a lighting rig shot by shot.



The Unity Timeline

Prefabs

Once you're in the Timeline working on animating your scene, you'll start to work with Prefabs, Unity's templates for grouping objects that need to be reused multiple times.

Unity also supports Nested Prefabs. This setup is perfect for quickly iterating on a lighting rig. As an example, here's the Lighting Prefab structure for a sample animation beat. You can see the shot-by-shot structure, driven by activation tracks. Since only the lighting objects that exist during the shot are displayed, all others are hidden.

A big benefit of working with Prefabs is that once an object has been instantiated in your scene, you don't need to keep the scene checked out in source control to work on your lighting. This means that other members of your team can work on the scene at the same time.

Post-processing

Post-processing in Unity is the set of rendering effects that you apply, based on your existing rendered scene, before generating the final render. That means you get instant visual feedback for the effects you choose, instead of having to add finishing steps afterward, dramatically improving the scene without altering your existing content.

Sherman before and after Color Grading

Color grading is an essential part of any production process, especially in cinematic projects. Here are some of the adjustments that can achieve impressive results directly in Unity with just a few settings.

Tonemapping

Since the entire render happens in HDR linear space, it's important to remap the final image values in a range suitable to display on screen. This project is set to use the Academy Color Encoding System (ACES) tonemapper and color space for grading. It gives you highly contrasted images with very strong, dark tones. To achieve the uplifting color palette in the sample project, gamma was an important parameter to experiment with – as it will be with most cinematics that adopt this style of tonemapping.

Shadows, Midtones, Highlights

The Unity platform incorporates highly controllable Trackballs into the UI. If you've worked in lighting before, trackball controls are a familiar tool – they're easy to manipulate, and lighting pros really love them.

Color Curves

Though only one can be displayed at once, many grading curves are available. This project uses Hue vs Sat, Sat vs Sat, and Lum vs Sat.

Depth of Field

Depth of Field (DoF) is not just the byproduct of a fast lens; it's also an essential tool to drive the focus in your story. Images in an animation scene move quickly, so people's attention has to be carefully guided. The Cinemachine virtual camera tool in Unity also has an extremely useful option for tracking focus in your scene.

Vignette

Vignette is a powerful effect, but one that can easily be overused. It mimics an artifact of filming through a lens, visually representing the loss of light in the corner of the frame. For realism, this effect should always be rounded.

Bloom

Bloom is a beautiful artifact created by strong light bleeding on your camera sensor and dirty lenses. You can simulate this effect in Unity, but, as with many effects, a subtle touch is crucial, and this technique should be used in moderation.

Dithering

This is a camera effect designed to remove banding issues on a subtle gradient when outputting to an 8-bit display.

Conclusion

Lighting for animation is complex and takes time to master. But it only requires a few shifts in thinking and parallel behaviors to apply it to a real-time pipeline. In this ebook, we covered almost everything a traditional lighting artist would need to know to light a real-time rendering production.

Many Unity users say they can't see themselves ever going back to "the good old days" of traditional lighting. Real-time animation is taking the craft to the next level and creating new worlds of opportunity. With more time and room for taking risks, experimenting, and iterating without negative consequences, the results can only be better.

Choose your path

Ready to learn how real-time creation can accelerate your pipeline without disrupting your artists? Unity is accessible to more creators, artists, and TDs than ever. Take the Unity Learn free, one-hour tutorial *Introduction to Lighting and Rendering* and get completely comfortable with how lighting works in Unity.

If you're brand new to the art of lighting for real-time rendering, refer to the glossary on the following pages.

Real-time lighting glossary

Phrase	What it is	When you'll use it	Why it matters
activation track	The track in Unity's Timeline Editor that contains animation within a clip.	To add individual animations to your Timeline that play, or activate, at a specified position within a sequence.	It's easier to manage and activate groups of assets in a shot together, rather than animating every component of a lighting rig shot by shot.
ambient occlusion map	Grayscale image that, when applied to a surface, indicates areas that should occlude (block out) ambient light so they appear darker.	To simulate the soft shadows that occur in creases, holes, and surfaces that are close to one another.	It adds realism to your lighting by approximating the amount of ambient light that should hit the surface of these darker details.
Baked Global Illumination (baked GI)	Lighting baked into textures called lightmaps and into Light Probes.	When you've generated traditional lightmap textures offline during the precompute process, rather than inside Unity.	Better performance is realized even with extremely high-fidelity textures because these assets are not being computed at rendering time.
catchlight	The bright highlights found in a subject's eyes when a light source is reflected in them.	Anytime a character is lit with available light including the sky, a window, or other available light source.	It can add realism, make a character appear alive, or create a dramatic effect (such as revealing a character hidden in gloom or shadows).
Cinemachine	Unity's suite of smart, codeless cameras that allow artists to tune, iterate, experiment, and create camera behaviors in real-time.	During animation setup, when determining where to aim and focus in a scene, or whether to use dollies and tracking like a physical camera.	Cinemachine's virtual cameras allow you to iterate new ideas on the fly, without expensive camera logic development (or rendering time).
global illumination (GI)	A group of techniques that model both direct and indirect lighting to provide realistic lighting results.	To generate the indirect lighting in the scene, primarily as a function of the direct lighting.	GI brings a more realistic, balanced, cohesive look to a scene that's less labor-intensive and more true-to-life than the effect created by local lighting on individual objects.
High Definition Render Pipeline (HDRP)	HDRP is Unity's pre-built, scriptable pipeline for authoring cinematic-quality projects using physically based lighting and advanced visual effects.	When you need advanced rendering and shading features for projects that require a high degree of visual fidelity using high-end hardware.	HDRP in Unity gives animation studios the ability to achieve graphical fidelity and visual realism in their productions.
light layers	The ability to assign lights into different layers, which are then independently configured.	To make objects in the same scene react only to certain distinct lights as needed.	Fine-tuning of lights and how different objects respond to them provides opportunities for higher-end details and effects.
light probes	Light probes are positions in the scene where the light is measured (probed) to provide indirect lighting values for dynamic scene objects during rendering.	In areas between objects, where light is passing through empty space in your scene.	Light probes are important tools in providing high-quality lighting (including indirect bounced light) on moving objects in your scene.

Phrase	What it is	When you'll use it	Why it matters
lightmaps	A pre-rendered texture (overlaid on top of scene geometry) that contains the effects of light sources on static objects in the scene.	When using Unity's Baked Global Illumination system to provide realistic lighting results.	Lightmaps make it possible for much of the lighting to be entirely precomputed, so real-time performance is not compromised.
lightmapper	Unity's underlying system to generate data for lightmaps and light probes by shooting light rays, calculating the light bounces, and applying the resulting lighting into textures.	Only when the surface is made up of non-overlapping UVs with small area and angle errors and there is sufficient padding between the charts.	Various lightmappers will often produce different lighting looks, as they might rely on distinctive techniques to produce the lighting data.
multi-scene editing	The ability in Unity's Editor to have multiple scenes open in the Editor simultaneously to enable multiple people to collaborate simultaneously in a scene within a seamless workflow.	When configuring a setting that affects multiple scenes or when more than one person needs to modify the scene, e.g., both the lighting artist and the environment artist.	An important time-saver when multiple artists need to access the same scene simultaneously, or when certain scene files (for example base lighting like GI) need to be reused across different compositions.
physically accurate	Physical Light Units (PLU) such as candela, lumen, lux, and EV, which are controlled in Unity in the same way as real photographic lighting.	When you want to simulate real-life light measurements, like those you see on light bulb packaging or a photographic light meter.	The ability to achieve highly realistic results is a major principle of HDRP, which includes physically accurate lighting, material and camera interactions.
Prefabs	A Unity workflow system that allows you to create, configure, and store an object's properties as a reusable asset.	Whenever you want to reuse an object's configuration in multiple places in a scene, or across multiple scenes in a project instead of recreating a new one every time.	Save many hours on any shot by duplicating, modifying, experimenting with, and instantly propagating changes to a single instance of an asset's settings.
punctual lights	A light is considered to be punctual if it emits light from a single point.	Any time you place a spotlight or a point light, you're using punctual lights.	It's possible to create tightly targeted, very unique illumination effects that don't require strict physical accuracy in your scene.
ray marching	An advanced ray tracing algorithm in which rays are not merely computed where they intersect with solid surfaces, but also sampled and modified as they pass through space.	When your scene includes contact shadows to capture small details that regular shadow-mapping algorithms fail to achieve.	Useful for rendering where a solid surface is difficult for a ray to find – hence its ability to reveal smaller details in shadow.
reflection probes	A reflection probe captures a view of its surroundings and stores the results, which are then used on objects to produce accurate reflections of their surroundings.	To help you create accurate reflective materials and realistically reactive visuals for objects in your scenes.	Subtle and accurate reflections, balanced with controlling the area and detail covered, maximize a scene's believability without creating a drag on performance.

Phrase	What it is	When you'll use it	Why it matters
reflection probe capture point	The position of a reflection probe relative to the scene it's mirroring.	When placing reflection probes in a layout and setting local and world properties.	To ensure reflection angles and positions portray a precise and accurate view.
shadow cascades	Splitting shadow maps into zones by proximity to the camera to reduce pixelation in the closer shadows.	When you're using a directional light that covers a large portion of the scene, leading to visible aliasing on closer shadow pixels.	Managing shadows with cascades is less of a drag on performance than using a high-resolution map across the whole shadow.
Scriptable Render Pipeline (SRP)	A Scriptable Render Pipeline allows you to control rendering via C# scripts, giving you a high degree of customization. There are two pre-built systems in Unity – High Definition Render Pipeline (HDRP) and the Universal Render Pipeline (URP) – that tailor the rendering process to your target platform so the project is optimized for specific hardware.	Choose the HDRP to deliver cinematic graphics for projects specifically targeted to high-end hardware with high-fidelity visuals. Or, choose the URP when you don't need cinematic fidelity but want greater flexibility and extensibility than the built-in render pipeline and high-quality graphics across broad use cases.	Committing to the render pipeline at the start gives you access to a visual way of creating and editing sophisticated pipeline tools without having to write code.
Volume framework	The Volume framework is how you partition your scene into global or local areas so that you can control lighting and effects at a finer level, rather than tuning an entire scene.	You can add as many volumes to your scene as you want to create different spaces, then light each one individually for realistic effect.	Gives you fine scene-by-scene control over environment settings, such as fog color and density, that alter the mood of your scene in calculating the final render.
volumetrics	A lighting effect that allows the viewer to see rays of light that appear to have body or depth, beaming through the environment.	When you want to add an atmospheric fog look to your scene.	Provides realistic simulation of the interaction of lights with fog, allowing for physically plausible rendering of glow and lightrays – but can be costly to performance.


unity