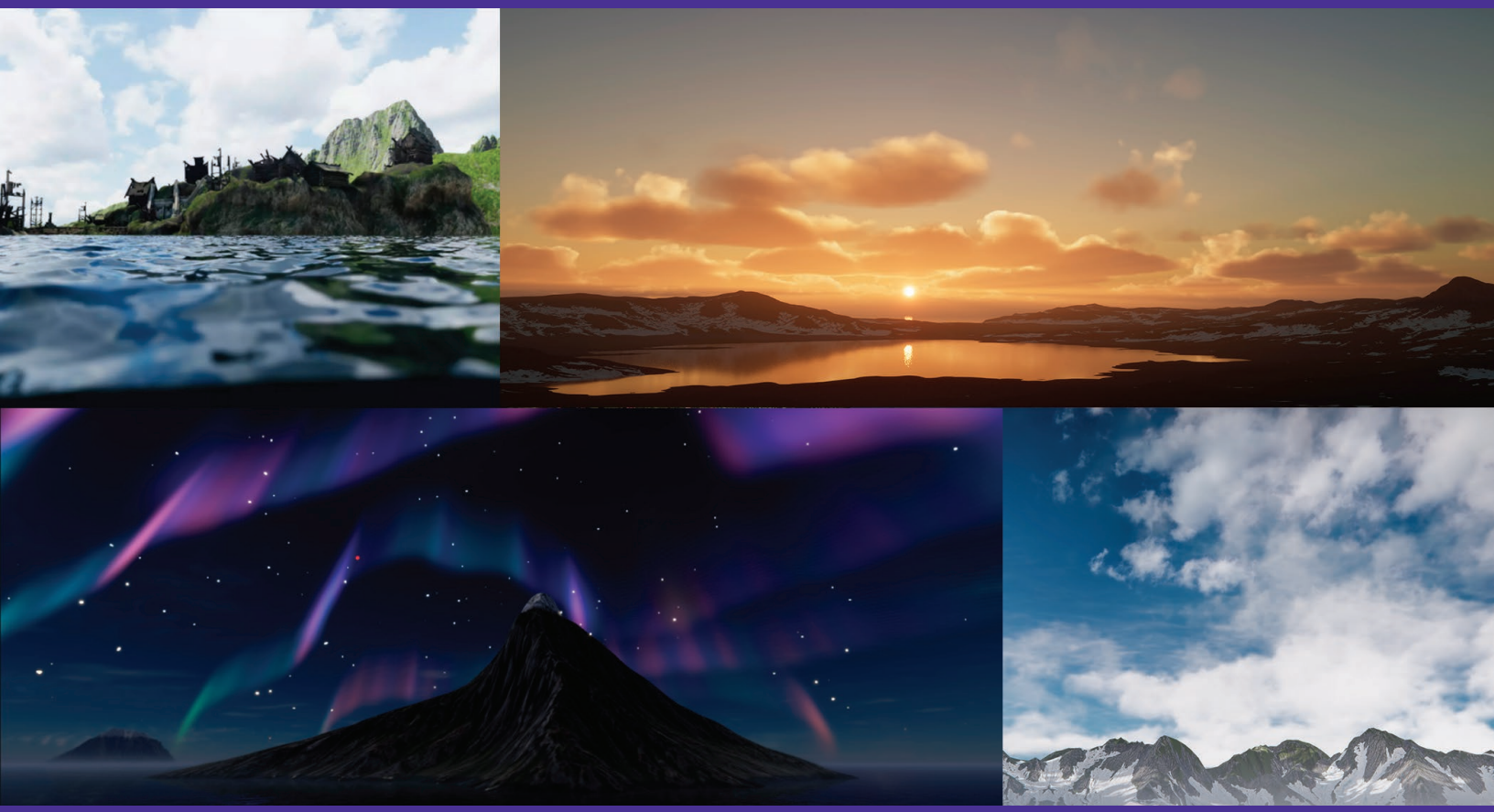
→ E-BOOK

HD レンダーパイプラインにおけるライティングと環境

(Unity 6 版)



Unity®

Contents

はじめに

ゲームの世界は単なる背景ではありません。ゲームのあらゆる瞬間が繰り広げられ、相互に作用する舞台です。

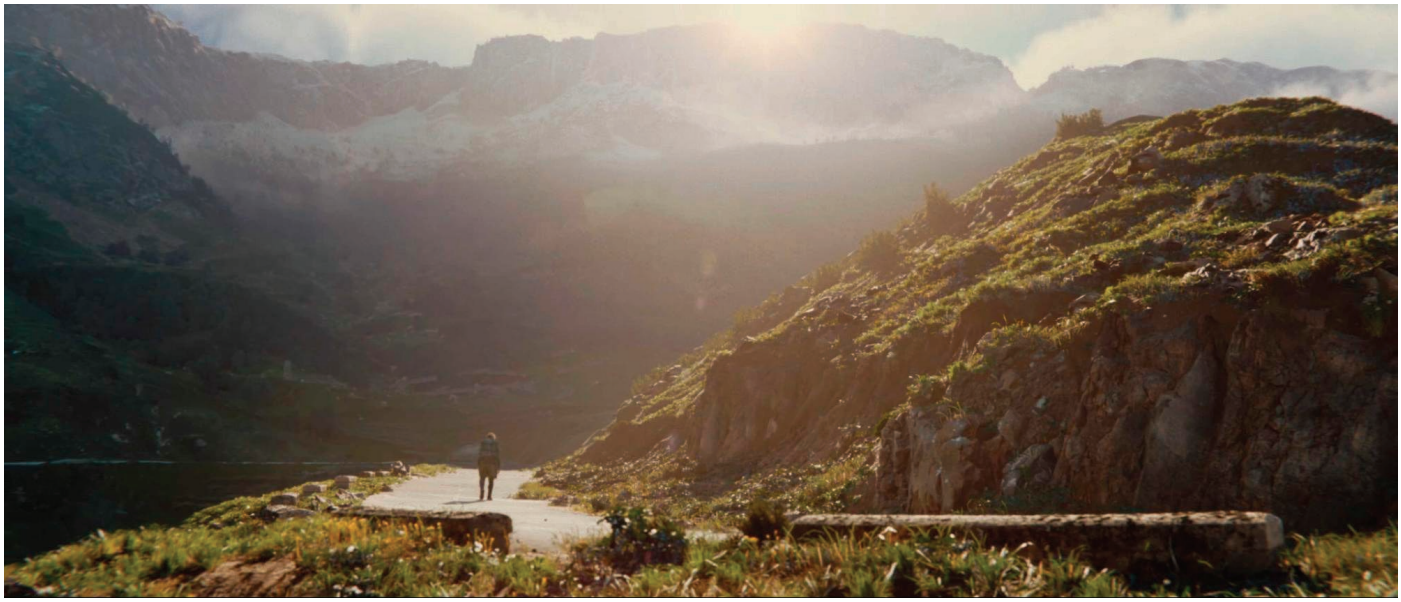
Unity 6 の HD レンダーパイプライン (HDRP) でゲーム世界の構築に取り組めば、リアルタイムグラフィックスの枠を押し広げる、息をのむような環境を作成できます。HDRP によって、物理的に正確な光線、忠実度の高い材質、映画のような効果を実現でき、シーンの没入感が高まります。

ボリュメトリックフォグとシャドウでシーンに深みを加えましょう。SpeedTree の動的な植生で風景を生き生きとさせ、Decal Projector で汚れやひび割れ、摩耗を施して古びた表面にしましょう。つや消し金属の柔らかい光沢から、ガラスの層を通る光の複雑な動きまで、レイトレーシングで新次元の臨場感を表現しましょう。

今回、この HDRP ガイドを更新し、最新のワールド作成ツール一式を含めました。Terrain Tools で、文字通り山を動かしたり、渓谷を削ったりしましょう。Cloud Layers で空を彩り、Lighting Scenario で昼から夜へとリアルタイムに遷移させましょう。HDRP の高度な水システムは、リアルな波、コースティクス、泡で海や川を覆うことができます。

皆さんのゲーム世界ではどのようなストーリーが始まりますか?HDRP を使って、ビジョンを具現化しましょう。

HDRP を使用してリアルなゲーム世界を構築できます。上の 2 つの画像は Unity のハイエンドデモ『*Time Ghost*』のものです。



このガイドの利用方法

本書は、HDRP の豊富なツールセットを活用できるように用意されたモジュール式のリファレンスガイドです。ライティング、環境効果、レンダリング手法、最適化戦略など、関心のあるトピックに直接飛んで構いません。

各章は独立していますが、多くのトピックが相互に関連していることに注意してください。例えば、HDRP のライティングシステムは、物理ライトユニット、Volume、ポストプロセス技術など、複数のセクションが関係します。

同様に、地形、水、空、フォグのシステムなど環境に関連する機能も、互いに連動して没入感のあるゲーム世界を作り出します。このガイドブックを最大限に活用するには、関連する章も合わせて調べ、HDRP の機能をより詳しく理解するようにしてみてください。

HDRP のライティングと環境

Unity の元々のライティングシステムが HDRP の多様な機能で拡張され、現実世界の光に似せてシーンがレンダリングされます。

- **物理ベースのライト単位と高度なライティング:**HDRP では、現実世界の光の強度と単位を使用します。既知の光源の基準に合わせて、物理カメラを使用して露出を設定します。スポットライトやエリアライトの新しい形状オプションを使用して、ライトの配置を制御できます。スクリーンスペースグローバルイルミネーションや Screen Space Refraction などのリアルタイムの効果を適用できます。
- **空の景色:**様々なテクニックで自然な空を創造することができます。物理ベースの Sky ボリュームオーバーライドで大気のシミュレーションをプロシージャルに行ったり、ボリュメトリッククラウドやクラウドレイヤーを追加したり、HDRI を適用して静的な空のシミュレーションを行ったりできます。
- **Terrain (地形):**HDRP で強化されたスカルプティングツール、テクスチャレイヤリング、カスタムブラシで、現実的な風景を作り出しましょう。ハイトマップで地形を作り、風に揺れる植生を追加して、リッチかつ動的なシーンを生み出せます。
- **水システム:**相互作用する水面をシミュレートします。物理的に正確な波のシミュレーション、水流、そして泡を追加し、シーンに命を吹き込みましょう。
- **フォグ:**フォグを使用して、シーンに奥行きや立体感を加えましょう。ボリュメトリックを有効にして、フォグエフェクトを前面のオブジェクトに組み込むことで、光のシネマティックな軌跡をレンダリングできます。ボリュメトリックライトやシャドウをライトごとに管理して、ローカルボリュメトリックフォグコンポーネントを使用し、3D マスクテクスチャーでフォグの密度を微細に調整することが可能です。
- **ボリュームシステム:** HDRP には直感的なシステムが採用されており、カメラの位置や優先度に応じて、様々なライティングの効果や設定の適用を調整できます。ボリュームを重ねたりブレンドしたりすることで、シーン内のあらゆる領域を非常に細かく制御できます。

- **ポストプロセス:**HDRP のポストプロセスは、既存のボリュームシステムからうまく処理を行い、Volume オーバーライドにてまとめて管理されています。アンチエイリアシング、トーンマッピング、カラーグレーディング、被写界深度、その他のさまざまな効果を追加できます。
- **高度なシャドウ:**HDRP は、シャドウに対するビジュアル面とパフォーマンス面での高度な制御を提供しています。色合いやフィルタリング、解像度、メモリバジェット、更新モードを調整できます。コンタクトシャドウやマイクロシャドウを利用して、細かいディテールやさらなる深みを際立たせましょう。
- **高度なリフレクション:**さまざまな手法を使用して反射面をレンダリングできます。リフレクションプローブでは、従来のリフレクションマッピング手法に加え、フラットな表面向けのより高度なオプションを持つ平面リフレクションプローブを使用できます。さらにスクリーンスペースリフレクション (SSR) によって、深度バッファを使用したリアルタイム手法を利用できます。
- **拡張性:**HDRP は Unity の Scriptable Render Pipeline (スクリプタブルレンダーパイプライン) を基盤にしています。経験豊富なテクニカルアーティストやグラフィックスプログラマーであれば、パイプラインを標準機能以外にも拡張できます。

HDRP に触れるのがまったく初めてという方は、Up and Running with HDRP チュートリアルを確認してください。

『Time Ghost』

『*Time Ghost*』は、『*The Blacksmith*』、『*Adam*』、『*Book of the Dead*』、『*The Heretic*』、『*Enemies*』などのプロジェクトを手掛けたチームが 2024 年に開発したリアルタイムの映画的なデモです。

このプロジェクトでは、Adaptive Probe Volume (APV) や Scenario Blending によるライティングの改善や GPU Resident Drawer を使用したパフォーマンスの強化など、HDRP の最新手法を紹介しています。『Time Ghost』では、Entity Component System (ECS) アーキテクチャ上に構築された Unity Entities パッケージを含む Unity の Data-Oriented Technology Stack (DOTS) を活用して、最大 1,200 万におよぶ植生インスタンスをレンダリングしています。

『*Time Ghost*』は HDRP を使用してリアルなライティングを作り出しています。

Unity 6 の新しい **GPU Resident Drawer** によってバッチ処理とインスタンス化の効率が向上し、デモの広大な環境を最小限の CPU オーバーヘッドでレンダリングできます。

『*Time Ghost*』では、**VFX Graph** と **Shader Graph** も幅広く活用されています。これらのツールは、リアルな爆発、動的なフォグ、複雑な環境効果の作成に役立っています。

VFX Graph が『Time Ghost』の視覚効果 (VFX) を支えました。

『*Time Ghost*』は こちら で視聴できます。Unite Barcelona 2024 の 舞台裏セッション では、ライティング、環境、キャラクター開発の取り組み方について、Unity Originals チームが詳しく説明しました。

Asset Store にある 2 つのサンプルパッケージが『Time Ghost』の背後にある芸術と技術の探求に役立ちます。

— **Time Ghost:Environment**:このパッケージは、**DOTS ECS** とプロシージャルツールを活用して、多数のシーンエンティティを管理し、リソースを割り当てます。

— **Time Ghost:Character**:このデモの主人公は、**Sentis AI** によるリアルタイムのキャラクターアニメーションと物理演算に基づくインタラクションを示しています。

これらのアセットをダウンロードして Unity 6 プロジェクトにインポートし、『*Time Ghost*』で使用されている高度な手法を学ぶことができます。

『Time Ghost』のメインキャラクターがダウンロード可能になりました。

Unity 6.1 における HDRP のアップデート

Unity 6.1 の HDRP に盛り込まれた新機能と改善点を、2022 LTS (この e-book の旧版の対象) の HDRP から移行する方向けに紹介します。

Lighting

LightBaker:HDRP の新しいバックエンド LightBaker v1.0 では、シーンの変更を毎フレーム継続的に確認するのではなく、ベイクの開始時にシーンの状態を "スナップショット" として取得します。手動でキャンセルしない限りベイクは中断されることなく実行され、不要な再計算が減り、エディターのパフォーマンスが向上します。

Baking Profile: Lighting ウィンドウでは、GPU ライトマッパー用の GPU Baking Profile が利用可能になりました。この機能は、ライトマップをタイルに分割し、ベイク高速化かメモリ使用量軽減のどちらかに最適化するプリセットを選択できます。

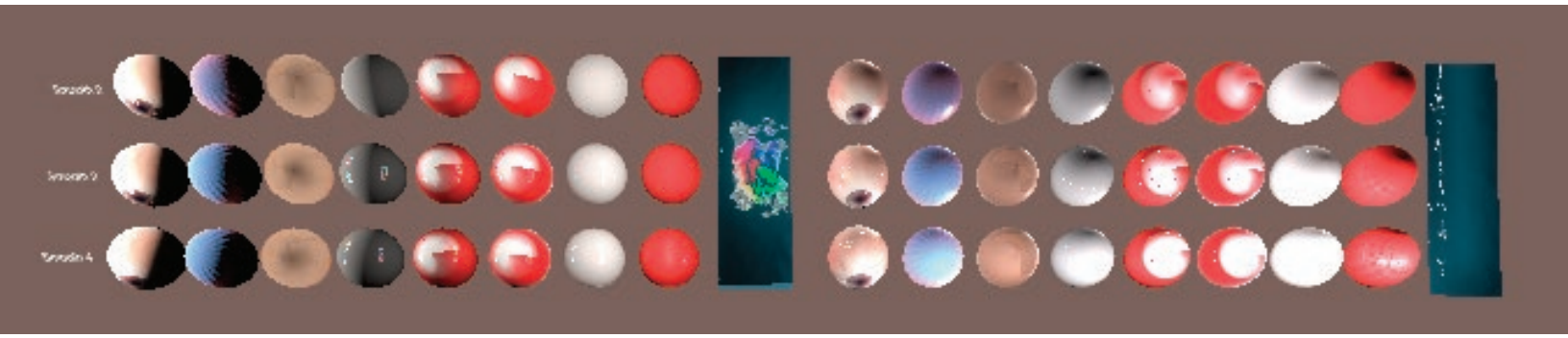
Adaptive Probe Volume (APV) のアップデート: LightBaker では、ライトマップとは別に Light Probe (ライトプローブ) と APV をベイクできるようになりました。APV は スカイオクルージョン のベイクにも対応しており、ゲームオブジェクトは アンビエントプローブ から取得した空の色を使用してライティングを動的に調整できるようになりました。これにより、APV を使用して昼夜のサイクルをシミュレートできます。

レイトレーシングの改善: レイトレーシング の加速構造をより細かく制御するための新しい API がいくつか用意されています。RayTracingAccelerationStructure には、いくつかの新しいメソッドが含まれています。

- UpdateInstanceGeometry では、新しい RayTracingMode.DynamicGeometryManualUpdate モードを使用して **下位加速構造 (BLAS)** を手動で更新できます。このモードでは、フレームごとにジオメトリを自動的に更新するのではなく、特定のインスタンスを "ダーティ" としてマークし、加速構造の再構築を手動でトリガーできます。これにより、レイトレーシングされたシーンでの動的オブジェクトのパフォーマンスを改善できます。
- AddInstancesIndirect は、変換行列用の GraphicsBuffer を使用して、複数のレイトレーシングインスタンスを **レイトレーシング加速構造 (RTAS)** に登録します。GPU がこれらのインスタンスを管理できるようになり、CPU オーバーヘッドが軽減され、パフォーマンスが向上します。インスタンスは、メッシュベースまたはプロシージャルに設定できます。
- RemoveInstances は、LayerMask またはレイトレーシングモードに基づいてレイトレーシングインスタンスを削除できるようにします。これにより、ランタイムにシーンの変化に応じて加速構造を動的に更新し、パフォーマンスを最適化できます。
- CullInstances は、フィルタリングとカリングのパラメーターを設定し、特定の基準に基づいて加速構造を簡単に更新できるようにします。

ライトマップのバイキュービックサンプリング:この機能は、ライトマップをぼかして滑らかにし、低解像度でも高品質に見えるようにできます。これは、ベイクしたライティングを使用する大規模な環境において、メモリやパフォーマンスの制約から低解像度のライトマップを必要とする場合に便利です。**Project Settings (プロジェクト設定) > Graphics (グラフィックス) > Use Bicubic Lightmap Sampling** で有効にします。

エリアライトの改善:エリアライトは、距離による減衰に "ピロー" ウィンドウ関数を使用するようになりました。これにより、光のフェードアウトがより滑らかになりました。さまざまなマテリアルタイプとの相性もよく、CPU のメモリ使用量も抑えられます。



エリアライトは、距離による減衰に "ピロー" ウィンドウ関数を使用します。

パストレーシングの改善:円盤状およびチューブ状のエリアライトが パストレーシング でサポートされるようになりました。**ボックスライト** も修正され、適切なイルミネーション動作が得られるようになりました。さらに、**ボリュメトリックフォグコントロール** がエリアライトに加わり、その影響を微調整して、よりリアルな大気の効果を表現できるようになりました。

パストレーシングは、円盤状およびチューブ状のエリアライトをサポートしています。

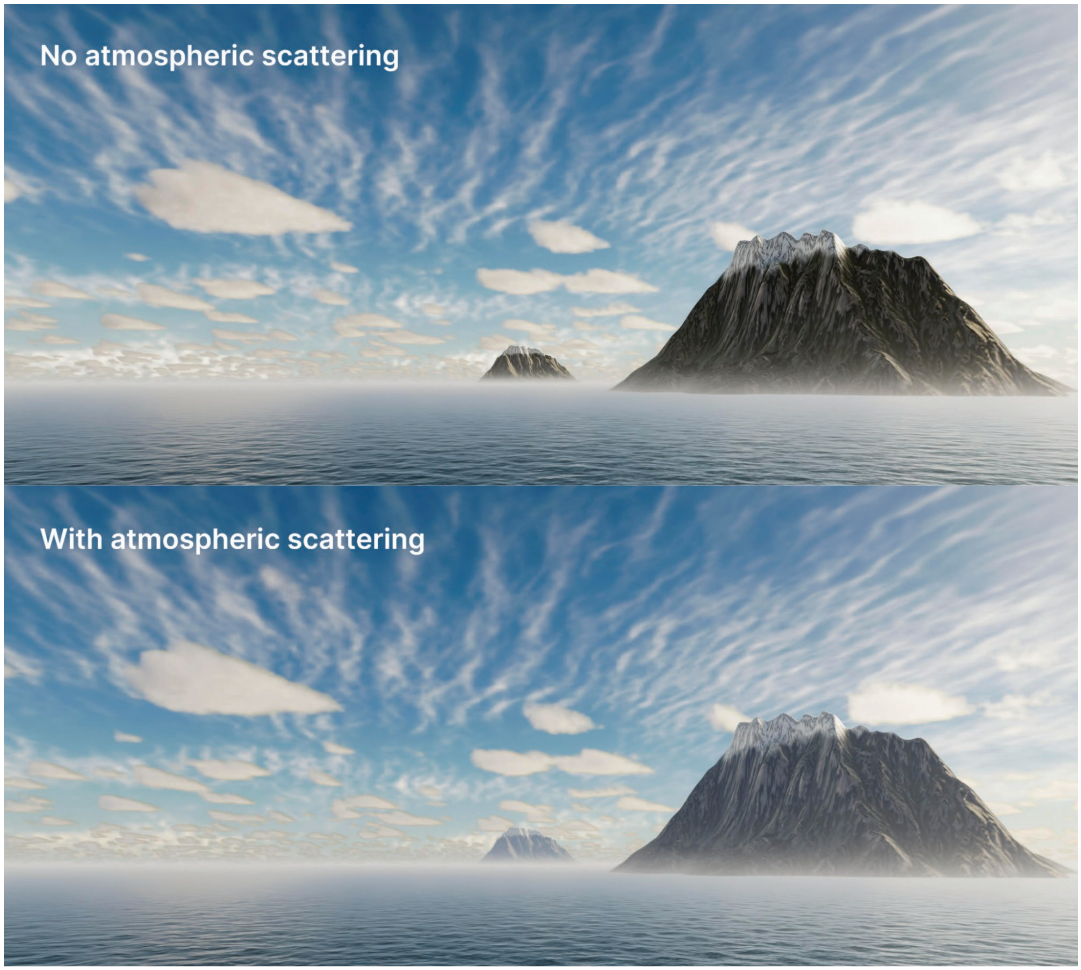
環境効果

Planet パラメーター: Visual Environment オーバーライド の共有パラメーター群が、ボリュメトリッククラウド、フォグ、物理ベースの空など、さまざまな環境効果に影響を与えるようになりました。

Physically Based Sky Volume:PBR Sky Volume は、事前計算ステップを一部削除することで最適化されており、パフォーマンスへの影響を最小限に抑えながらリアルタイムで空を更新できます。オゾン層のモデルが新たに加わり、地平線付近の空の色精度が向上し、よりリアルな夕暮れの表現が可能になっています。

オゾン層は地平線付近の色精度を高めます。

また、空気遠近法では、山や雲など遠くにある物体を見た際の、大気中の微粒子による光の吸収をシミュレートできるようになりました。



Physically Based Sky Volume (物理ベースの空のボリューム) は大気散乱をシミュレートできます。

Volumetric Clouds:雲 はファークリップ面によって切り取られなくなり、地球の曲率の制御は Visual Environment 設定に集約されました。

水システム:水システム は、水面上のゲームオブジェクトの Transform (平行移動、回転、負のスケーリングなど) に対するサポートを強化します。Unity 6.1 では、Shader Graph に Water Decal (水デカール) ターゲットが導入され、泡や水の横方向のデフォーメーション (波のうねりなど) を表現できます。

Volumetric Fog:ボリュメトリックライティング のパフォーマンスが最適化されました。

パフォーマンスと最適化

動的解像度: Unity 6.1 では、**Spatial-Temporal Post-Processing (STP)** が導入されました。これは、空間アップスケーリング (近くのピクセルからディテールを補正する) と時間アップスケーリング (前のフレームを使用してエッジを微調整する) を組み合わせた、新しい高品質な画像アップスケーリング手法です。

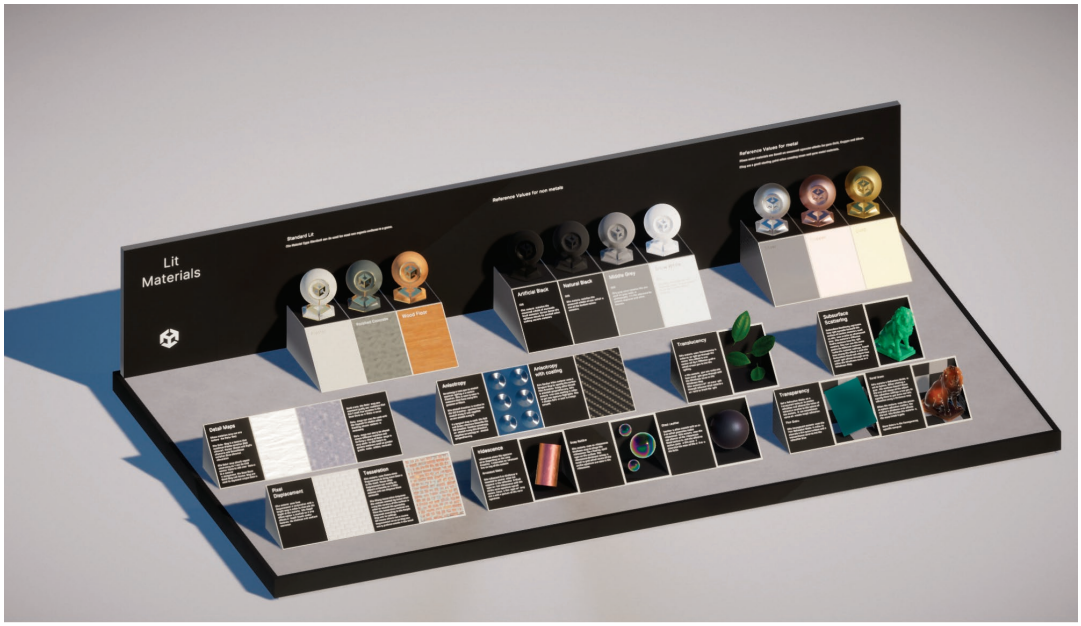
GPU Resident Drawer:新しい GPU Resident Drawer は、Batch Render Group API を活用して、複雑なシーンのレンダリング時のパフォーマンスを大幅に向上させます。

Variable Rate Shading (VRS): Variable Rate Shading (VRS; 可変レートシェーディング) が Custom Pass に対応するようになり、画面の特定領域でシェーディング解像度を動的に調整して、見栄えを維持しつつ GPU の負荷を軽減できるようになりました。

Package Manager のサンプル

Package Manager の HDRP サンプルプロジェクト は、整合性のために見直されました。説明も充実し、パイプライン全体でリソースが共有されています。

- **Lit Material サンプル:**シーン内のすべてのマテリアルで **HDRP/Lit シェーダー** が使用され、さまざまな マテリアルタイプ (Standard、Subsurface Scattering、Anisotropy など) が示されています。各サンプルでは、HDRP がさまざまサーフェスでリアルなライティングをどのように処理しているかを紹介しています。
- **Transparent Material サンプル:**屈折性のあるマテリアル、透明オブジェクトの重ね合わせ、透明なマテリアルによるシャドウなど、HDRP が透明度を処理する方法を示します。ラスタライゼーション、レイトレーシング、パストレーシングを切り替えて、さまざまなレンダリング手法を探りましょう。
- **Volumetric サンプル:**ボリュメトリックフォグ、3D テクスチャ、Shader Graph の効果、ブレンドモードを紹介しています。サンプルでは、フォグエフェクト、プロシージャルノイズ、スモーク要素を再現する方法を示しています。
- **Environment サンプル:**島の地形と海の水システムの上にいくつかの異なる雲を設定して表示します。
- **Water サンプル:**水の新しいサンプルシーン **Cave** が含まれています。このシーンでは、Custom Pass と Local Volumetric Fog (ローカルボリュメトリックフォグ) が水システムのコースティクスバッファをサンプリングし、水の高度なレンダリング手法を示しています。**Rolling Wave** シーンでは、波のうねりなどの効果を可能にする水平方向のデフォーメーションの実装方法を示しています。



Package Manager にサンプルパッケージがあります。

レンズフレア

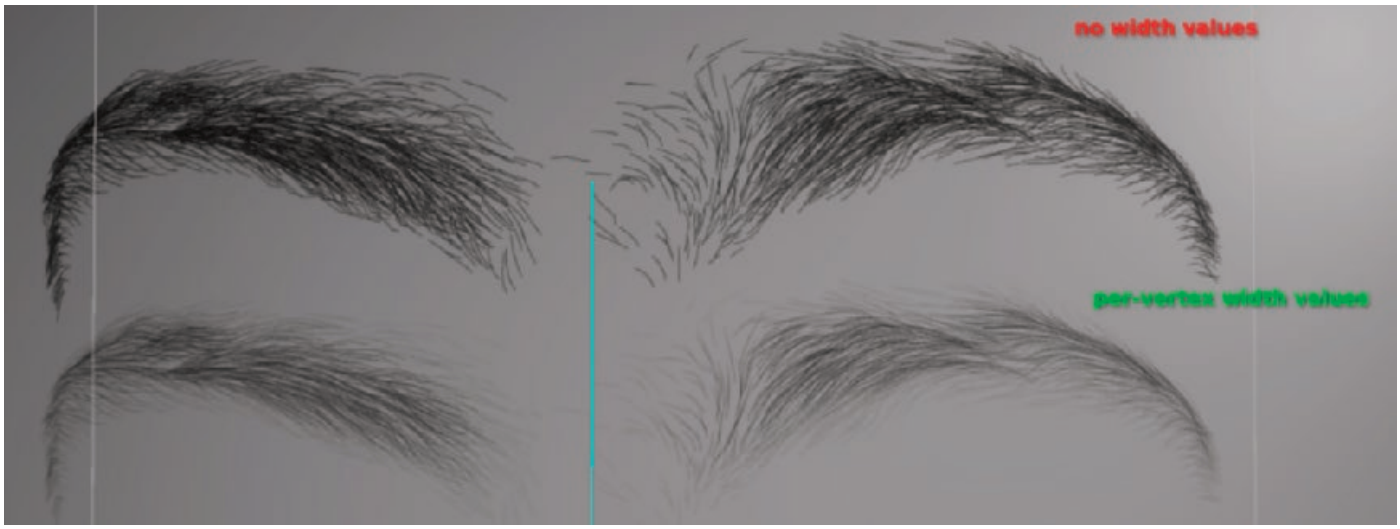
Unity 6 では、レンズフレア がいくつかの新機能によって強化され、視覚的忠実度とワークフローが改善されました。

- **XR サポートの改善:**レンズフレアは XR アプリケーションとの互換性が向上し、さまざまなデバイスで一貫したフレアエフェクトを実現します。
- **リング型プロシージャル形状:**新しいリング状のプロシージャルフレアにより、明るい光源で役立つ同心円フレアを作成できます。
- **再帰的なレンズフレアアセット:**レンズフレアアセットを他のレンズフレア内の要素として使用できるようになり、より複雑で多層的なフレアエフェクトが可能になりました。
- **放射状グラデーションのサポート:**プロシージャル形状で放射状グラデーションがサポートされ、より自然なフレア表現が可能になりました。
- **ライトオーバーライド機能:**新しいライトオーバーライド機能により、1 つの光源で複数のフレアに影響を及ぼすことができます。その結果、複数のライトに対するレンズフレアエフェクトの設定が簡素化されます。

Unity 6 ではレンズフレアが強化されています。

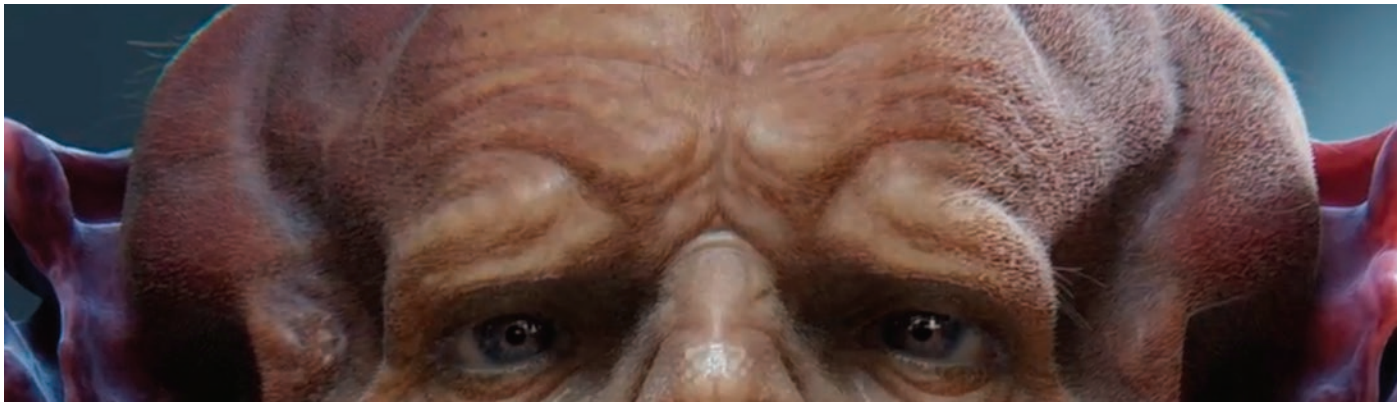
髪

High-Quality Line Rendering (高品質なラインレンダリング) の Volume オーバーライド により、頂点ごとに髪の幅をセンチメートル単位で定義できるようになり、制御性と精度が向上しました。



頂点ごとの幅の値により、制御性と精度が向上しています。

0.01 cm の均一な幅で産毛を作成する例を示します。



頂点ごとの幅の値を均一な 0.01 cm にすると、産毛を作成できます。

さらに、画面上の占有率に基づく新しい LOD モードでは、ビューポートの可視性に応じて髪の毛束を動的にカリングします。アーティストはアニメーションカーブを使用して毛束の密度を制御できます。X 軸はスクリーンスペースの占有率を表し、Y 軸はレンダリングされる毛束の割合を定義します。これにより、パフォーマンスと見栄えが良くなります。

HDRP のインストール

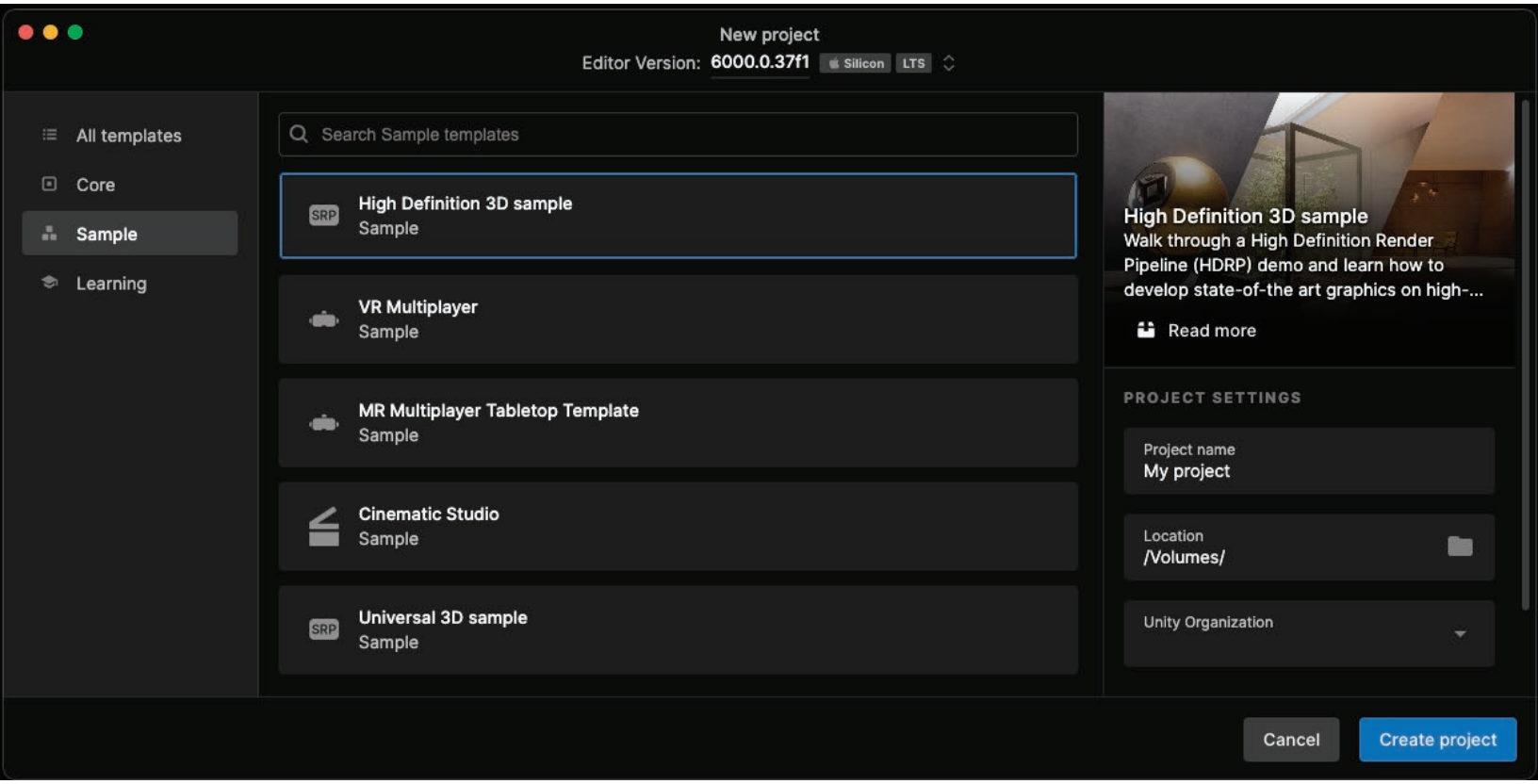
プロジェクトで HDRP を設定するには、**HDRP Template Project** を使用するか、**High Definition RP パッケージ** を手動でインストールします。

HDRP テンプレートには、物理ベースのライティング設定、デカール、ボリューム、高精度のマテリアルが含まれているため、高品質の視覚的なプロジェクトの開始点として最適です。

テンプレートによる HDRP プロジェクトの作成

Unity Hub を開き、**Projects** タブに移動します。**New Project** をクリックし、**Sample** タブに移動して、テンプレートに **High Definition 3D Sample Scene** を選択します。

Project Name を入力し、**Create Project** をクリックします。HDRP とその依存パッケージが自動的にインストールされます。



Unity Hub から HDRP プロジェクトを作成します。

HDRP の手動インストール

既存のプロジェクトに HDRP を追加する場合は、Unity プロジェクトを開き、**Window > Package Manager** に移動します。

Package Manager で、ドロップダウンメニューから **Unity Registry** を選択します。パッケージリストで **High Definition RP** を探して選択します。

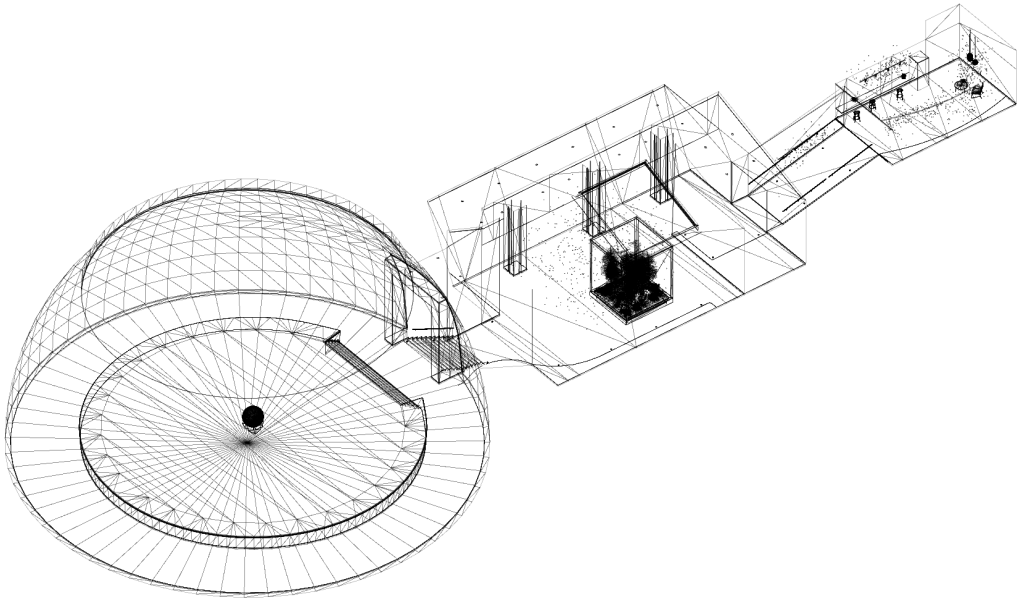
Install をクリックします。エラーが発生した場合は、HDRP Wizard を使用して設定の問題を修正します。

HDRP のプロジェクト設定の詳細については、ドキュメントの Getting started in HDRP を参照してください。

HDRP サンプルシーン

Unity Hub から入手できる 3D サンプルシーンは、HDRP の入門用テンプレートプロジェクトです。この軽量プロジェクトは、いつでも素早くロードしてリファレンスとして活用できるライトウェイトなゲームレベルです。

このガイドでは、このプロジェクトを使用して、HDRP のさまざまな機能を説明します。



3D サンプルシーンの環境を表すワイヤーフレーム

小さな部屋がいくつも繋がった構成になっており、ライティングの設定が異なる 3 種類のエリアがあります。日光を表すディレクショナルライトは、現実世界と同じ強度である 100,000 ルクスに設定されており、各部屋のライティング環境に合わせてカメラの露出が補正されています。
WASD キーとマウスを使用して、FPS Controller を移動させてみましょう。

- **Room 1** は、頭上からの日光に照らされた円形の舞台です。デカールによって、コンクリートの床に埃や水たまりが追加されています。
- **Room 2** は、天窓からボリュメトリックな光の柱が差し込み、ガラスケース内植生には高度なマテリアルが使用されています。
- **Room 3** は、室内の人工照明とエミッシブマテリアルのディスプレイです。

サンプルシーンは 3 つの部屋で構成されています。

HDRP 3D サンプルシーンの詳細については、こちらのブログ記事 を参照してください。テンプレートのシーンについて詳しく説明しています。

その他の HDRP のサンプルコンテンツ

サンプルの学習が終わったら、他のプロジェクトも見てみると役立ちます。

『Time Ghost: Environment』サンプルパッケージには、Unity の映画的なデモ『*Time Ghost*』用に作成された環境の1つが含まれています。Unity Originals チームがこの環境シーン作成のために、DOTS の Entities パッケージと追加ツールをどのように使用したか詳しく見てください。このデモでは、APV ライティング、Scenario Blending、GPU Resident Drawer など、Unity 6 と HDRP の最新の改善内容を活用しています。

Time Ghost: Environment

Time Ghost:Character サンプルパッケージは、『*Time Ghost*』用に作成されたキャラクターを紹介しています。このキャラクターは、クロスをデフォーメーションさせる機械学習モデルを実証するもので、『The Heretic』や『Enemies』で見られる Hair System に基づいて構築されています。『Time Ghost』制作 の詳細については、こちらの Unity Discussions の投稿 を参照してください。

Time Ghost: Character ではクロスを中心に取り上げています。

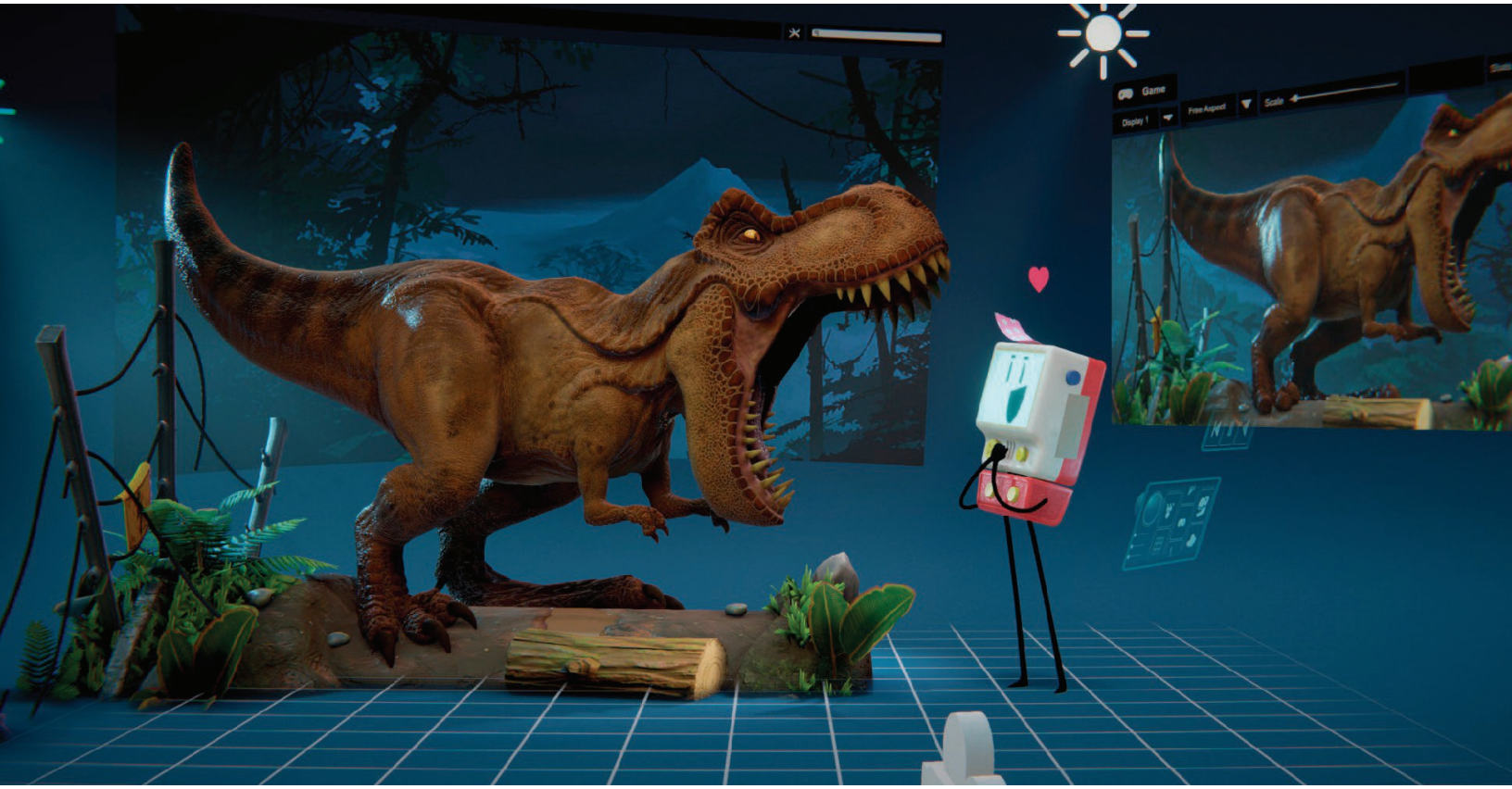
『Enemies』デモプロジェクト には、デジタルな人間のキャラクターの紹介だけではなく、心を揺さぶるアニメーション背景環境も含まれています。このデモは、Probe Volume やレイトレーシング効果を活用し、NVIDIA の Deep Learning Super Sampling (DLSS) のネイティブサポートを統合することで、4K 解像度での動作を可能にしています。

『Enemies』デモは入念に作り込まれた内装が特徴です。

Sponza Palace Atrium は、グラフィックプログラマーやアーティストに広く利用されています。屋内と屋外のエリアの両方を備えており、ライティングのテスト環境として最適です。このバージョンは HDRP でリマスターされています。

Sponza Atrium

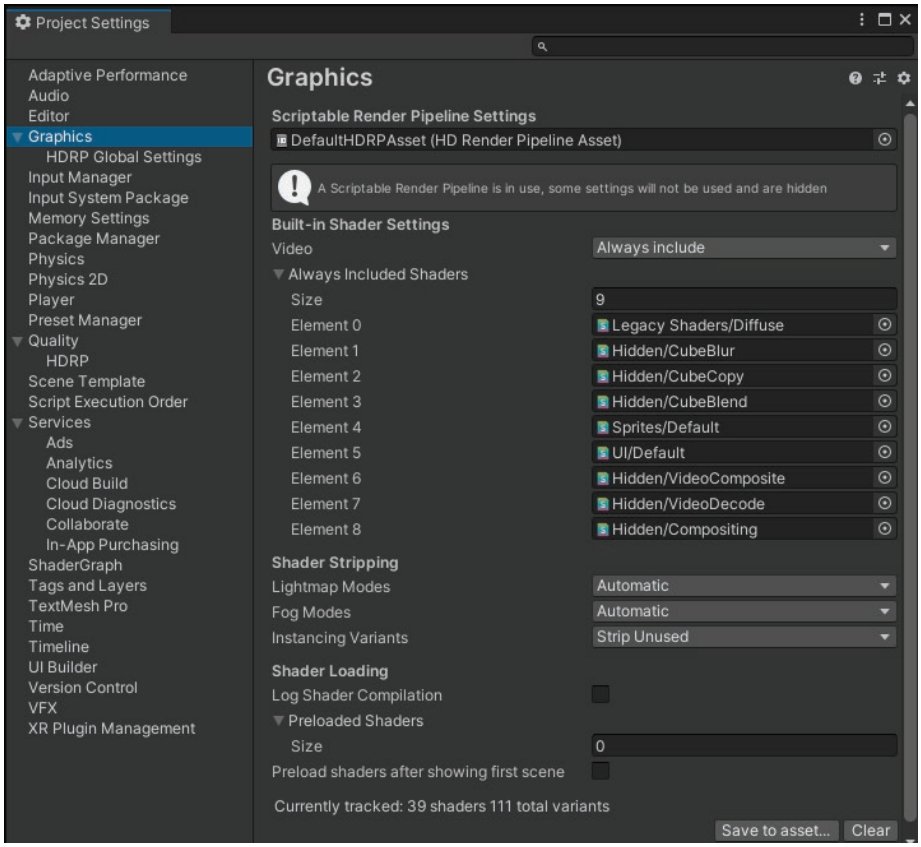
Asset Store から Cinematic Studio Template をインストールして、映画製作の技術やアニメーション映像の作り方を学びましょう。このテンプレートでは、様式化されたレンダリングと写実的なレンダリングの両方を取り入れたユーモラスな短編映画『Mich-L』を使って、ショットの設定とライティングの方法を学ぶことができます。



Cinematic Studio Sample

Project Settings

Project Settings (**Edit > Project Settings**) の **Graphics**、**HDRP Global Settings**、Quality には、欠かせない基本設定がいくつかあります。



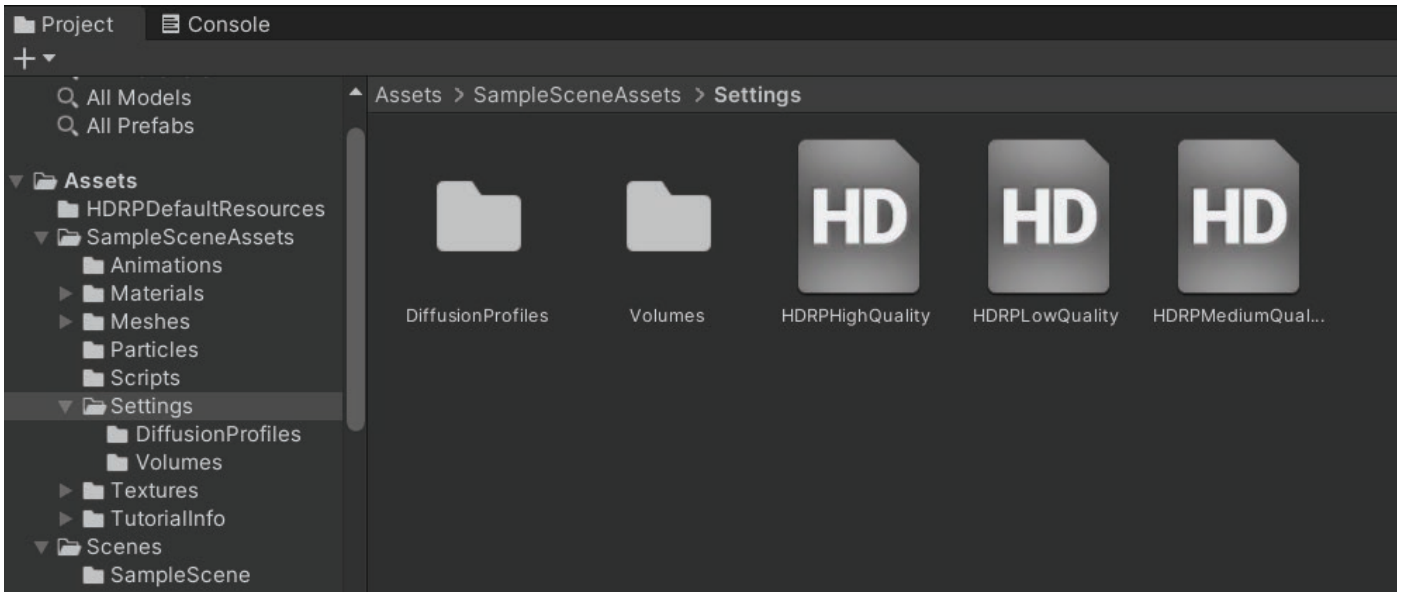
Project Settings

グラフィックス設定

最上部の **Scriptable Render Pipeline Settings** フィールドは、すべての HDRP 設定を保持するディスク上のファイルを指しています。

プロジェクトごとにこのような **パイプラインアセット** を複数使用できます。各アセットが別々の設定ファイルだと考えてください。例えば、各種ターゲットプラットフォーム専用の設定を保管するために使用したり、プレイヤーがランタイムに切り替えることができる複数の画質レベルを設定したりできます。

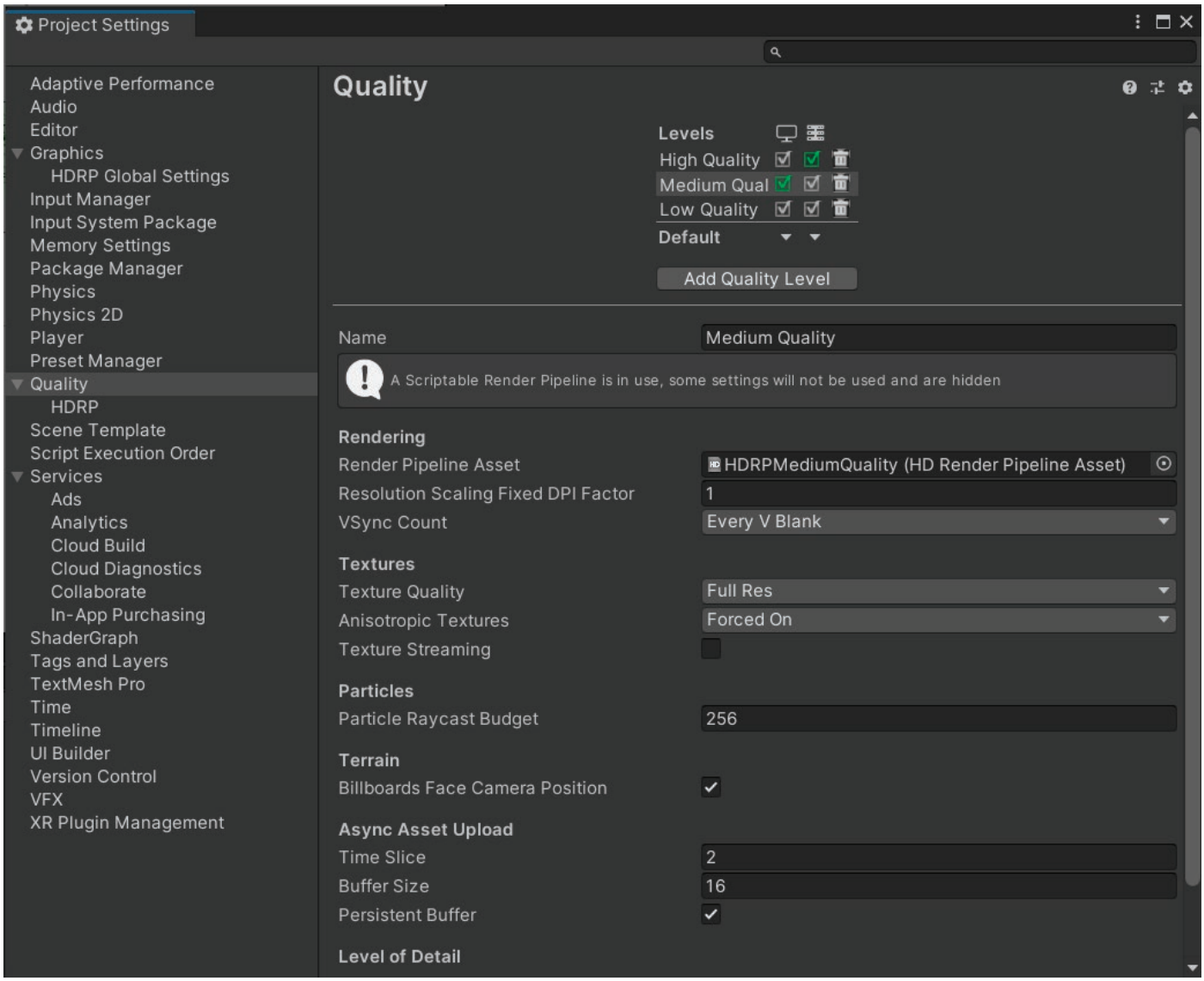
この **3D サンプルシーン**は、Settings フォルダーにあるいくつかのパイプラインアセットから始まります。**HDRPHighQuality**、**HDRPLowQuality**、そして **HDRPMediumQuality** です。
さらに、**HDRPDefaultResources** フォルダーには **DefaultHDRPAsset** があります。



3D サンプルシーンには、低、中、高の品質のパイプラインアセットが含まれています。

品質設定

Quality　設定では、任意のパイプラインアセットを、事前に定義した品質のレベルへ一致させるようにすることができます。最上部の **Level** を選択して、特定の **レンダーパイプラインアセット** を有効にします。これは **Rendering** オプションに表示されます。

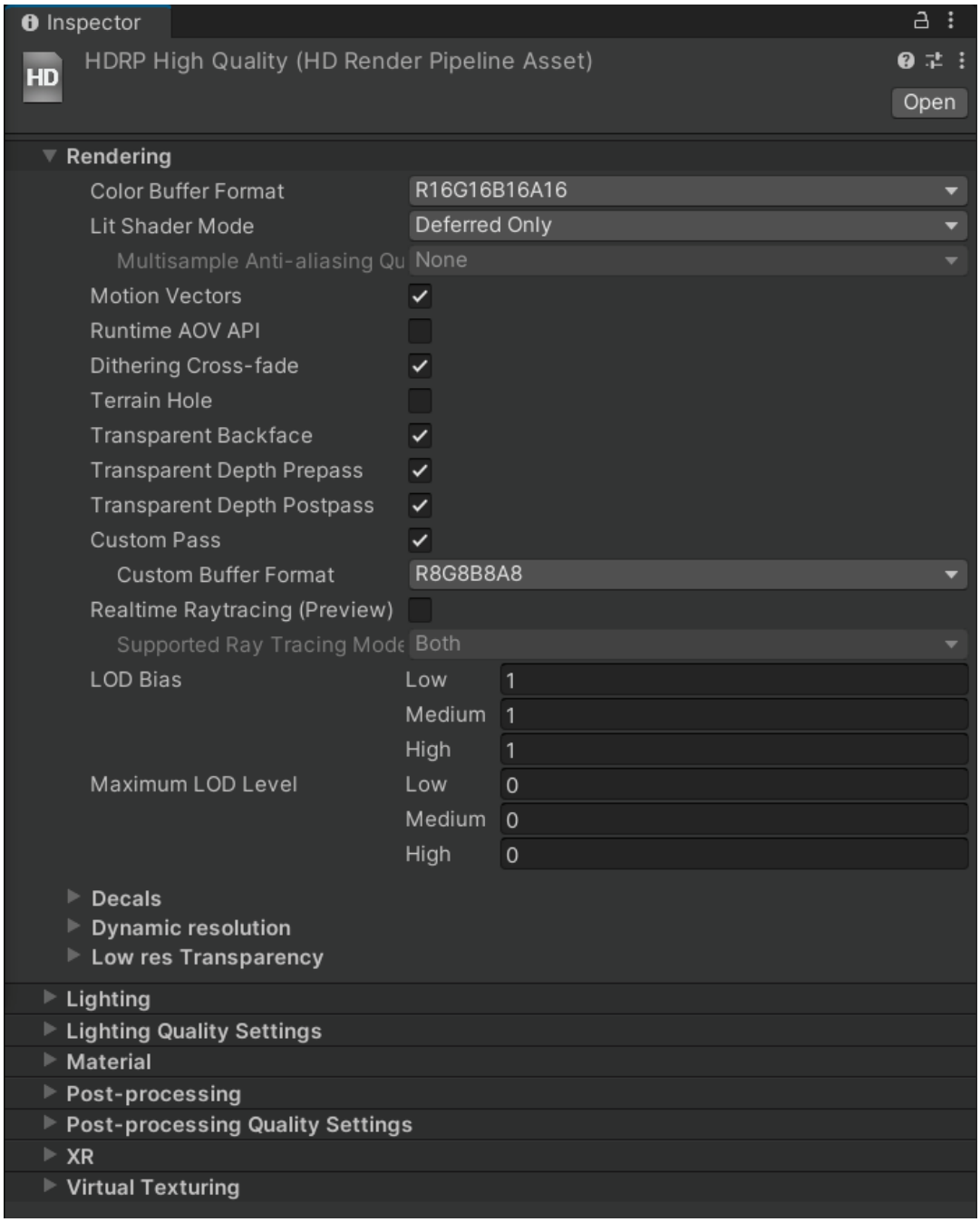


設定画面の最上部の品質レベルでパイプラインアセットをアクティベートします。

デフォルト設定をカスタマイズしたり、品質レベルを追加したりした後で、追加のパイプラインアセットとペアリングできます。

品質レベルは、パイプライン内でアクティブ状態になっている特定のビジュアル機能を表します。たとえば、1 つのアプリケーションに複数のグラフィックスレベルを作成できます。ランタイム時には、プレイヤーがハードウェアに合わせてアクティブな品質レベルを選択できます。

Quality/HDRP のサブセクションで、実際のパイプライン設定を編集できます。Project ウィンドウでパイプラインアセットを選択し、インスペクターで設定を編集することもできます。



パイプラインアセットの編集

HDRP の最適化

パイプラインアセットで多くの機能を有効にすればするほど、消費リソースも増える点にはご注意ください。一般的には、目標とする効果を実現するために必要なものだけを使用するように、プロジェクトを最適化します。不要な機能がある場合は、オフにすることで、パフォーマンスを高めてリソースを節約できます。

不要な場合は無効にしても問題ない機能の典型例は次のとおりです。

— **HDRP Asset** 内:デカール、低解像度透明度、透明バックフェイス/デプスプリパス/デプスポストパス、SSAO、SSR、コンタクトシャドウ、ボリュメトリック、サブサーフェススキャタリング、歪曲

— カメラの **Frame Settings** (メインカメラ、リフレクションなどの統合的な効果に使われるカメラ、またはカスタム効果に使われる追加のカメラ) 内: 屈折、ポストプロセス、ポストプロセス後、トランスミッション、リフレクションプローブ、平面リフレクションプローブ、ビッグタイルプリパス

パフォーマンス向上のための HDRP 設定について、詳しくはこちらのブログ記事をご覧ください。

HDRP グローバル設定

HDRP Global Settings（HDRP グローバル設定）セクション（またはバージョン 12 より前の **HDRP Default Settings**）は、プロジェクトのベースライン設定を決定します。カメラの位置に応じて、ローカルまたはグローバルの Volume コンポーネントを配置することで、シーン内のこれらの設定を上書きすることができます (下記の Volume を参照)。

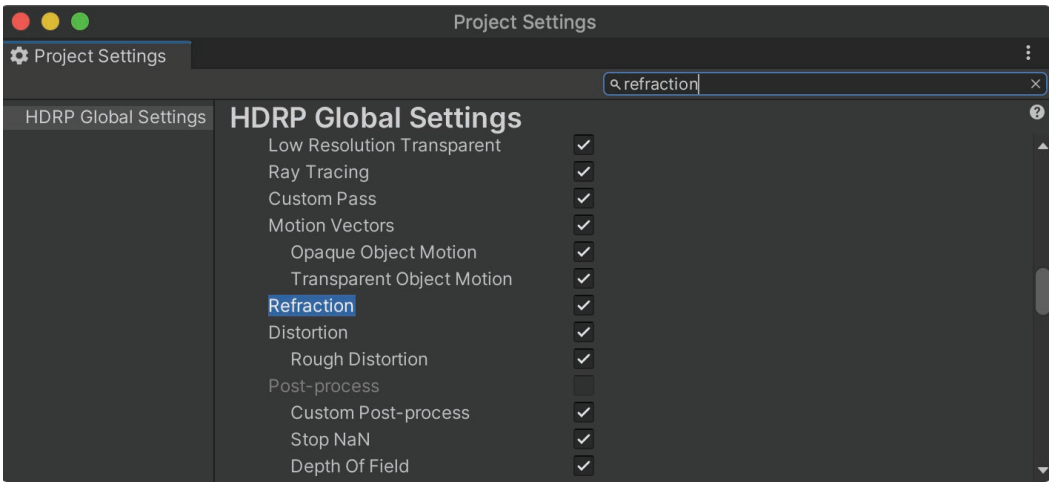
Global Settings は、最上部のフィールドで定義されている個別のパイプラインアセットに保存されます。ここでデフォルトのレンダリングおよびポストプロセスのオプションを設定します。

HDRP 機能の有効化

プロジェクトを進めている時に、場合によっては、**グローバル**設定へ戻って、特定の機能を切り替えてオンまたはオフに設定することが必要になる場合もあります。一部の機能は、**HDRP グローバル設定** 内の対応するチェックボックスをオンにしないとレンダリングされません。

レンダリングのパフォーマンスやメモリ使用量に悪影響を与える可能性があるため、必要な機能のみを有効にしてください。また、用途に応じて **Volume Profiles** に表示される設定もあれば、**Frame Settings** に表示される機能もあります。

HDRP のさまざまな機能について調べる際は、Project Settings の右上にある検索フィールドを活用しましょう。表示対象が関連するパネルのみに絞られるようになり、検索用語が強調表示されます。



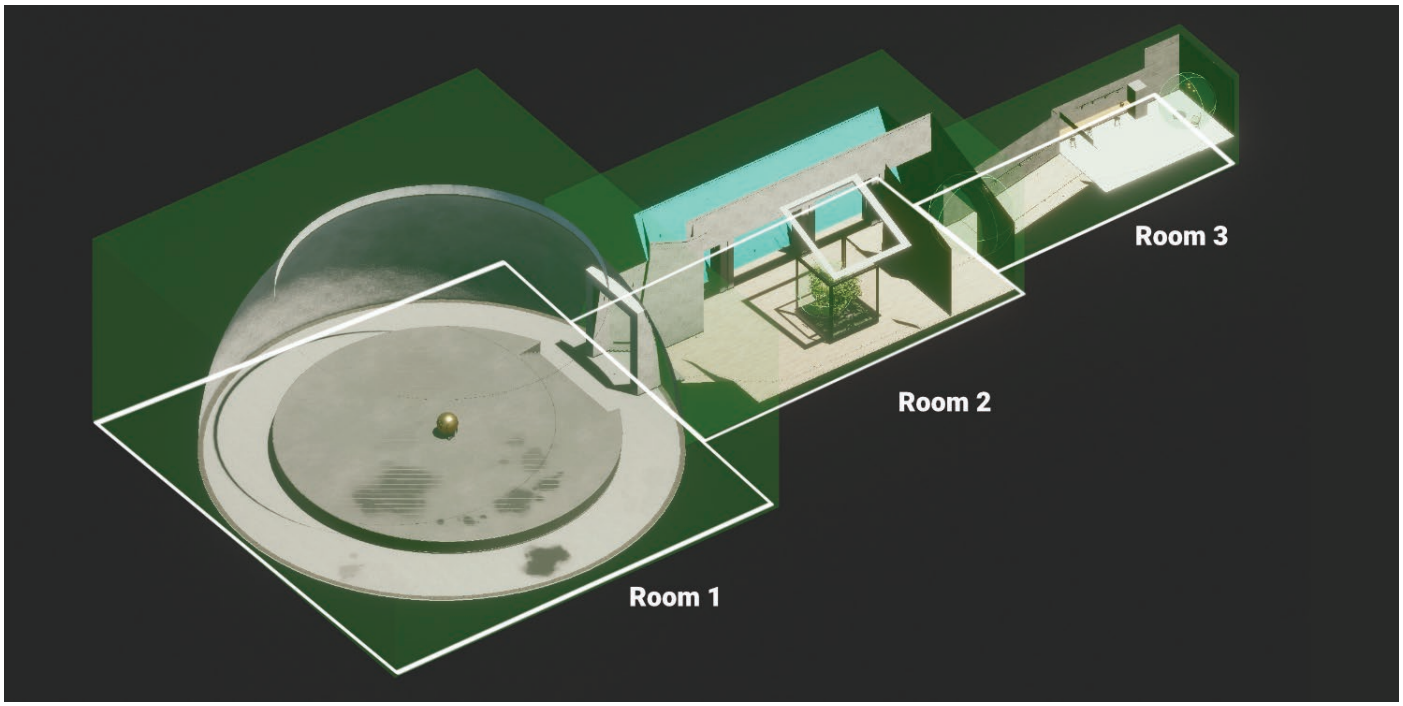
HDRP 機能を検索します。

HDRP グローバル設定で機能を有効化しても、すべてのカメラでいつでもレンダリング可能になるわけではありません。**Projects Settings > Quality** で Quality レベルが選択されているレンダーパイプラインアセットが、その機能をサポートしていることを確認することも必要です。

例えば、カメラがボリュメトリッククラウドをレンダリングできるようにするには、**HDRP Global Settings > Frame Settings > Camera > Lighting** と、現在有効なレンダーパイプラインアセットの **Lighting > Volumetrics** の両方で切り替える必要があります。

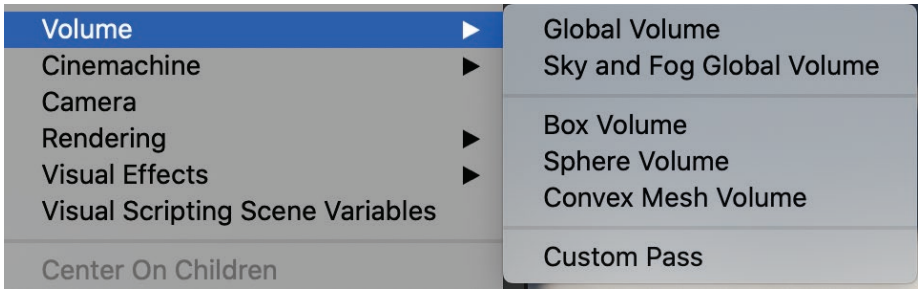
ボリューム

HDRP は **Volume フレームワーク** を使用します。このシステムでは、シーンを分割し、カメラの位置に合わせて特定の設定や機能を有効化できます。たとえば、この HDRP テンプレートレベルは3 つに分かれており、それぞれに独自のライティング設定があります。その形で、各部屋も別々のボリュームが内包しています。



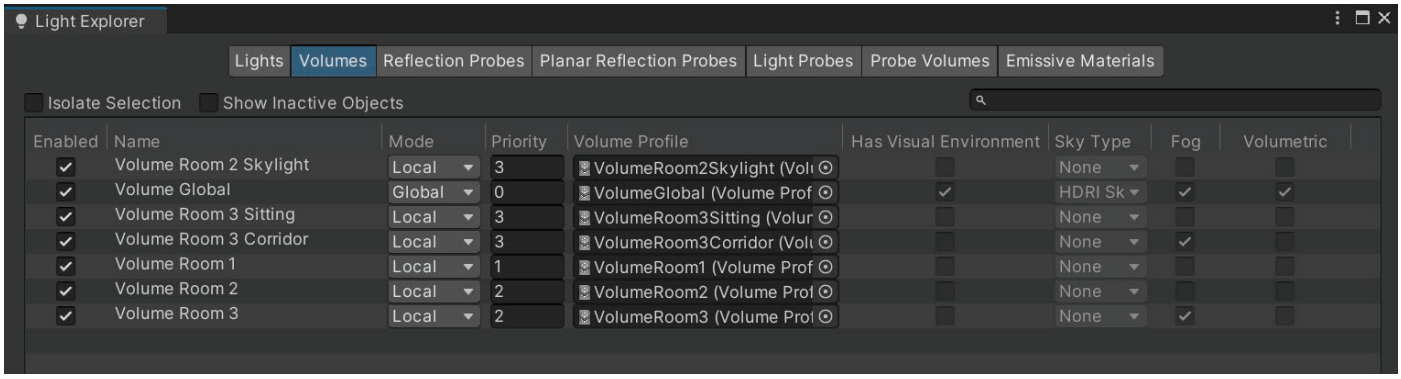
それぞれにライティング条件が異なるスペースをボリュームが覆っています。

ボリュームとは、Volume コンポーネントが含まれたプレースホルダーオブジェクトに過ぎません。ボリュームは、**GameObject > Volume** メニューから、プリセットを選択することで作成できます。または、適切なコンポーネントを含めたゲームオブジェクトを手動で作成します。



プリセットを使用してボリュームオブジェクトを作成

ボリュームコンポーネントは任意のゲームオブジェクトに追加できるため、階層から見つけるのが困難になる場合があります。Light Explorer **(Window (ウィンドウ) > Rendering (レンダリング) > Light Explorer (ライトエクスプローラー) > Volumes (ボリューム))** を使用すると、ロードされているシーンのボリュームを見つけやすくなります。このインターフェースを使用して手早く調整することができます。

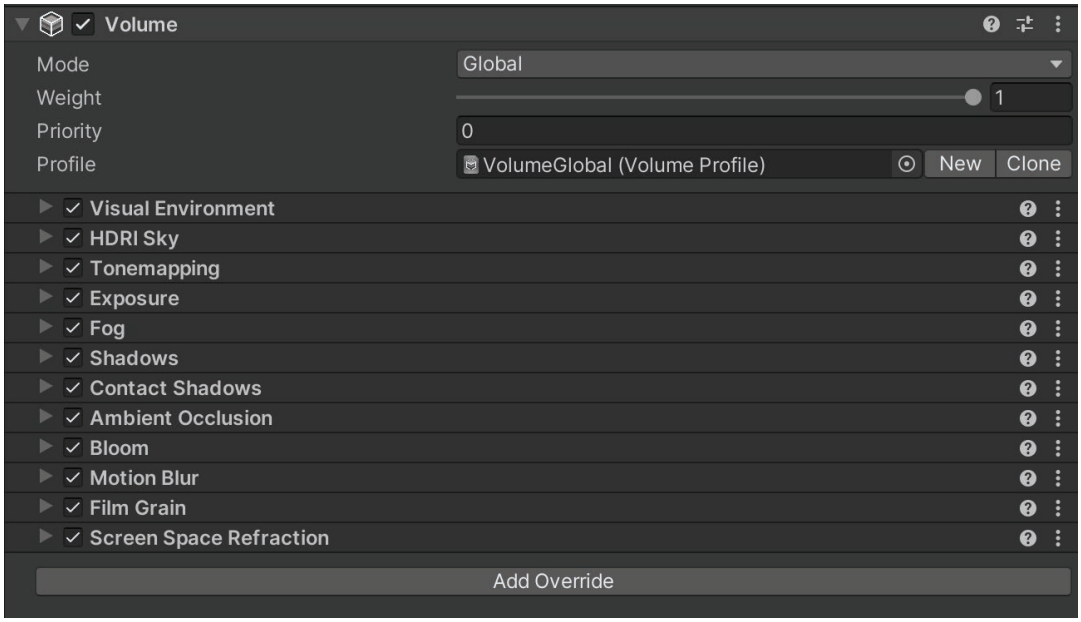


Light Explorer には、開いているシーン内のすべてのボリュームが一覧表示されます。

ローカルとグローバル

コンテキストに合わせて、Volume コンポーネントの **Mode** を **Global** か **Local** に設定します。

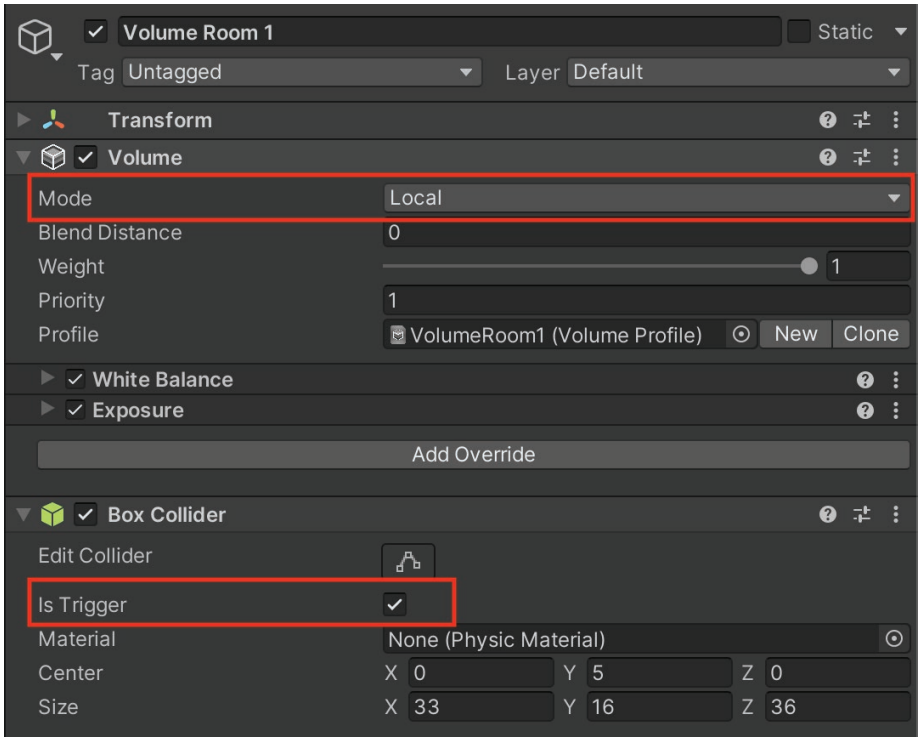
グローバルボリュームは、境界のない“汎用“ボリュームとして使用できます。その場合、シーン内のすべてのカメラに影響します。HDRP テンプレートシーンでは、レベル全体の HDRP 設定のベースラインが VolumeGlobal によって定義されます。



グローバル Volume オーバーライド

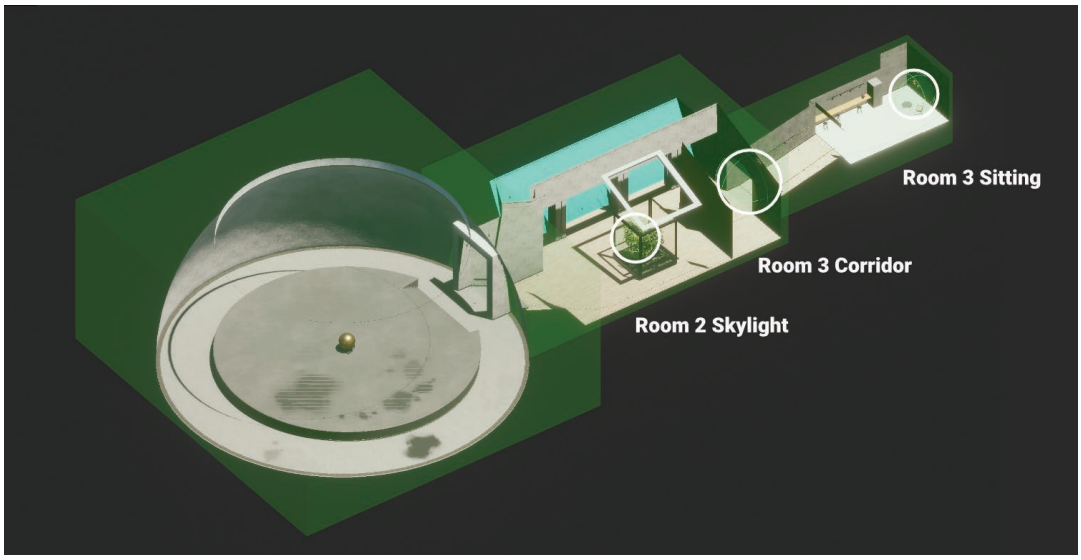
ローカルボリュームでは、設定を適用する限定的なスペースを定義します。コライダーコンポーネントを使用して境界を決定します。コライダーによって FPS プレイヤーコントローラーなどの物理ボディの動きが妨げられないようにするには、**IsTrigger** を有効にします。

このテンプレートシーンでは、各部屋に、グローバル設定をオーバーライドする BoxCollider を含むローカルボリュームがあります。



各部屋に、コライダーが IsTrigger に設定されたローカルボリュームがあります。

Room 2 には、ガラスケースの横の中央の明るい場所に小さな球面ボリュームがあります。同様に、Room 3 には、入口の通路と、つり下げの照明の下の椅子があるエリアに、より小さなボリュームがあります。



特殊なライティング条件では小さいボリュームを使用します。

SampleScene では、ローカルボリュームによって White Balance、Exposure、Fog がオーバーライドされます。明示的にオーバーライドされない要素は、すべてグローバルのデフォルト設定にフォールバックされます。

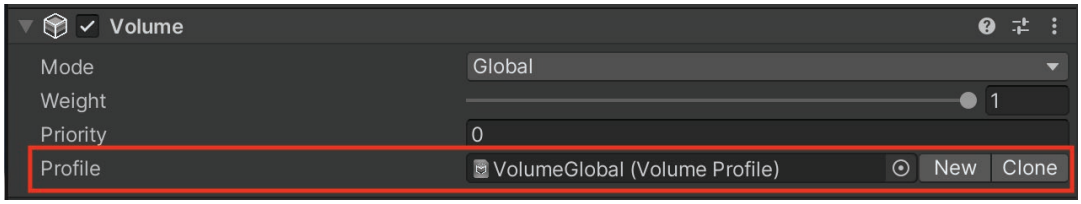
シーン内でカメラが動く際は、グローバル設定をオーバーライドするローカルボリュームにプレイヤーコントローラーが接触するまでは、グローバル設定が適用されます。

パフォーマンスに関するヒント

ボリュームは大量に使用しないようにしてください。各ボリュームの評価 (ブレンディング、立体化、オーバーライドの計算など) には、CPU コストが伴います。

ボリュームプロファイル

Volume コンポーネント自体には、実際のデータは含まれません。その代わり、Volume コンポーネントは Volume Profile を参照します。これは、シーンをレンダリングするための HDRP 設定を含むディスク上の ScriptableObject アセットです。新しいボリュームプロファイルを作成するには、Profile フィールドで、**New** または **Clone** ボタンを使用します。



Profile フィールドでボリュームプロファイルを切り替えるか、新しく作成します。

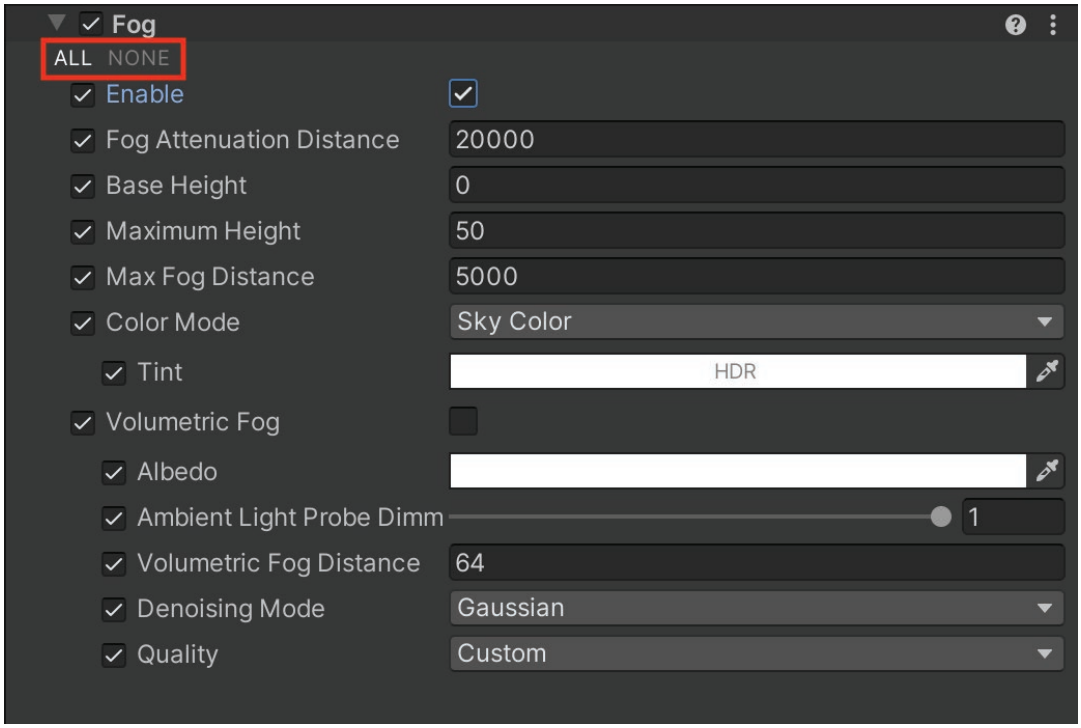
保存済みの別のプロファイルに切り替えることもできます。ボリュームプロファイルをファイルとして使用することで、以前の設定を再利用したり、ボリューム間でプロファイルを共有したりしやすくなります。

再生モード中にボリュームプロファイルに加えられた変更内容は、再生モードを終了した後も残ります。

Volume オーバーライド

各 ボリュームプロファイル では最初、一連のプロパティがデフォルト値に設定されています。この値を編集するには、Volume オーバーライド を使用し、個々の設定をカスタマイズします。たとえば、Volume オーバーライドでは、ボリュームのフォグ、プロセッシング、露出を変更できます。

ボリュームプロファイル を設定したら、**Add Override** をクリックし、プロファイル設定をカスタマイズします。フォグのオーバーライドは次のようになります。



Volume オーバーライドとしてのフォグの例

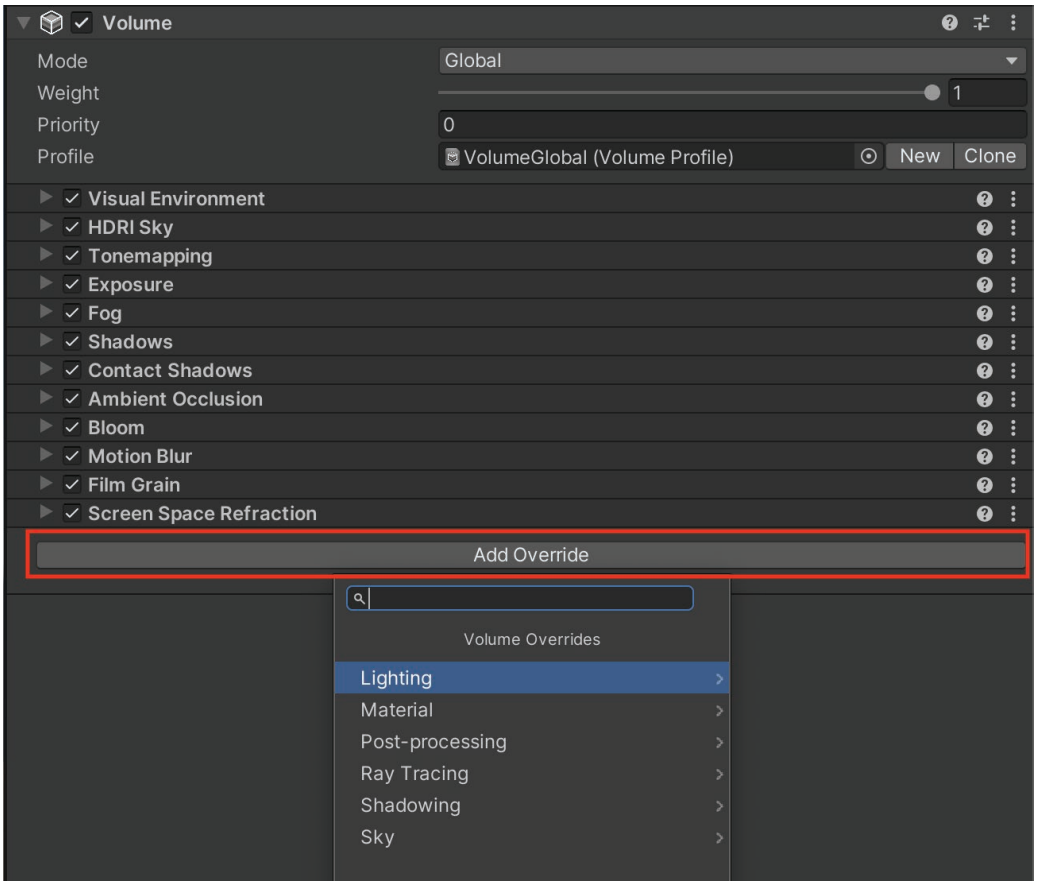
Volume オーバーライドの各プロパティーでは、左側にチェックボックスがあり、そこで対象のプロパティーの編集を有効化できます。ボックスを無効のままにすると、HDRP では、ボリュームの初期値が使用されます。

各ボリュームオブジェクトには、複数のオーバーライドを設定できます。それぞれのオーバーライドで、必要に応じてプロパティーを編集できます。左上の **All** または **None** のショートカットを使用すると、オンとオフを一括で切り替えることができます。

オーバーライドのワークフロー

オーバーライドの追加は、HDRP の重要なワークフローです。プログラミングの継承という概念を理解していれば、Volume オーバーライドも理解しやすくなります。

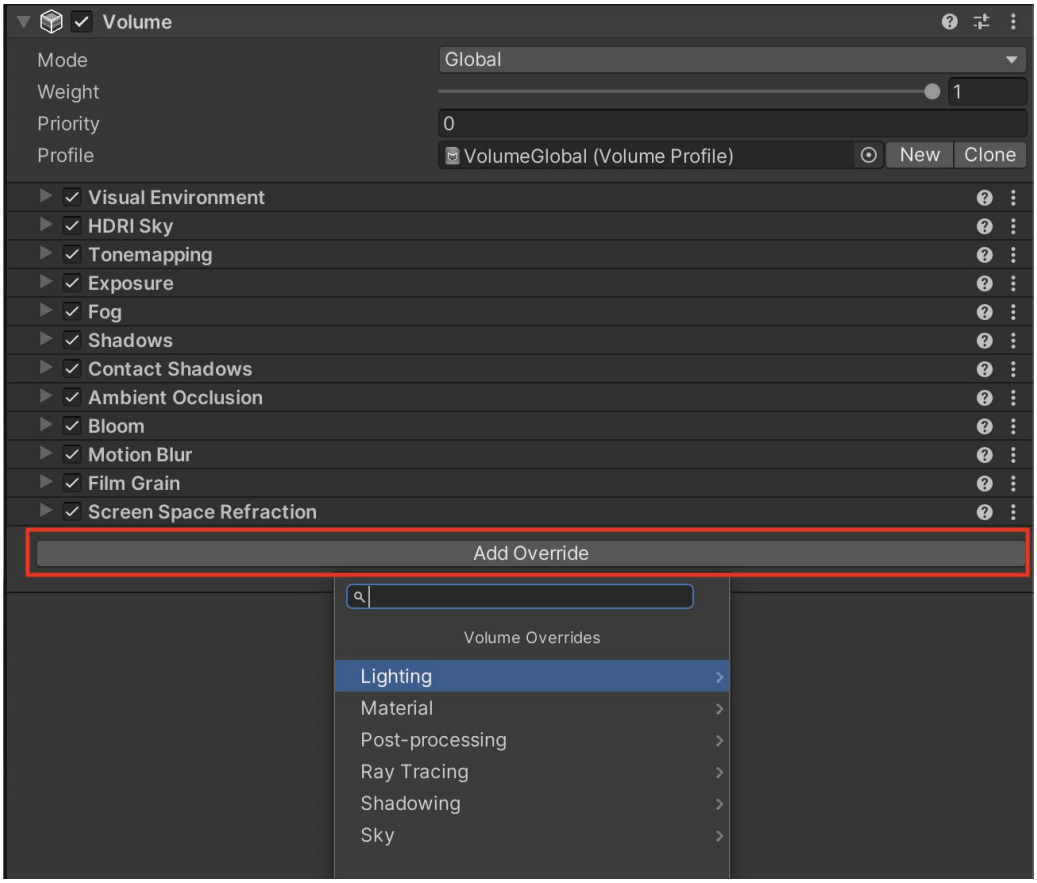
上位レベルのボリューム設定は、下位レベルのボリュームのデフォルト設定として使用されます。ここでは、HDRP のデフォルト設定がグローバルボリュームに渡されます。それがローカルボリュームの “ベース“ として使用されます。



Volume オーバーライドを使用して HDRP 機能を追加

グローバルボリュームによって、HDRP のデフォルト設定がオーバーライドされます。次に、ローカルボリュームによってグローバルボリュームがオーバーライドされます。**Priority**、**Weight**、**Blend Distance** (概要は後述) を使用して、ボリュームの重複に起因する競合を解消します。

特定の Volume コンポーネントの現在の値をデバッグするには、Rendering Debugger (レンダリングデバッガー) の Volume タブを使用します。



ボリュームのデバッグ

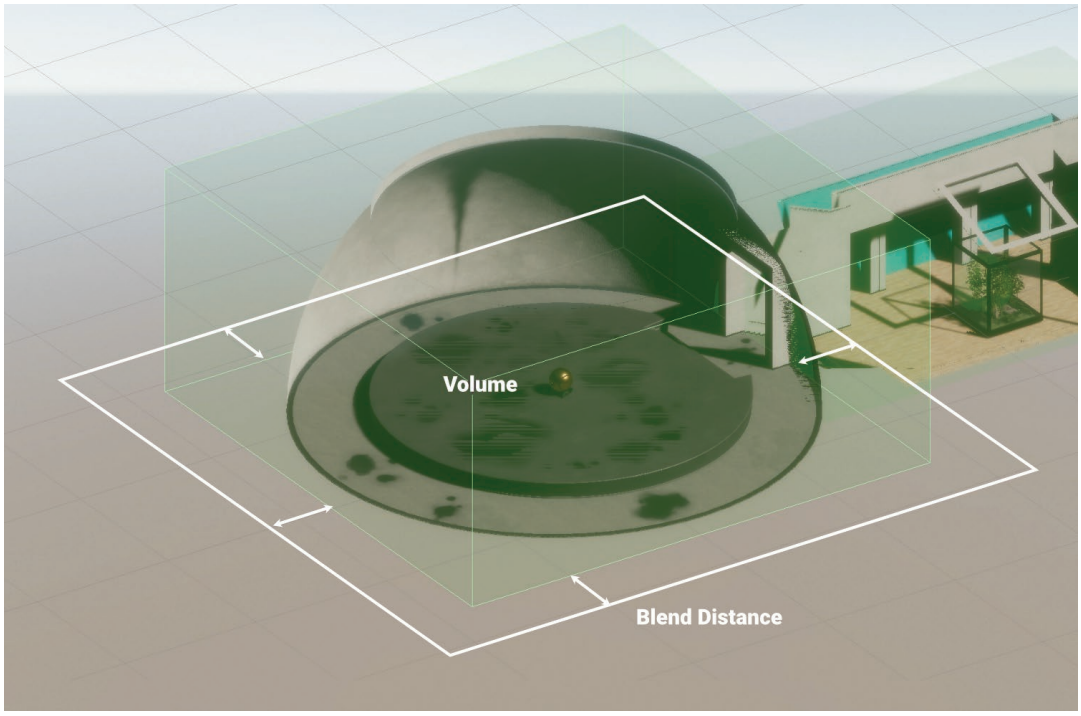
詳細については、HDRP ドキュメントの Volume ドキュメントのページ を参照してください。

ブレンディングと優先度

レベルごとに複数のボリュームが必要な場合もあるため、HDRP ではボリューム間のブレンドが可能になっています。これにより、ボリューム間の遷移がスムーズになります。

ランタイム時、HDRP はカメラ位置を基に、最終的な HDRP 設定に影響するボリュームを判断します。

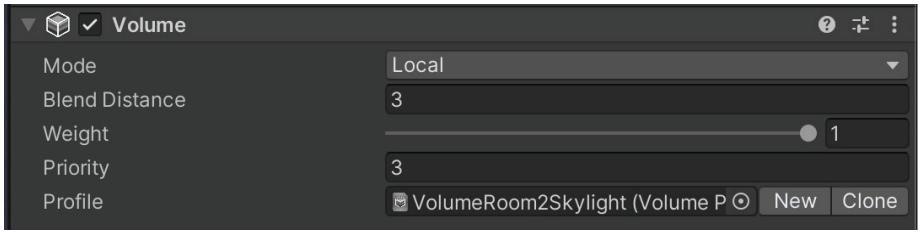
Blend Distance によって、ボリュームのコライダー外にどの程度離れたらフェードオンまたはフェードオフを開始するかを指定します。Blend Distance の値を 0 に設定すると即座に遷移が開始されます。正の値を設定すると、カメラが指定された範囲内に入ったときに Volume オーバーライドによってブレンディングが開始されます。



Blend Distance を使用して、ボリューム周辺の遷移ゾーンを定義します。

ボリュームフレームワークは柔軟性が高く、必要に応じてボリュームとオーバーライドを組み合わせることができます。同一スペース内に複数のボリュームが重複している場合、**Priority** に基づいて、優先されるボリュームが決定されます。この値が大きいほど優先度が高くなります。

通常は、あいまいになるのを防ぐために Priority を明示的に設定します。明示的に設定しない場合、Priority が同じであれば作成された順番で優先順が決定されますが、それによって予期しない結果が生じることがあります。



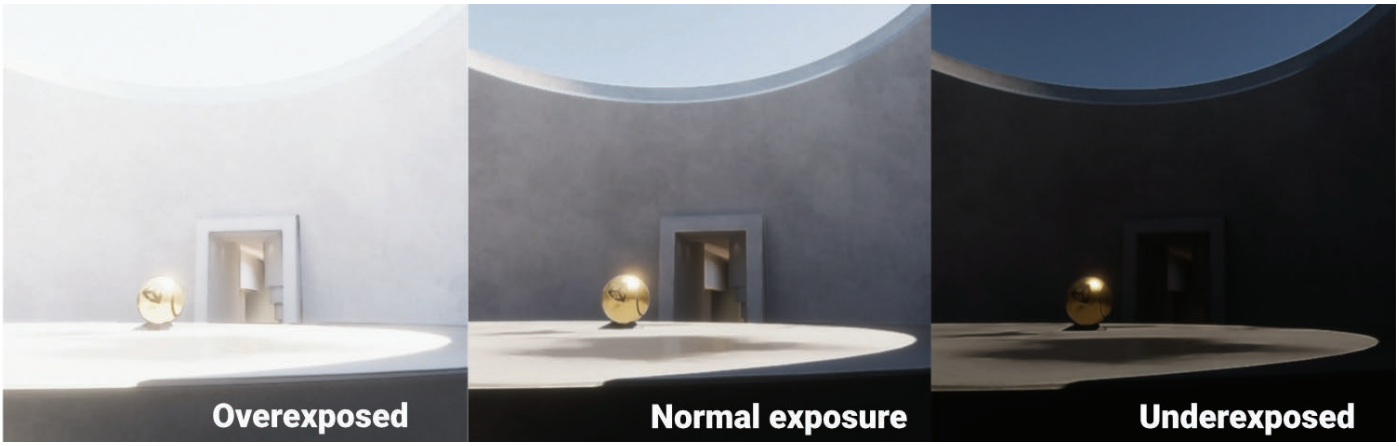
ローカルボリュームが重複する場合は、Blend Distance、Weight、Priority を使用します。

露出

HDRP は、現実世界のライティングモデルを使用して各シーンをレンダリングします。そのため、従来の写真術で使われてきたプロパティーと類似するプロパティーが多数あります。

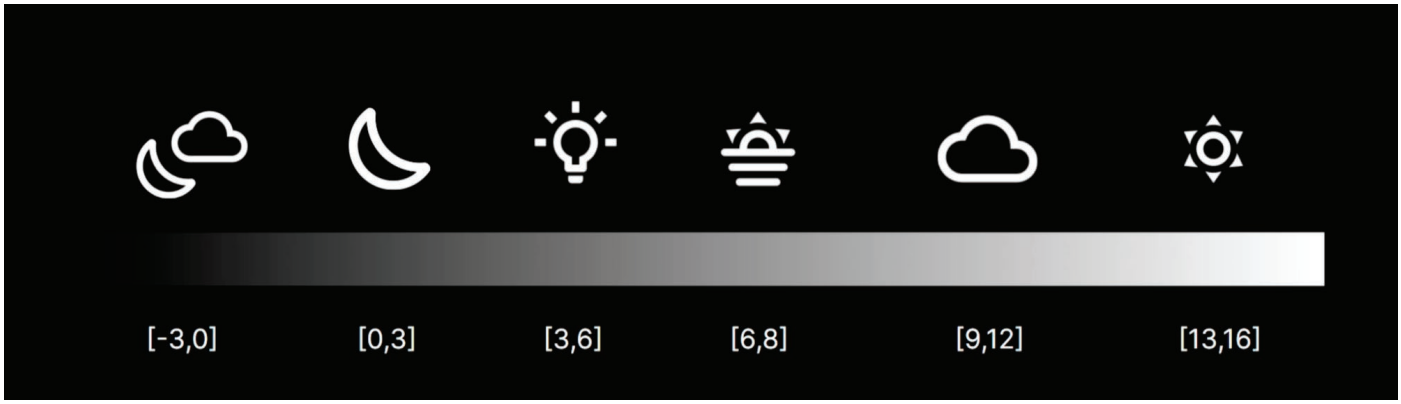
露出値を理解する

露出値 (EV) は、カメラの シャッタースピード と F 値 (レンズの開口部の大きさ、つまり絞り) の組み合わせによって決まる数値です。明度を最適化し、シャドウとハイライトの両方でディテールを高精度に捉えるためには、露出を適切に設定する必要があります。そうしないと、画像の露出がオーバーまたはアンダーになり、好ましくない結果につながってしまいます。



露出オーバー、露出アンダー、適正露出の比較。

HDRP における露出範囲は、一般的にこのスペクトルのどこかに当てはまります。



月のない夜から、明るい晴天日までの露出範囲

露出値が大きくなるほどカメラに入る光が少なくなり、より明るい状況に適します。露出値 13 から 16 の範囲は、晴天の日中の屋外に適しています。一方、暗く、月がない夜空には、露出値 -3 から 0 が適しています。

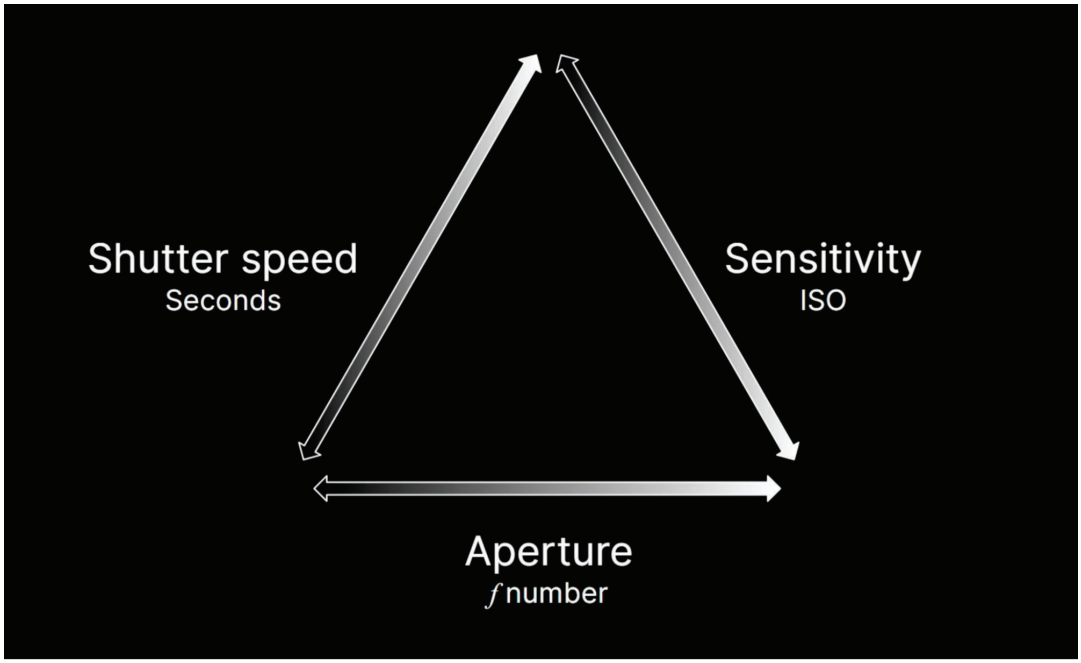
実際のカメラ設定で、次のようなさまざまな要素を変更し、露出値を調整できます。

— シャッタースピード：画像センサーを光にさらす時間

— F 値：絞り/レンズ開口部のサイズ

— ISO：フィルム/センサーの感度

写真家の間では、これは *露出の三角形* と呼ばれています。Unity では、現実のカメラと同様、これらの数値をさまざまな形で組み合わせて同じ露出値を実現できます。



露出の三角形

HDRP では、すべての露出値が EV100 で表現されます。これは感度が ISO100 フィルム の感度に固定されていることを意味します。

露出値の計算式

次の計算式によって、実際の露出値が計算されます。

$$EV = \log_2 \left(\frac{f\,\text{number}^2 / \text{shutter speed}}{\text{ISO} / 100} \right)$$

露出の計算式

これは底を 2 とする対数スケールです。露出値が 1 単位増加すると、レンズに入る光の量が半分に減少します。

HDRP では、実際の画像の露出に合わせることができます。カメラやスマートフォンでデジタル写真を撮影するだけで、その画像からメタデータを取得し、F 値、シャッタースピード、ISO を特定できます。

デジタル写真の Exif データを基に露出を合わせます。

そのうえで、上記の計算式によって露出値を計算します。Exposure オーバーライド（後述）と同じ値を使用すると、レンダリングされる画像は、現実の画像と同じ露出になります。

この方法を使うことで、レベルのライティングを設定する際にデジタル写真を参照情報として使用できます。その画像を完全に再現することが目的ではないかもしれませんが、実際の写真に合わせることによって、勘に頼ることなくライティングを設定できます。

露出のオーバーライド

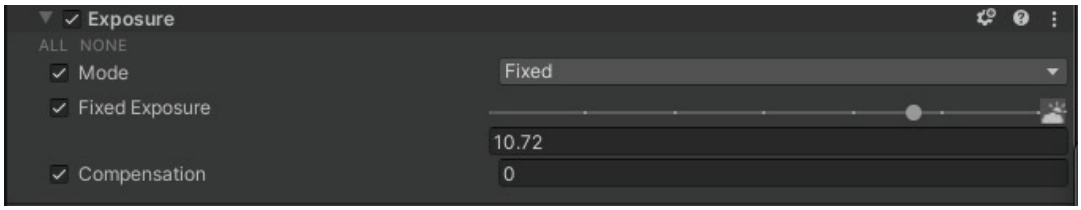
HDRP では、露出は Volume オーバーライドです。ローカルボリュームまたはグローバルボリュームに追加することで利用可能なプロパティを確認できます。

Mode ドロップダウンで、以下のうち1つを選択できます。**Fixed**、**Automatic**、**Automatic Histogram**、**Curve Mapping**、**Physical Camera**

Compensation では、露出を変更、調整できます。これは通常、レンダリングされた画像を上下に "ストップ" (F 値の段) 単位で微調整するために使用されます。

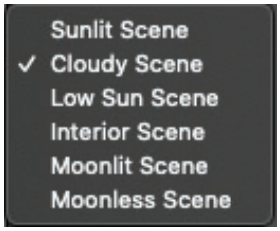
Fixed モード

Fixed モードでは、露出値を手動で設定できます。



Fixed モードの露出

Fixed Exposure スライダーの目盛りが参考になります。また、右側のアイコンをクリックするとプリセットのドロップダウンが表示されます (Sunlit Scene の 13 から Moonless Scene の -2.5 まで)。フィールドの値を任意の数値に設定することもできます。



Fixed モードの露出のプリセット

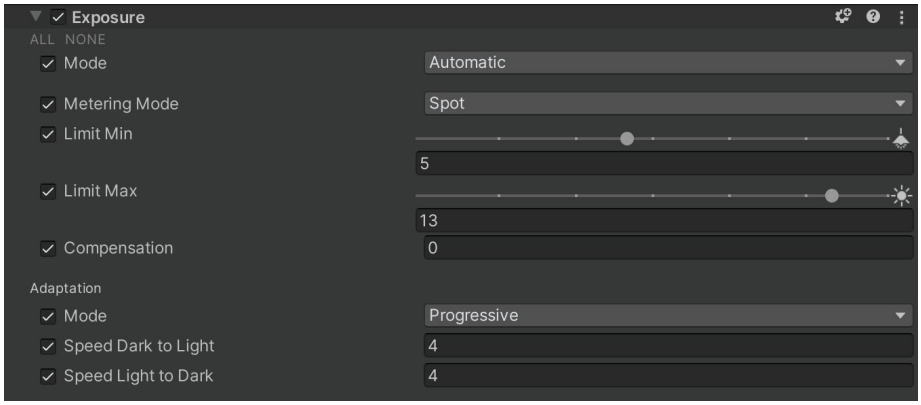
Fixed モードはシンプルですが、柔軟性に劣ります。通常は、ライティングが比較的均一で、1 つの露出値で対応できるボリュームやシーンを使用する場合に効果的です。

Automatic モード

Automatic (自動) モードでは、画面上の明度の範囲に応じて露出が動的に設定されます。この設定は、人間の目が暗さの変化に順応 して黒と認識する基準が変わる仕組みとよく似ています。

Automatic モードは、さまざまなライティングの状況下で有効ですが、カメラが非常に暗い場所や非常に明るい場所を向いたときに、画像の露出オーバーや露出アンダーが意図せず生じる場合もあります。

露出レベルを適切な範囲内に収めるためには、**Limit Min** と **Limit Max** を使用します。テストプレイを行って、レベル全体で上限値が適切な露出の範囲内に収まっているか確認してください。



Automatic モードの露出

Metering Mode をマスクオプションと組み合わせると、フレーム内のどの部分を自動露出に使用するか決定できます。

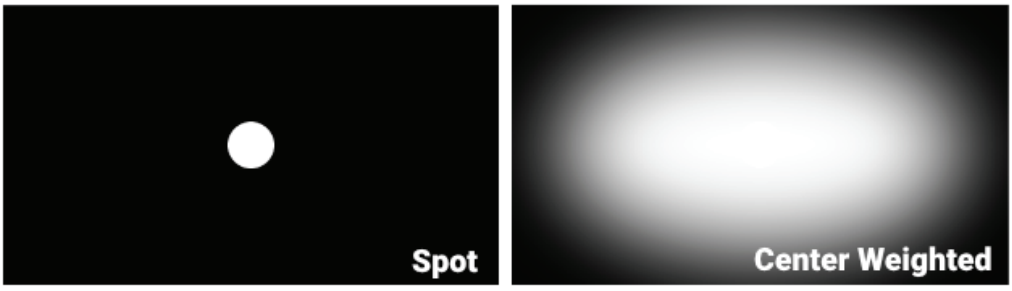
Adaptation モードでは、暗い場所と明るい場所の間をカメラが遷移する際に、自動露出をどのように変化させるかを制御します。スピードを調整するオプションを利用します。人間の目と同じように、非常に暗い場所から非常に明るい場所 (またはその逆) にカメラが移動すると、どこを向いているのか一時的にわからなくなる場合があります。

Metering Mode のオプション

Automatic、Automatic Histogram、Curve Mapping モードでは、Metering Mode を使用して、フレーム内のどの部分を露出の計算に使用するかを制御します。Metering Mode は次のように設定できます。

Average:フレーム全体を使用して露出を測定します。

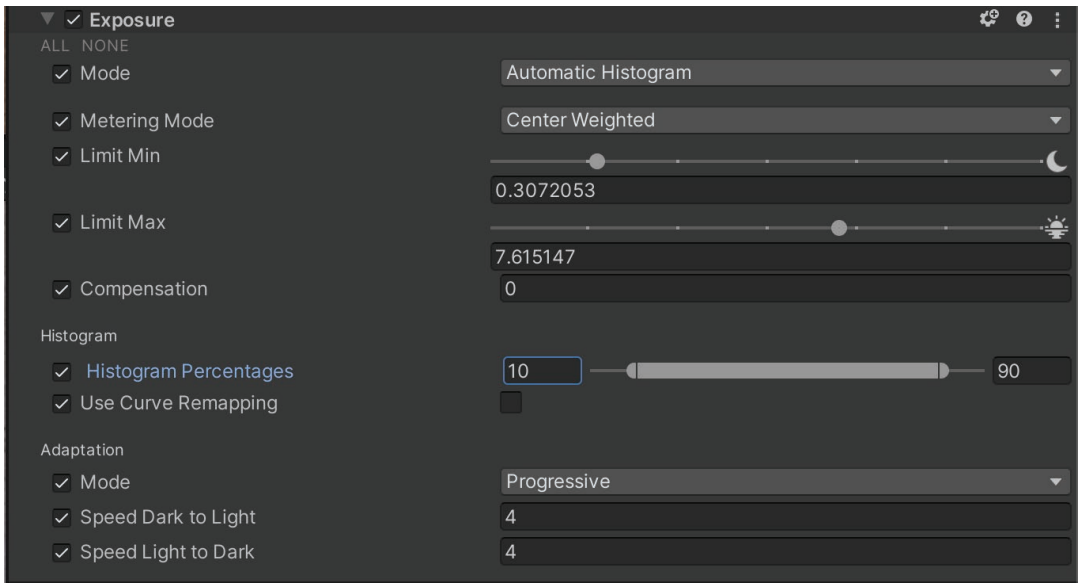
- **Spot:**画面の中央部のみを使用して露出を測定します。
- **Center Weighted**:画面中央のピクセルを優先し、フレームの端に向かうにつれて影響を弱めていきます。
- **Mask Weighted:**指定した画像 (Weight Texture Mask) を使用して、露出を測定する際にどのピクセルを最も重視するかを決めます。
- **Procedural Mask:**プロシージャルに生成されたテクスチャを基に露出を評価します。オプションを使用して、中心位置、半径、柔らかさを変更できます。



Metering Mode の Spot と Center Weighted

Automatic Histogram

Automatic Histogram モードは、Automatic モードをさらに発展させたものです。露出設定時に画像のヒストグラムを計算し、最も暗いピクセルと最も明るいピクセルを無視します。

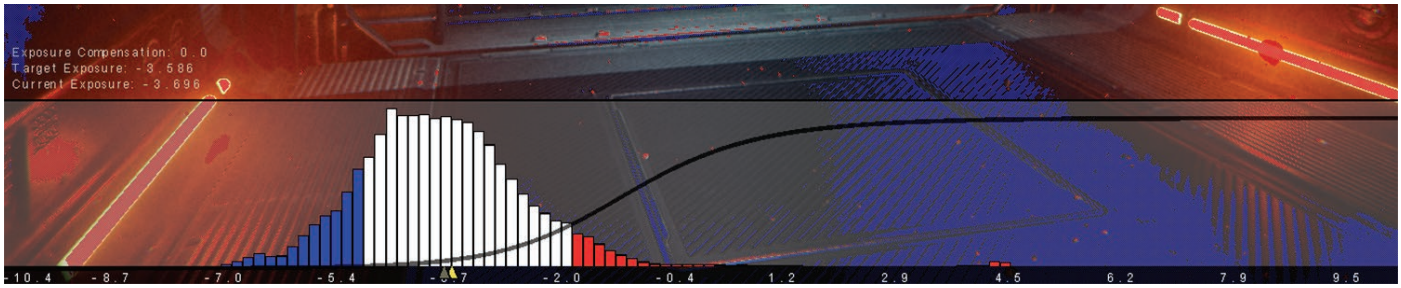


Automatic Histogram モード

露出の計算から非常に暗いピクセルと非常に明るいピクセルを排除することで、フレーム内に極端に明るいピクセルや暗いピクセルがある場合でも、安定した露出を得られます。これにより、発光が強いサーフェスや黒いマテリアルがあったとしても、レンダリング結果で深刻な露出過多や露出不足が生じることはありません。

Histogram Percentages 設定を使用すると、ヒストグラム内の指定したパーセンテージ範囲に含まれないデータをすべて無視します (例えば、ヒストグラムの左端と右端から、最も明るいピクセルと最も暗いピクセルを除外することができます)。

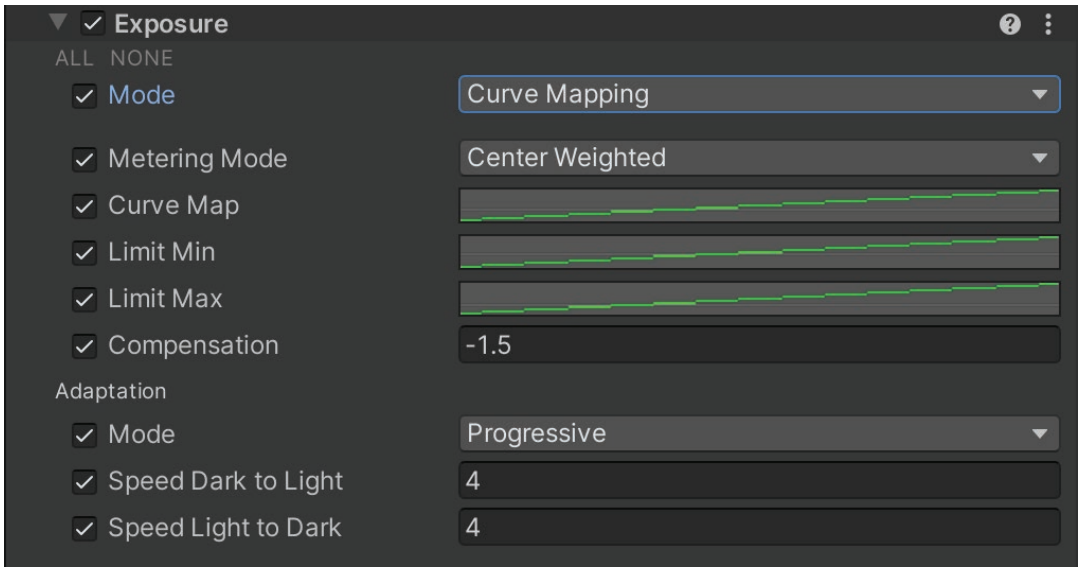
Curve Remapping を使用すると、露出カーブも再マップできます (後述の Curve Mapping を参照)。



Automatic Histogram では、ヒストグラム中央部にあるピクセルを基に露出を計算します。

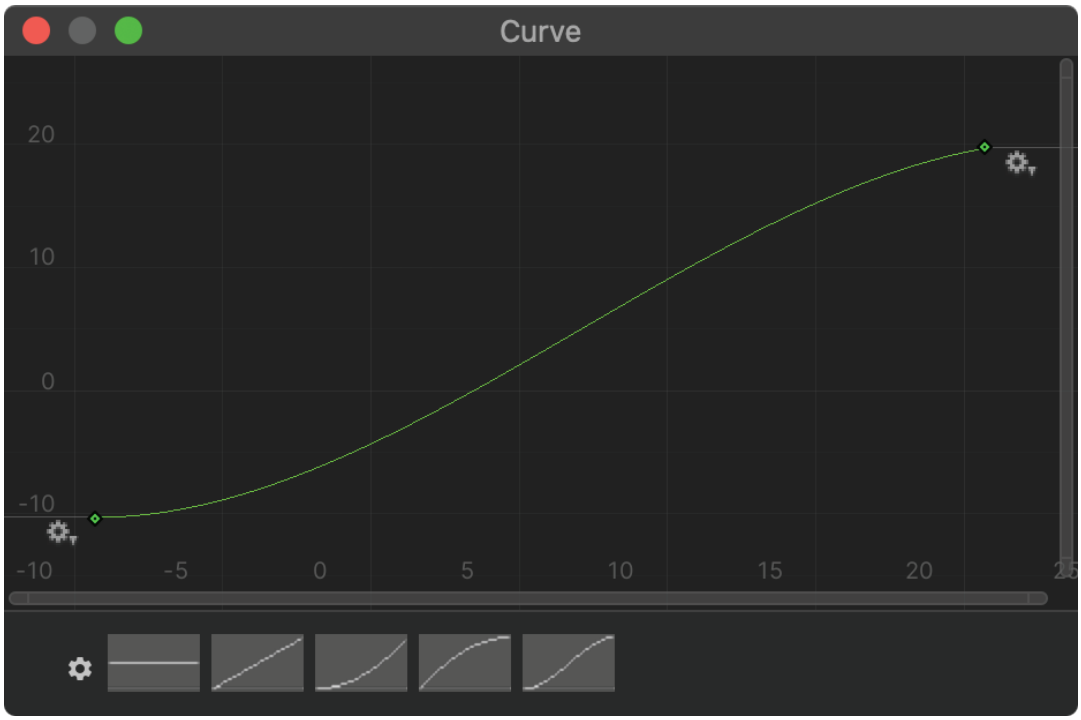
Curve Mapping

Curve Mapping モードは、Automatic モードのもう 1 つのバリエーションです。



Curve Mapping モード

この設定では、カーブの X 軸は現在の露出を表し、Y 軸はターゲットとする露出を表します。露出カーブの再マップによって、非常に高精度な結果を生成できます。

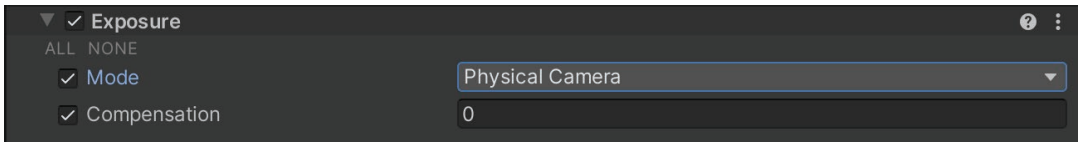


カーブを変えて露出を調整します。

Physical Camera

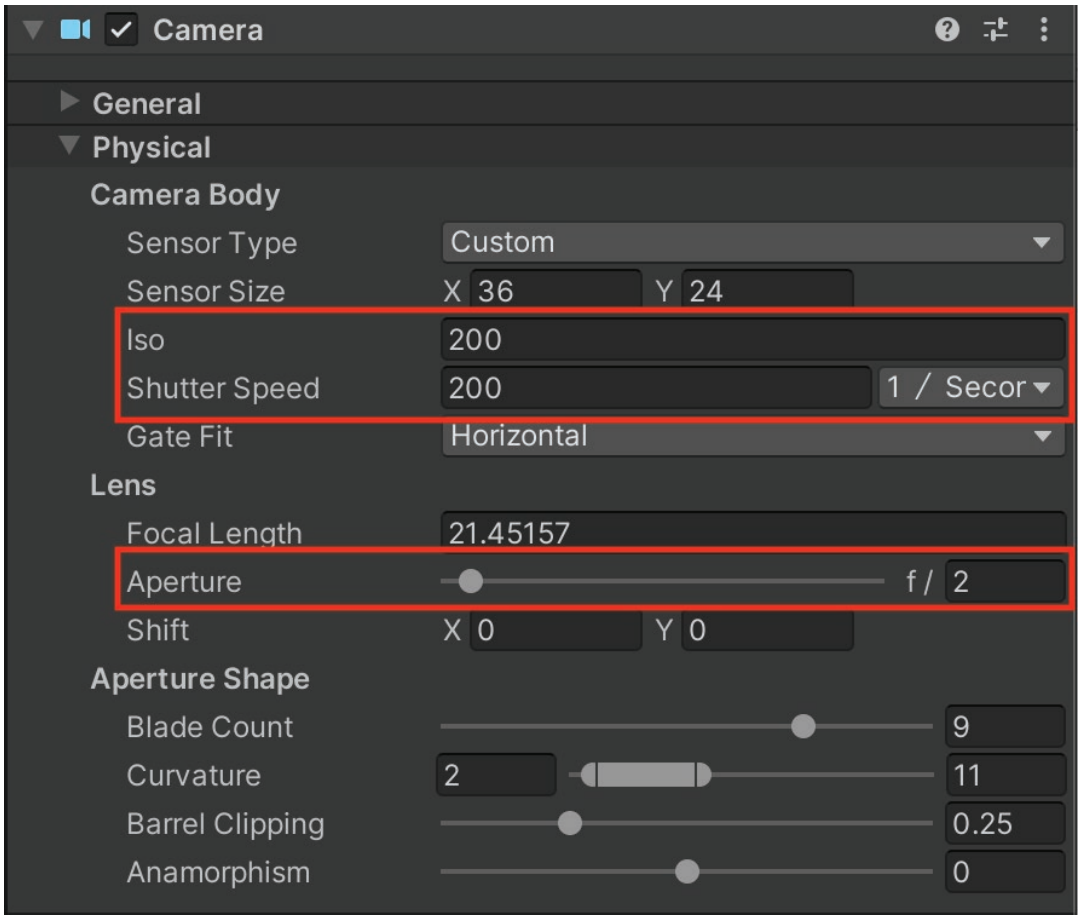
写真に詳しい方は、Physical Camera (物理カメラ) モードを使用すると、カメラのパラメーターを設定しやすいかもしれません。

Exposure オーバーライドの **Mode** を **Physical Camera** に切り替えてから、Main Camera を確認します。



Physical Camera モード

Physical Camera を有効にします。インスペクターに次のプロパティーが表示されます。



カメラの Physical Camera プロパティー

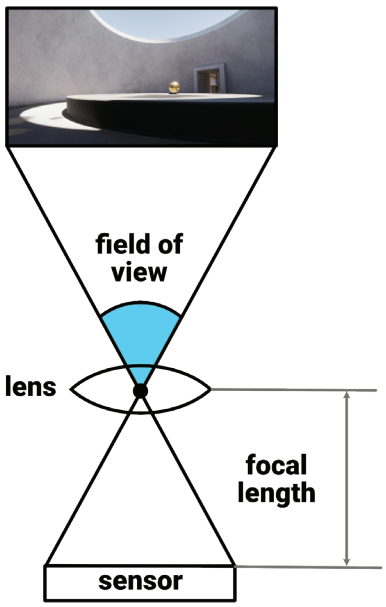
露出関連で重要なのは、**ISO** (感度)、**Aperture** (F 値)、**Shutter Speed** です。参照用の写真に合わせる場合は、画像の Exif データから適切な設定をコピーします。または、こちらの表を参考にすると、F 値とシャッタースピードを基に露出値を推測できます。

Physical Camera のその他のパラメーター

露出とは無関係ですが、Physical Camera のその他のプロパティも、現実世界のカメラの属性に合わせる助けになります。

例えば、Unity (および他の多くの 3D アプリケーション) では通常、**有効視野 (FOV)** を使用して、カメラが一度に撮影できるゲーム世界内の範囲を決定します。

しかし、実際のカメラでは、有効視野 はセンサーの大きさとレンズの焦点距離によって決まります。直接有効視野を設定するのではなく、Physical Camera の設定を使用すると、実際のカメラのデータの **Sensor Type**、**Sensor Size**、**Focal Length** を入力できます。データを入力すると、それに対応する有効視野の値が自動的に算出されます。



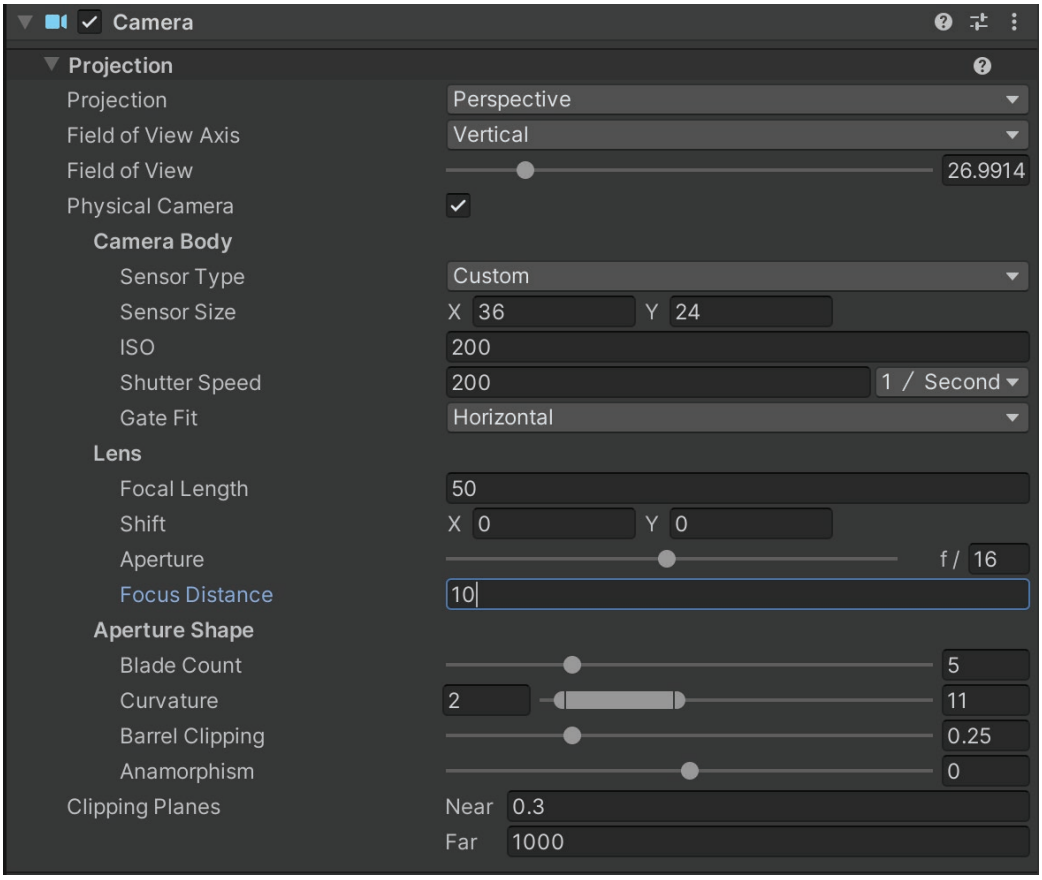
焦点距離、センサーサイズ、有効視野の関係

実際の写真に合わせて露出を設定するには、その画像ファイルに含まれるカメラのメタデータを利用します。Windows と macOS の両方で、デジタル画像から Exif データを読み取ることができます。そのうえで、対象のフィールドをバーチャルカメラにコピーできます。

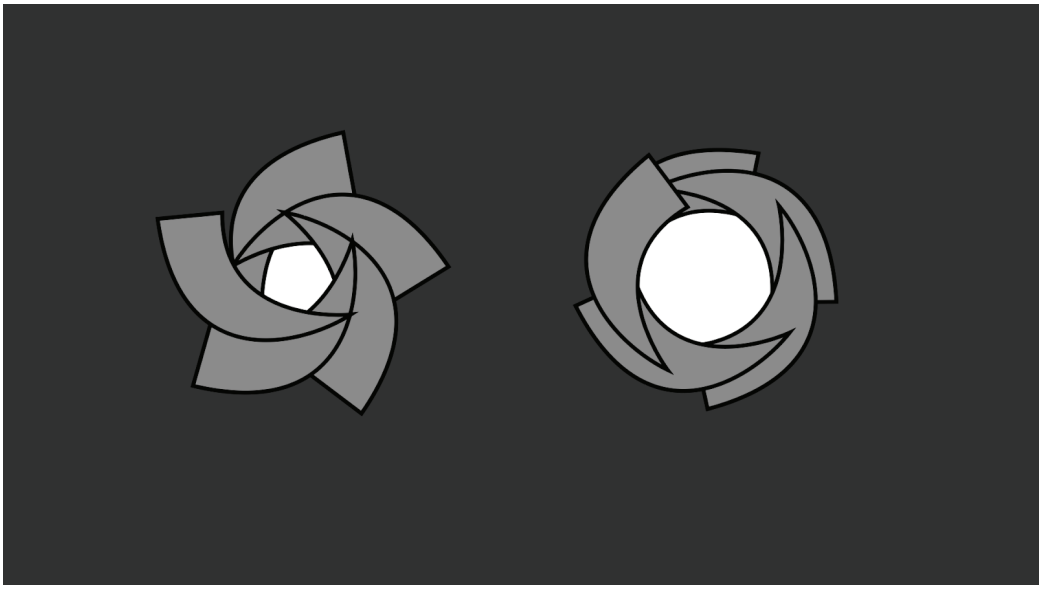
注意:場合によっては、メタデータからカメラのメーカーとモデル名を確認し、そのメーカーのウェブサイトで正確なセンサーのサイズを調べる必要があります。こちらの記事 には、一般的な画像センサーフォーマットのおおよその値が記載されています。

以下のパラメーターのいくつかは、Depth of Field (被写界深度) Volume に影響します。

Camera の Inspector から **Focus Distance** (焦点距離) を制御できます。Depth of Field Volume コンポーネントで、**Focus Mode** と **Focus Distance モード** を **Physical Camera** に設定します。



Blade Count、**Curvature**、**Barrel Clipping** を使用して、カメラの絞りの形状を変更できます。これは、Depth of Field Volume コンポーネントによって生じるボケの見え方に影響します。



物理カメラのパラメーターを使用して絞りの形状を設定します。5 枚羽根の絞りの場合、五角形 (左) または円形 (右) のボケを表現できます。

Lights

HDRP には、シーン内のイルミネーションの制御に役立つさまざまなタイプのライトや形状が用意されています。

ライトのタイプ

Unity の他のレンダーパイプラインと同様に、次のタイプのライトを利用できます。

- **ディレクショナル:** 無限遠に位置する光源からの光のような特性を持ちます。強度が落ちない完全な平行光線です。ディレクショナルライトは、多くの場合、日光として使用されます。一般的に、屋外のシーンでは、これがキーとなるライトです。
- **スポット:** 現実世界のスポットライトに似ており、円錐、ピラミッド、ボックスといった形状があります。スポットは、前方 Z 軸に沿って、円錐/ピラミッド形状のエッジに向かって落ちていきます。
- **ポイント:** 空間内のある 1 点からすべての方向に光を放つ全方向性ライトです。ランプやキャンドルなどの放射光源に適しています。
- **エリア:** 特定の形状 (矩形、チューブ、ディスク) の表面から光を投影します。エリアライトは、窓や蛍光灯のような、中央部の強度が均一である面的な光源においての使用に適しています。

Range を使用して、スポットライト、ポイントライト、エリアライトをどのように減衰させるかを調整できます。多くの HDRP ライトは、現実世界の光源と同じように、逆 2 乗の法則 に従って減衰します。

形状

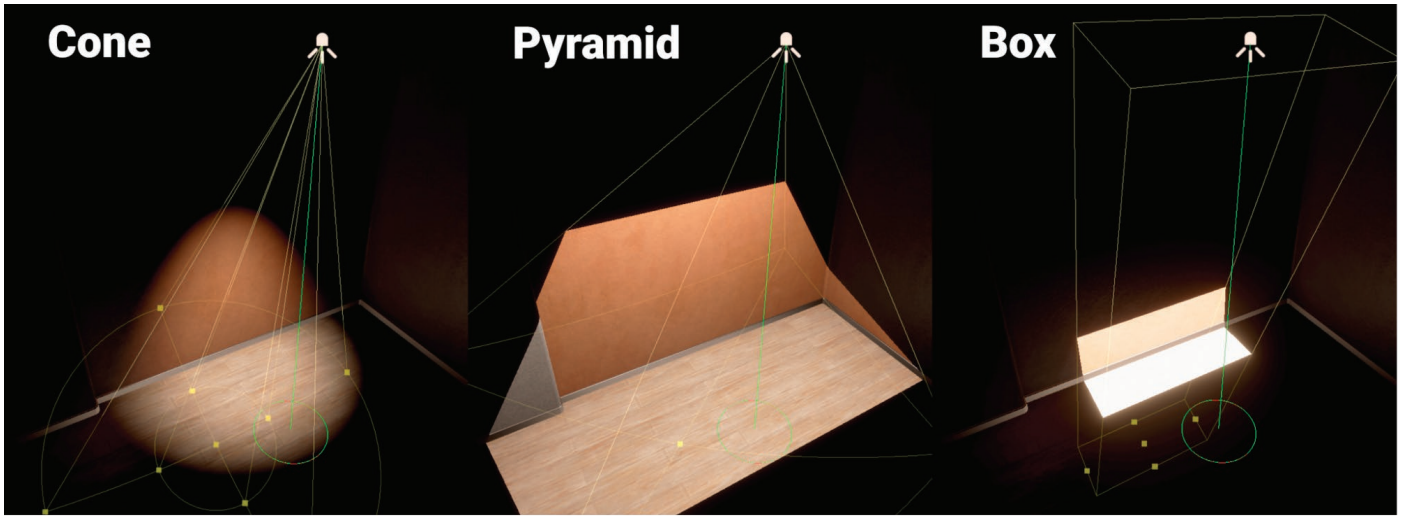
スポットライトとエリアライトには、各ライトの減衰方法を制御するために独自の形状が追加されています。

HDRP のスポットライトでは、3 つの形状を使用できます。

— **Cone** (円錐) は、1 点から円錐形に広がる光を投影します。角度で指定する **Outer Angle** とパーセンテージで指定する **Inner Angle** を使用して、円錐の形と角度による減衰を調整します。

— **Pyramid** (ピラミッド) は、1 点から四角錐型に広がる光を投影します。**Spot Angle** と **Aspect Ratio** (アスペクト比) を使用して四角錐の形状を調整します。

— **Box** (ボックス) は、矩形ボリューム全体へ均一に光を投影します。X サイズと Y サイズによって基部となる長方形を定めて、Range によって Y 次元を調整します。この光は、**Range Attenuation** がオンになっていなければ減衰しないので、ボックスの範囲内で日光のシミュレーションも行うことができます。



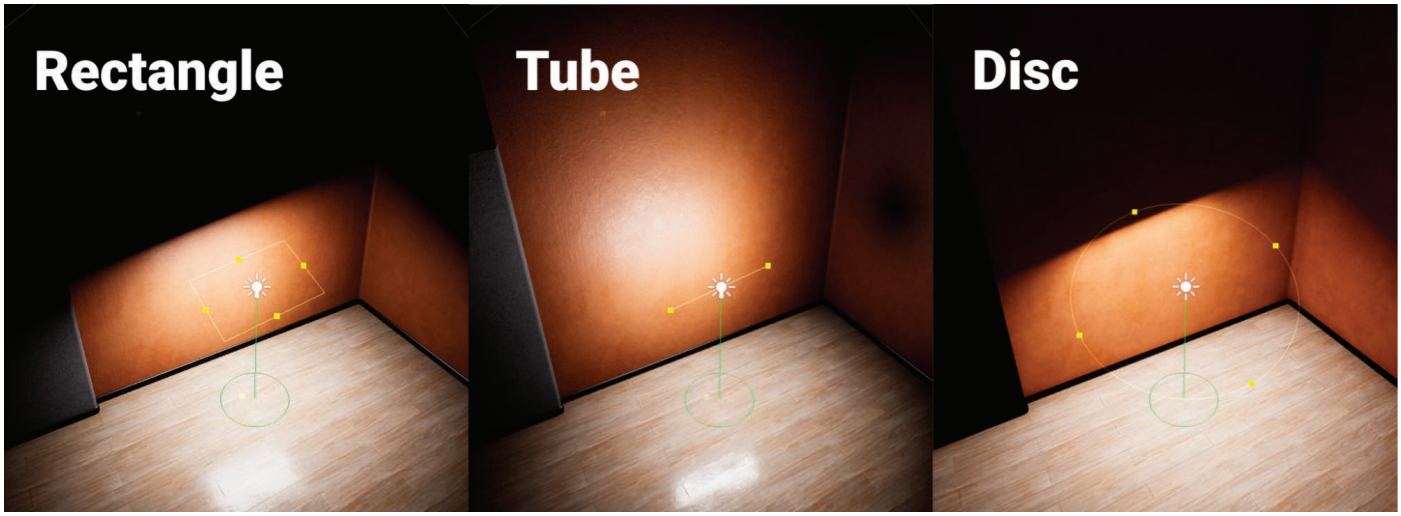
HDRP のスポットライトの形状

HDRP のエリアライトでは、3 つの形状を使用できます。

— **Rectangle** (矩形) は、矩形の形状からローカルの正の Z 方向に、Range で定義された範囲まで光を投影します。

— **Tube** (チューブ) は、1 本の線から全方向に、Range で定義された範囲まで光を投影します。このライトは Realtime Mode でのみ使用できます。

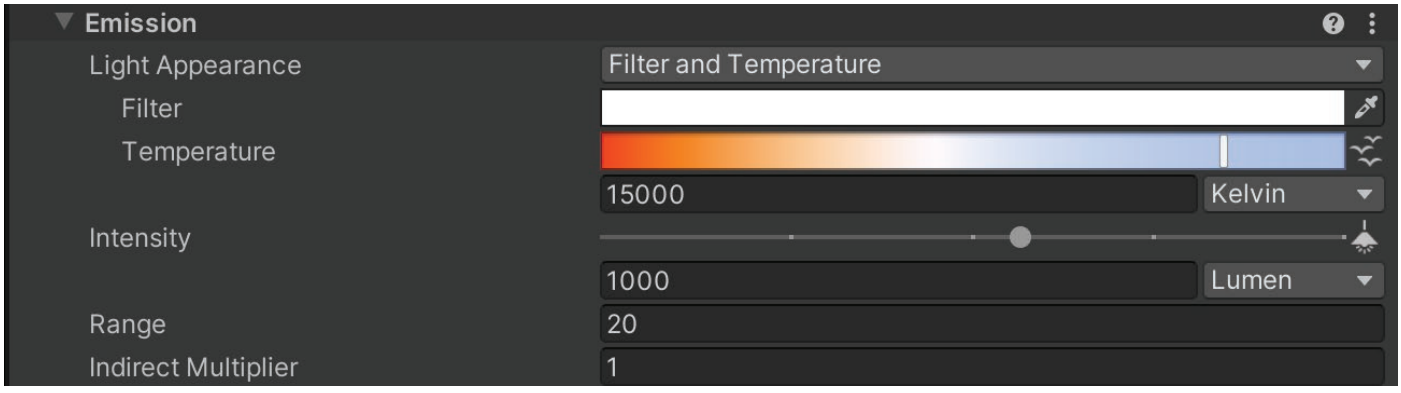
— **Disc** (ディスク) は、円盤形状からローカルの正の Z 方向に、Range で定義された範囲まで光を投影します。このライトは Baked Mode でのみ使用できます。



HDRP のエリアライトの形状

色と温度

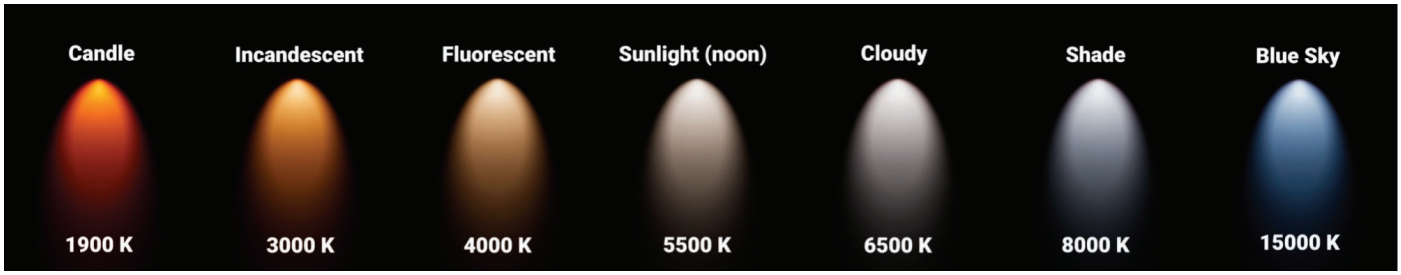
HDRP のすべてのタイプのライトに、ライトの外観を定義する **Emission** プロパティがあります。



Emission の Light Appearance プロパティを変更します。

Light Appearance を **Color** に切り替えると、RGB カラー を指定できます。または、これを **Filter and Temperature** に変更し、より物理的に正確なデータを入力できます。

色温度では、ケルビン温度 に基づいて色を 設定します。詳しくは、「ライティングと露出のチートシート」を参照してください。



ケルビン温度で表された色温度

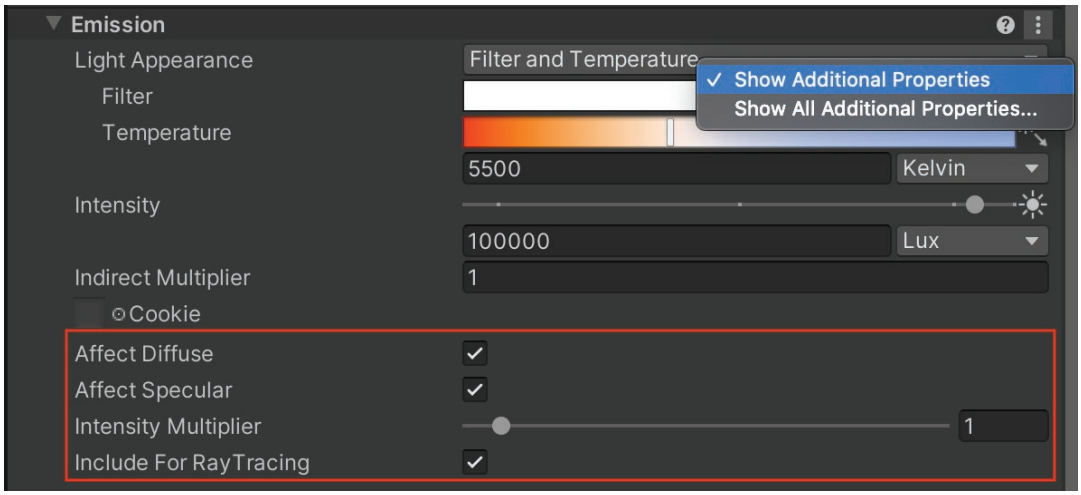
別の色を追加して**フィルター**のように作用させることで、別の色相を使ってライトに色を付けることもできます。これは、現実の写真撮影でカラージェルフィルターを使用する方法と似ています。

追加のプロパティ

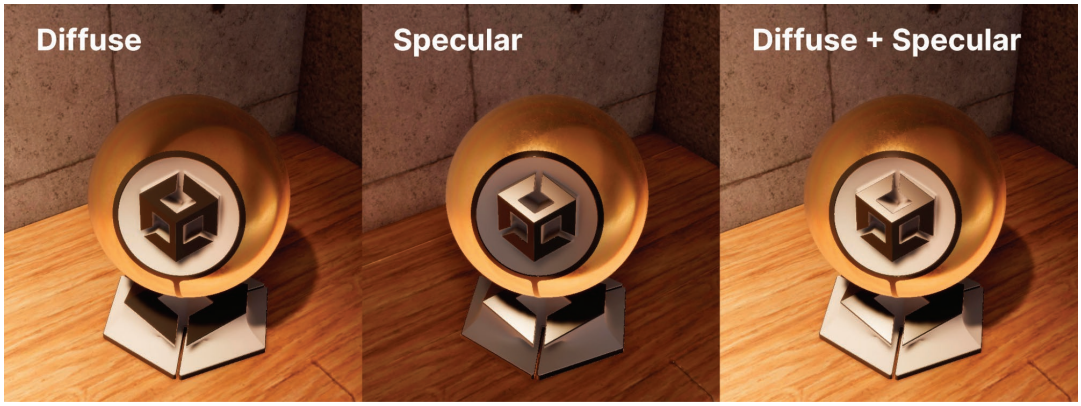
Inspector のプロパティの右上にある **その他のアイテムメニュー (:)** の下にも、高度なコントロールがいくつか用意されています。追加のオプションを見るには **Show Additional Properties** を選択します。

Affect Diffuse や **Affect Specular** などのトグルがあります。たとえば、カットシーンや映画のようなライティングでは、明るいハイライトを制御するライトと、よりソフトなディフューズライトを生成するライトを分離することができます。

Intensity Multiplier を使用して、ライトの元の強度の値を変更せずに全体的なライトの強度を調節することもできます。複数のライトを一括で明るくしたり暗くしたりするのに便利です。



各ライトに追加のプロパティがあります。

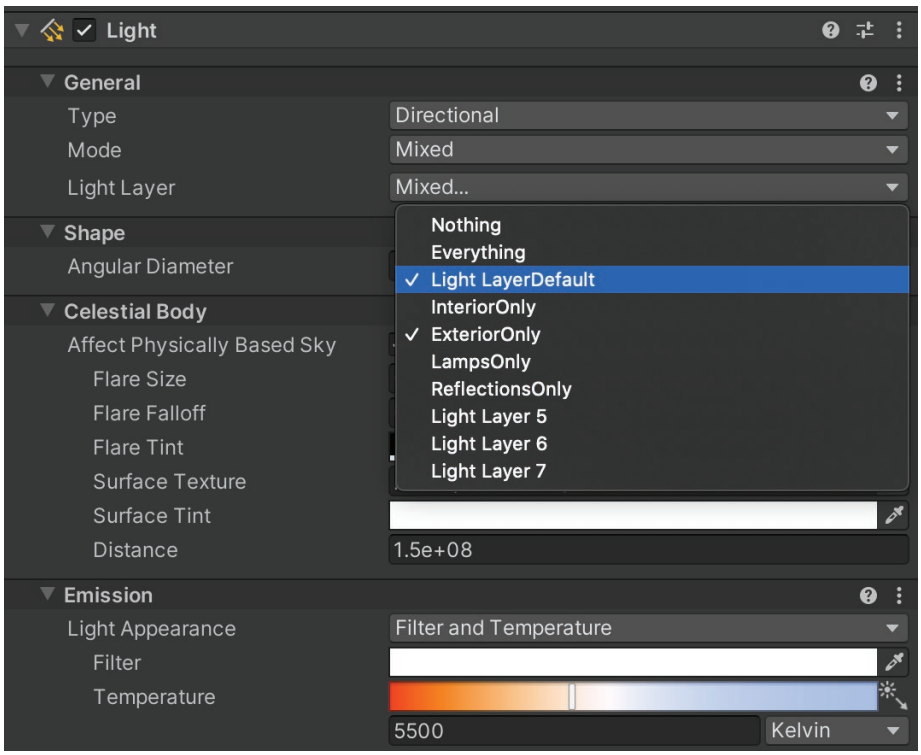


ディフューズライティングとスペキュラーライティングの設定を細かく調整できます。

ライトレイヤー

HDRP では、**Light Layers** (ライトレイヤー) を使用して、ライトの影響範囲をシーン内の特定のメッシュに限定できます。これは、ライトコンポーネントと MeshRenderer に関連付けることができる LayerMask になります。

Light のプロパティで、**その他のオプションボタン** (⋮) をクリックします。**General** (全般的な設定) に **Light Layer** ドロップダウンが表示されます。Light に関連付けるレイヤーマスクを選択します。

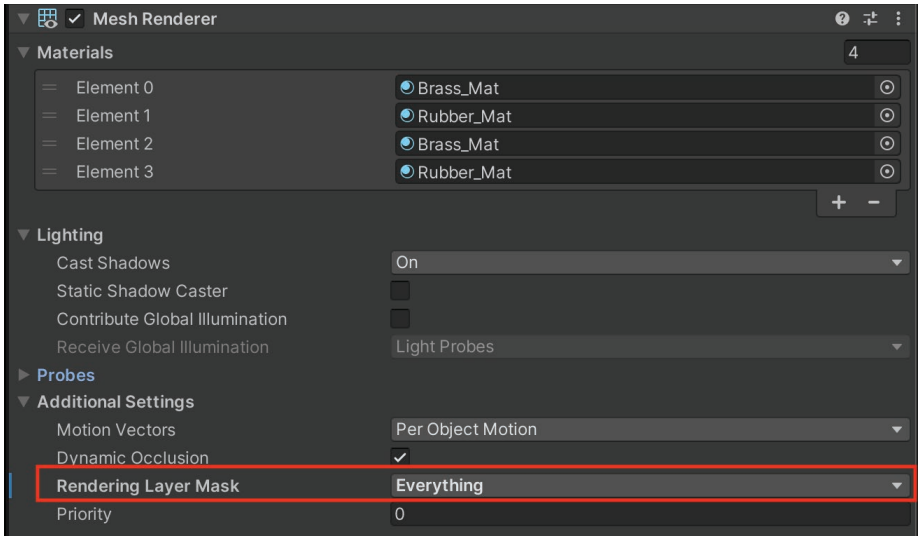


ライトレイヤーを選択します。

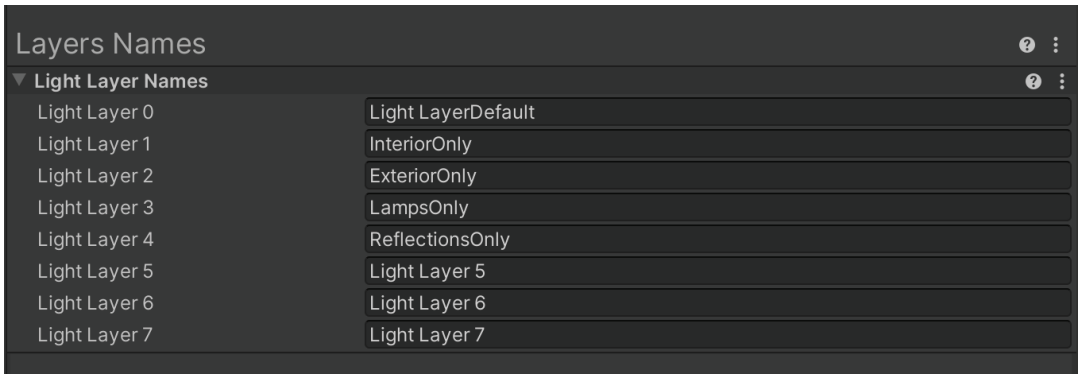
次に、**レンダリングレイヤーマスク**を使用して MeshRenderer を設定します。対応する LayerMask 上のライトのみがメッシュに影響します。これは、光漏れを解消し、意図した対象のみに確実にライトを当てるために欠かせない機能です。また、カットシーンのライティングを設定するワークフローにこの機能を組み込むことで、映画的な演出でキャラクターだけに専用のライトが当たるようにすることができます。

たとえば、建物内部のライトが意図せず壁を貫通し、屋外に出るのを防ぎたい場合は、屋内用と屋外用のライトレイヤーを設定します。この機能を使えば、ライトの構成を細かく制御できます。

Rendering Layer Mask を設定し、特定のライトのみがメッシュに適用されるようにします。



ライトレイヤーを設定するには、**HDRP Default Settings** にアクセスします。**Layers Names** セクションでは、**Light Layer 0** から **Light Layer 7** に文字列で名前を設定できます。



HDRP Default Settings の Layers Names

Light のプロパティの完全なリストなどの詳細については、Light コンポーネントに関するドキュメント を参照してください。

Light Anchor

Unity 6 には、カメラと被写体の間の角度と距離を制御することで、ライトを素早く設定できる **Light Anchor** システムが備わっています。また、9 つのプリセットで一般的なライトの角度を選択できます。

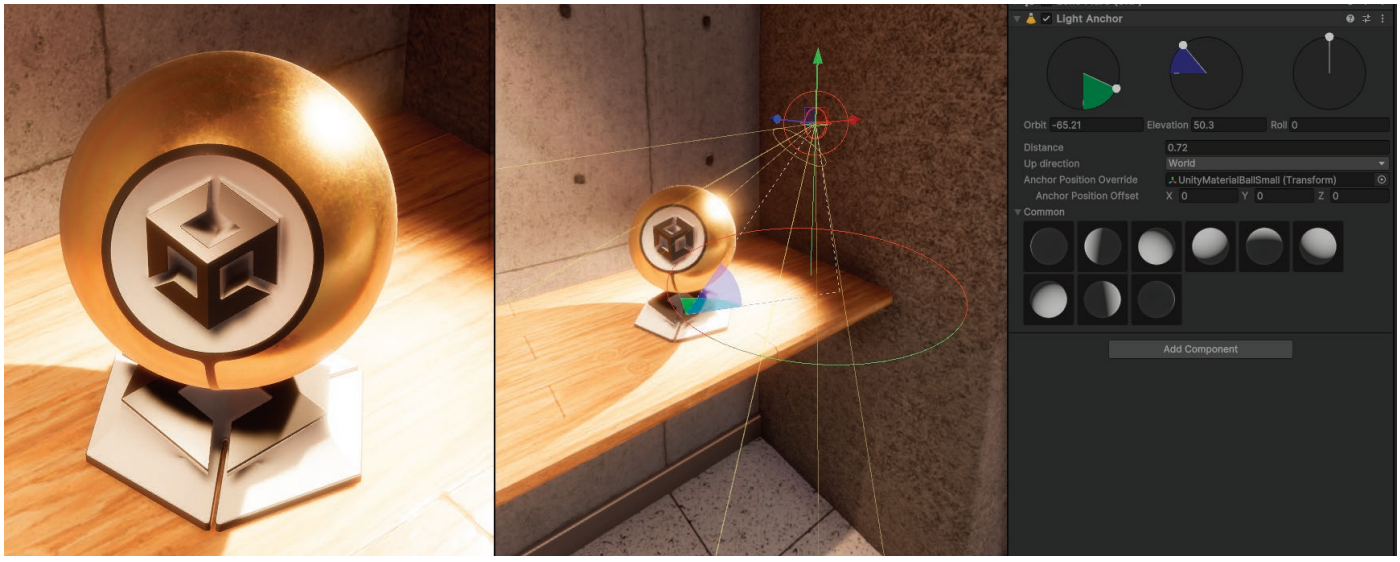
キャラクターや小道具の周りで複数のライトを使用して、シーン、製品、またはシネマティックのショットを照らす必要がある場合、Light Anchor コンポーネントを使用すると、アンカーターゲットを中心としたスクリーンスペース内で高速なライト操作を行うことができます。

まず、Light Anchor が動作するように、カメラが MainCamera としてタグ付けされていることを確認してください。次に、制御したい **スポットライト** に **Light Anchor** コンポーネントを追加します。ライトを被写体に合わせます。その位置が、スポットライトのアンカーポイントになります。アンカーポイントとスポットライトの**距離**を増やします。

シーンビューでライトのTransformを手動で調整するのではなく、ゲームビュー内でライトの**Orbit**、**Elevation**、**Roll** を調整することで、アンカーポイント周りのライトの位置を調整できるようになりました。

Light Anchor コンポーネント

詳しくは、Light Anchor に関するこちらのプレゼンテーションをご覧ください(日本語字幕あり)。



Transform を直接変更する代わりに、Light Anchor コンポーネントを使用してライトの照準を合わせることができます。

物理ベースのライト単位と強度

HDRPでは、光の強度の測定に物理ベースのライト単位(PLU)を使用します。これは、カンデラ、ルーメン、ルクス、ニトなど、光に関する現実世界でのSI測定単位と一致しています。PLUでは、正確性を確保するため、Unityの1単位は1メートルに等しいことが前提になっていることにご注意ください。

単位

物理ベースのライト単位には、*光束*と*照度*の両方の単位が含まれる場合があります。光束とは光源から *放出* される光の総量であり、照度とはオブジェクトに *当たる* 光の総量 (単位面積あたりの光束で表されるのが一般的) です。

商用のライティングや写真の分野では、用途によって使用される単位が異なる場合があります。そのため、Unityでは互換性を確保するために、次のように複数の物理ベースのライト単位に対応しています。

カンデラ:1単位が、1本のろうそくの光束に相当します。日本では以前は *燭 (燭光)* という単位でした。

- **ルーメン:**光束のSI単位で、1ステラジアン (立体角) に対する1カンデラの光束と定義されます。ルーメンは、商用電球の仕様によく見られます。Unityのスポット、ポイント、エリアライトで使用します。
- **ルクス:**1平方メートルのエリアに1ルーメンの光を放出する光源が、照度1ルクスに相当します。現実世界で光を測定する際は、ルクスを読み取るのが一般的です。通常、Unityのディレクショナルライトではこの単位を使用します。
- **ニト:**1平方メートルあたり1カンデラに相当する輝度の単位です。ディスプレイデバイスやLEDパネル (テレビやモニターなど) では、輝度をニト単位で測定するのが一般的です。

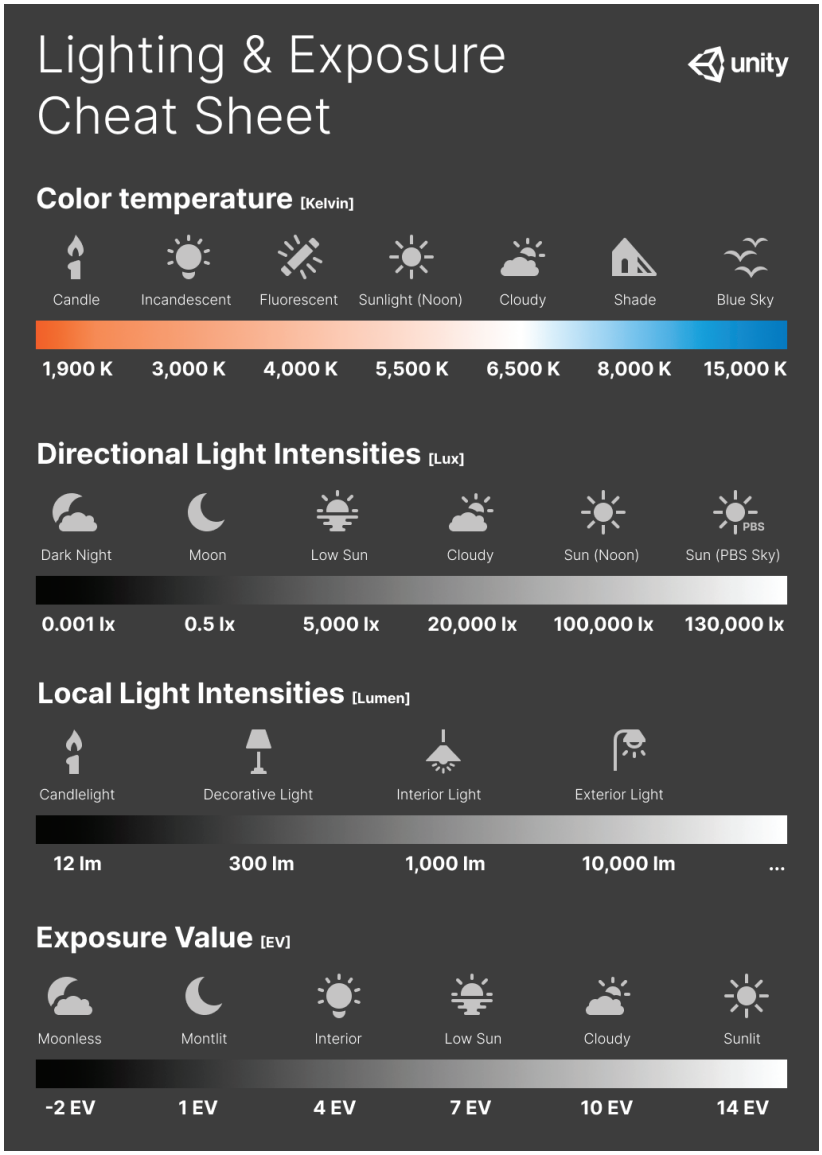
— **EV100:**EV100、つまり ISO100 フィルムにおける露出値に相当する強度を使用します (前述の露出値の計算式を参照)。露出を上げると、対数的な計算により、ライティングが倍増します。

— 現実世界の光源を再現するには、技術仕様に記載されている単位に切り替え、適切な光束または輝度を設定します。HDRP　によって物理ベースのライト単位に合わせることで、勘に頼らずに強度を設定できます。

— アイコンをクリックし、**Exterior**、**Interior**、**Decorative**、**Candle** のいずれかのプリセットを選択します。特定の値に合わせる必要がない場合は、このような設定から始めることをおすすめします。

ライティングと露出の一般的な値

ここに示すチートシートには、現実世界の一般的な 光 源の色温度と光の強度がまとめられています。また、さまざまなライティングシナリオに合った露出値も紹介されています。

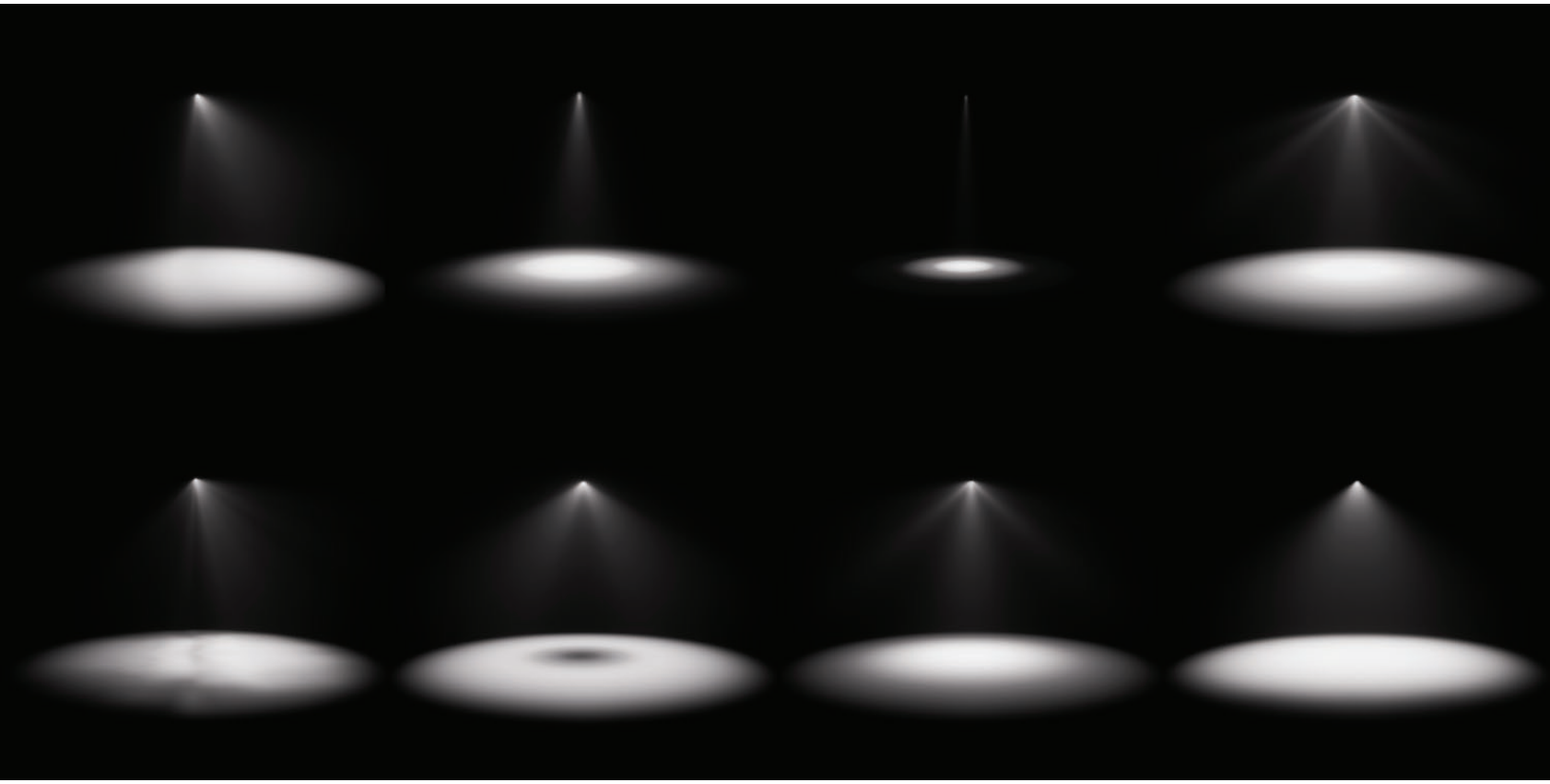


ライティングと露出のレベルに関するガイダンス

一般的なイルミネーションの値をまとめた詳細な表については、物理ベースのライト単位に関するドキュメントをご覧ください。

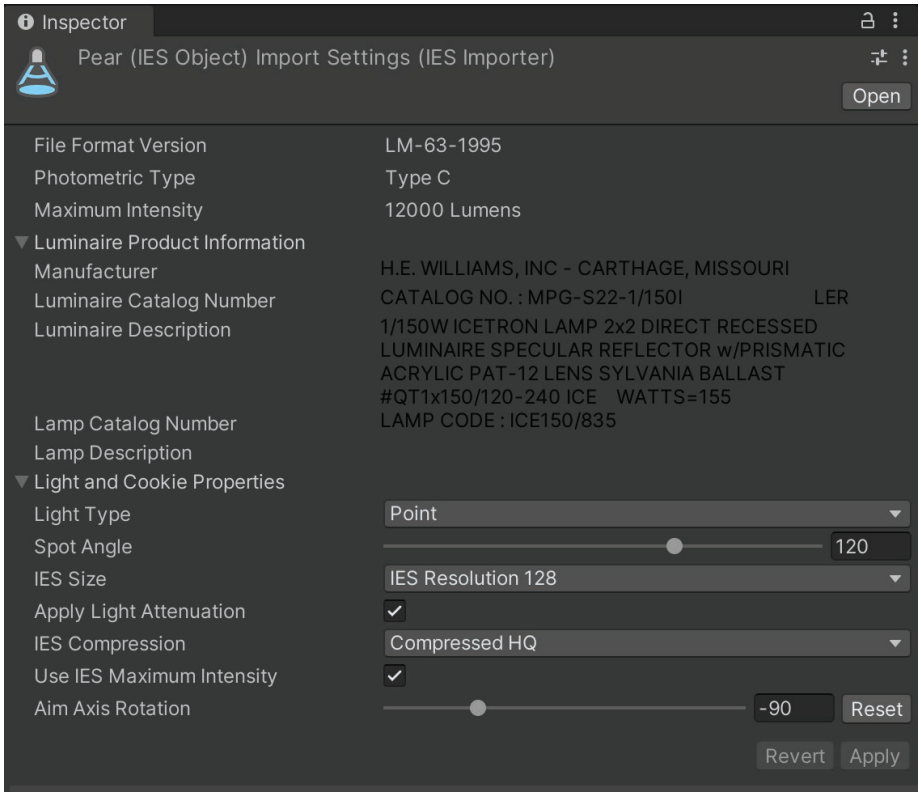
IES プロファイルとクッキー

IES プロファイル を使用すると、ポイント、スポット、エリアライトの減衰を、現実世界の光に近い形で再現できます。これは、特定の製造元の仕様をライトのパターンに適用する ライトクッキー のように機能します。IES プロファイルを使用すれば、ライトのリアリティをさらに高めることができます。



さまざまなライトに適用される IES プロファイル

IES プロファイルを **Assets > Import New Asset** からインポートします。インポーター によって、適切な強度で ライト のプレハブが自動的に作成されます。続いて、プレハブをシーンビューまたは Hierarchy にドラッグし、色温度を調整します。



IES プロファイルとインポート設定

IES プロファイルのソースの例は次のとおりです。

実際の製造元

- Philips
- Lithonia Lighting
- Efficient Lighting Systems
- Atlas
- Erco
- Lamp
- Osram

アーティストのソース

- Renderman

IES プロファイルのインポーターの詳細については、ドキュメントをご覧ください。

HDRP グローバル イルミネーション

グローバルイルミネーションを理解する

現実世界で光が表面に当たると、ただ止まるのではなく、反射したり、屈折したり、散乱したりします。ブラックホールでなければ、環境中に拡散する光子を捕らえることはできないでしょう。そのため、コンクリートや砂、石などのくすんだマテリアルでも光を反射します。

人間の目は、このような光の微かな挙動に反応します。光とさまざまなマテリアルの相互作用を正確に描写しているレンダリングは “リアル“ であると認識されます。

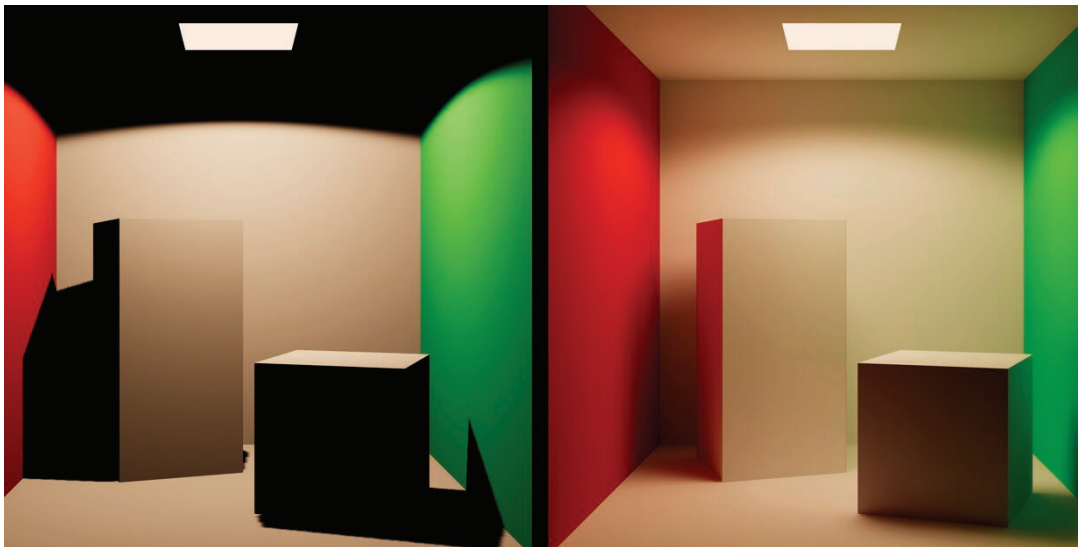
グローバルイルミネーション（GI）は、まさにそれを再現しようとしているのです。GI を使用しない場合、直接光の当たらない場所はデフォルトで暗くなります。GI は乱反射を再現しようとするもので、現実世界と同じように、色のついた光は 1 つの表面からまた別の表面へと移動します。このバウンスされた間接光は、ゲームの世界を現実に根付かせるのに役立ちます。

グローバルイルミネーションは、リアルな結果を生み出すことができます。出典：ArchVizPro Vol. 10.

HDRP グローバルイルミネーションの機能

GI は単一の技術ではなく、この現象を説明するのに役立つ一連のアルゴリズムです。Unity には、物理的環境における光の挙動を再現する様々なツールが揃っています。リアルなレンダリングには、以下のテクニックを組み合わせて使うことが多いでしょう。

- **ベイクしたグローバルイルミネーション（ベイクしたライトマップ）:**静的オブジェクトの間接光を事前に計算し、ランタイムで使用できるように結果を保存します。
- **リアルタイムグローバルイルミネーション:**間接光をリアルタイムで計算し、シーンの変化に対応します。
  - スクリーンスペースグローバルイルミネーション（SSGI）:リアルタイムの光伝搬用
  - レイトレーシングによるグローバルイルミネーション（RTGI）:物理的に正確な光の反射



グローバルイルミネーション（右）で、乱反射や間接光をシミュレートします。

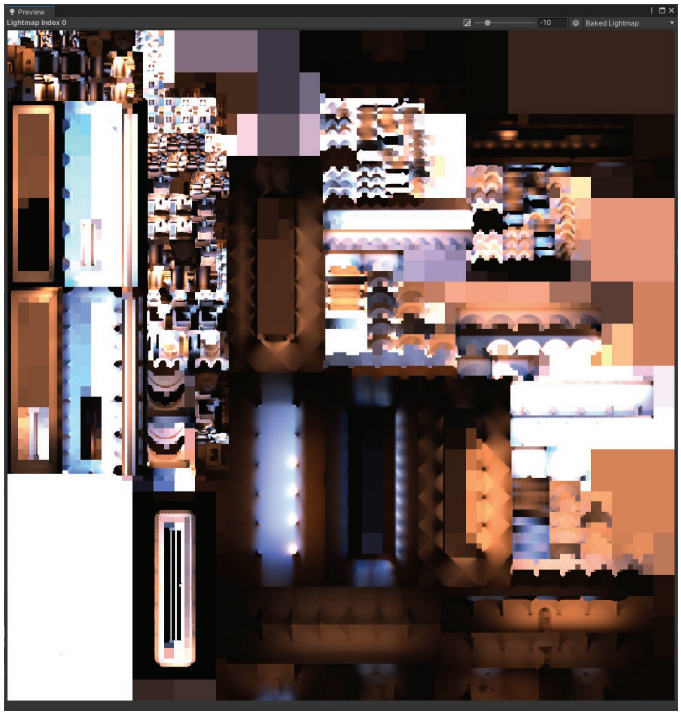
- **ライトプローブと Adaptive Probe Volume (APV):**事前に計算されたライティング情報を格納することで、リアルタイムで計算しなくても動的オブジェクトが間接光を受けることができます。
  - Light Probes:ベイクしたライティングを手動で配置した地点でキャプチャして補間します
  - APV:プローブ密度を動的に調整し、オクルージョンをサポートすることで、精度の高い大規模ライティングを実現します
- **環境ライティング:**ハイダイナミックレンジ画像（HDRI スカイボックスなど）を使用して、画像ベースの技術を駆使した環境ライティングのシミュレーションを行い、リアルなアンビエントライトを実現します。
- **リアルタイムレイトレーシング:**フォトリアリスティックなライトとマテリアルの相互作用を表現し、詳細な反射と影を捉えます。
- **パストレーシング:**カメラからの光線を再現し、HDRP がさまざまな効果（影、反射、屈折、間接光など）を統一されたプロセスで計算できるようにします。

HDRP のグローバルイルミネーションは、動的環境と静的環境の両方をサポートしています。これは、ゲームの世界を生き生きとしたものにし、反応性を向上させるのに役立ちます。

さらにヒントを見たい場合は、Unity で環境を照らす 4 つのテクニック の動画をご覧ください(自動保翻訳機能推奨)。

ベイクしたグローバルイルミネーション

ベイクしたグローバルイルミネーション（ベイクした GI）はシーン内の光の相互作用を事前に計算し、ライトマップと呼ばれるテクスチャとして結果を保存します。この手法は、光が表面でどのように反射および屈折するかを捉えるものですが、これをランタイム時に計算しようとすることはありません。

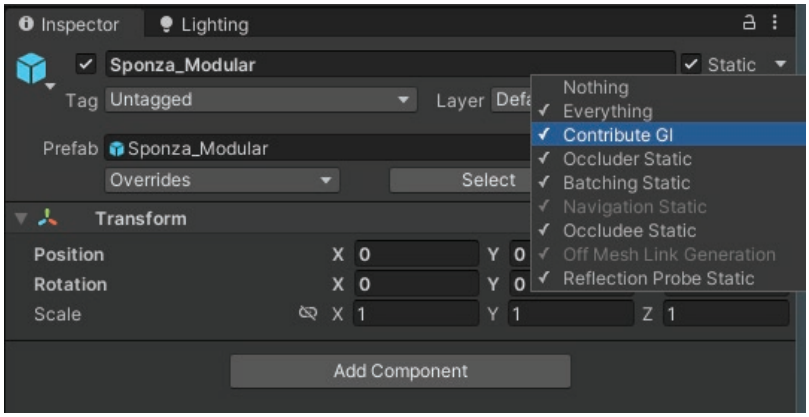


ベイクしたライトマップ

モバイルプラットフォーム向けの開発において、この戦略はゲーム環境にリアルなライティングを追加するためによく使われます。ライティングをベイクする際、集中的な計算はオフラインで一度だけ実行されます。

つまり、ランタイムにおいて、これらのライティング計算に関連する追加的なパフォーマンスコストは発生しません。

ベイクした GI には、いくつかの明確な利点が存在します。その 1 つがパフォーマンスにおけるもので、ライトの計算を事前に行うことで、ランタイム中の要求を大幅に軽減できます。このプロセスでは、ソフトシャドウや微かな乱反射など、複雑なニュアンスを捉えることができます。また、リアルタイムのライティング計算で発生しがちな、多くの視覚的なアーティファクトも排除します。



ライトマッピングは静的なオブジェクトに対してのみ機能します。

しかし、事前にベイクしたライティングは、静的な要素に対してのみグローバルイルミネーションをキャプチャします。シーン内の動的な要素や動く要素は、事前計算されたライティングデータの恩恵を十分に受けることができません。

注意すべきなのは、ライトマップがテクスチャアセット同様、追加のメモリとディスクスペースを必要とすることです。メモリリソースに制限があるプラットフォーム (モバイルなど) 向けに開発している場合、この要件に注意してください。

最後に、GI のベイクは、シーンの複雑さや使用ハードウェアによって、かなり時間がかかることがあります。静的ジオメトリの多い非常に大きな環境では、ライトマップの作成に数分、あるいは数時間かかることもあります。

ライトマッピングのワークフロー

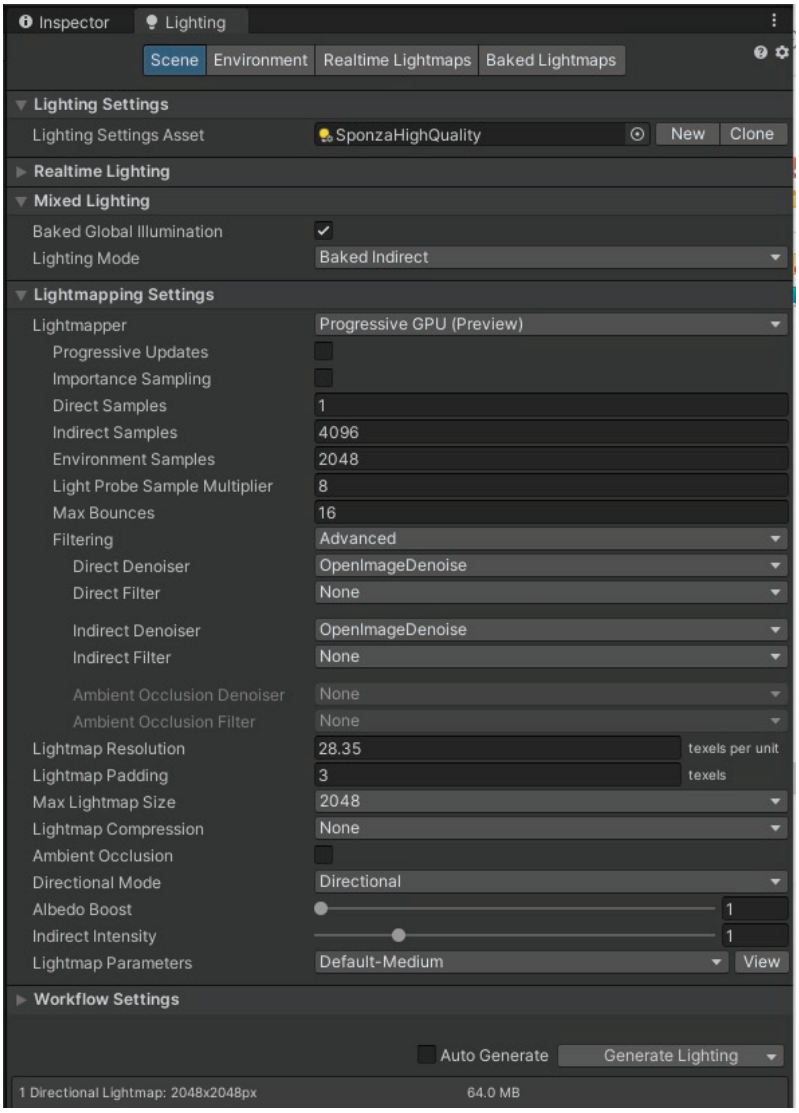
ベイクした GI を使ってシーンのライトマッピングを行う場合、以下の手順に従ってください。

1. **ライトモードを設定する:**ベイク処理の対象にするためには、各ライトを **Baked** または **Mixed** のいずれかに設定する必要があります。
2. **オブジェクトがライトマップ静的であるとマークする:**さらに、ベイク処理の対象とするオブジェクトは、Static ドロップダウン メニューから、**Contribute GI** または **Everything** のフラグを立てる必要があります。
3. **新しいライティング設定アセットを作成する:**Lighting ウィンドウを使用して、LightingSettings クラスの保存されたインスタンスを表す新しい Lighting Settings Asset (ライティング設定アセット) を生成します。ここには、ベイクされた GI (および以下で説明している Enlighten Realtime GI) 用に事前計算されたライティングデータが保存されます。

4. **ライティングモードを選択する:**Lighting Settings Asset で、**Bake Indirect** (間接ベイク)、**Shadowmask** (シャドウマスク) または **Subtractive** (減法) を選択します。ライティングモードの決め方については、こちらのガイドをご覧ください。
5. **Lightmap Settings で、基本パラメーターを調整する:Lightmap Resolution** (ライトマップ解像度) と **Sample Count** (サンプル数) を使用して、ベイクの精度と品質を制御します。一般的に、解像度が高くサンプル数が多くなるほど高品質な結果が得られますが、ベイク時間が長くなり、リソースの消費が激しくなる可能性があります。
6. **Generate Lighting** をクリックし、ライトマッピングを開始します。

ベイクしたライトマップの作成時には、計算と同時にプログレッシブにライトマップの練度を上げる Progressive Lightmapper が使われます。

ベイキング処理をリアルタイムでプレビューできます。必要な場合はベイクを中断し、設定を調整してリベイクしましょう。この反復型のプロセスにより、よりインタラクティブなライティングワークフローが実現されます。結果は、Lighting ウィンドウの **Baked Lightmaps** (ベイクしたライトマップ) タブに表示されます。



Progressive Lightmapper を有効化します。

ライトマップの最適化

ベイクした GI の最適化は、ビジュアルの品質と計算効率およびメモリ管理の戦略的バランスをとることを意味します。

ライトマッピングの一般的なヒントは以下の通りです。

- **ライトマップの解像度:**解像度が高いほど細部をキャプチャできますが、メモリ使用量も増加します。主要オブジェクトは優先的に解像度を上げ、背景に映り込む要素には解像度を下げましょう。
- **テクセルの無駄遣いに注意:**小石やワイヤーのような小さかったり細かったりするオブジェクトは、ライトマップのリソースを不当に消費する可能性があります。Static メニューまたは MeshRenderer で **Contribute Global Illumination** を無効にすると、シーンのライティングに大きな影響を与えるオブジェクト (色が明るかったり、エミッシブマテリアルがあるものなど) 以外は、GI の計算から除外されます。

小さいオブジェクトや薄いオブジェクトに無駄なテクセルを使わないようにしましょう。

- **サンプリング:**サンプル数はライトベイクの品質に直接影響します。サンプル数が多いほどライティングのディテールが豊かになりますが、ベイク時間が長くなります。
- **ノイズ除去:**低照度などの特定の条件下では、ベイクによって視覚的なノイズが発生することがあります。**Auto** を選択して、HDRP がノイズ除去アルゴリズムを自動的に選択できるようにします。それ以外の場合は、**Advanced** を選択して Direct Denoiser (直接ノイズ除去) と Indirect Denoiser (間接ノイズ除去) をそれぞれ設定します。
- **ライトマップ圧縮:**圧縮技術はメモリ使用量を減らすことができるものの、画質がわずかに低下する可能性があります。
- **アンチエイリアス:**パフォーマンスを最適化するには、アンチエイリアスレベルを下げることを検討してください。例えば、**Project Settings (プロジェクト設定) > Quality (品質)** で 8x Multi Sampling から 2x Multi Sampling に切り替えることができます。

GPU ライトマッピング

プログレッシブライトマッパーには、CPU または GPU の 2 種類のバックエンドから選択できます。プログレッシブ GPU ライトマッパー は、GPU と専用ビデオ RAM (VRAM) を使ってベイクしたライトマップの生成を加速します。

GPU ライトマッパーが利用できる場合、CPU ライトマッパーと比較してライティングデータの生成速度が桁違いに速くなります。Unity 6 の GPU ライトマッパーが製品版になり、いくつかの重要な改善が加えられました。

- **LightBaker v1.0**:この新しいバックエンドは、ベイク時にシーンの状態のスナップショットを瞬時にキャプチャし、プロセスをより予測しやすく、より安定させます。ベイク中にシーンを変更してもベイク処理は再起動されなくなり、エディターの応答性が向上しています。
- **GPU Baking Profile**:GPU Baking Profile を使用して、パフォーマンスと GPU メモリ使用量のバランスを取ります。
- **より小さな GPU メモリ要件:**Unity 6 では、新しいベイキングバックエンドにより、GPU メモリの最小要件が 2 GB VRAM に緩和されました。

LightBaker バックエンドにより、ライトマップ、ライトプローブ、シャドウマスク、アンビエントオクルージョンテクスチャの処理が改善されます。この新しいアーキテクチャは内部的にモジュール化が進んでおり、メンテナンスとデバッグが容易になっています。これらの改善は、高品質な結果を保ちながらライティングのイテレーション時間を短縮するように設計されています。

Auto モードでは、以前のプログレッシブベイク機能が引き続き使用されます。

GPU ライトマッパーを使用する場合、ベイク速度を最適化するために以下をご検討ください。

- GPU によって加速された他のアプリケーション、特に VRAM を使用するアプリケーションを閉じる。
- Intel Open Image のような CPU ベースのノイズ除去機能に切り替え、VRAM の領域を解放する。
- 複数の GPU がある場合、1 つをレンダリング用に、もう 1 つをベイク用に割り当てる。
- ライトマップのサイズが 4096 以上の場合は特に、アンチエイリアスのサンプル数を減らす。

マニュアルの パフォーマンスガイドライン を参照してください。

Lightmap UVs

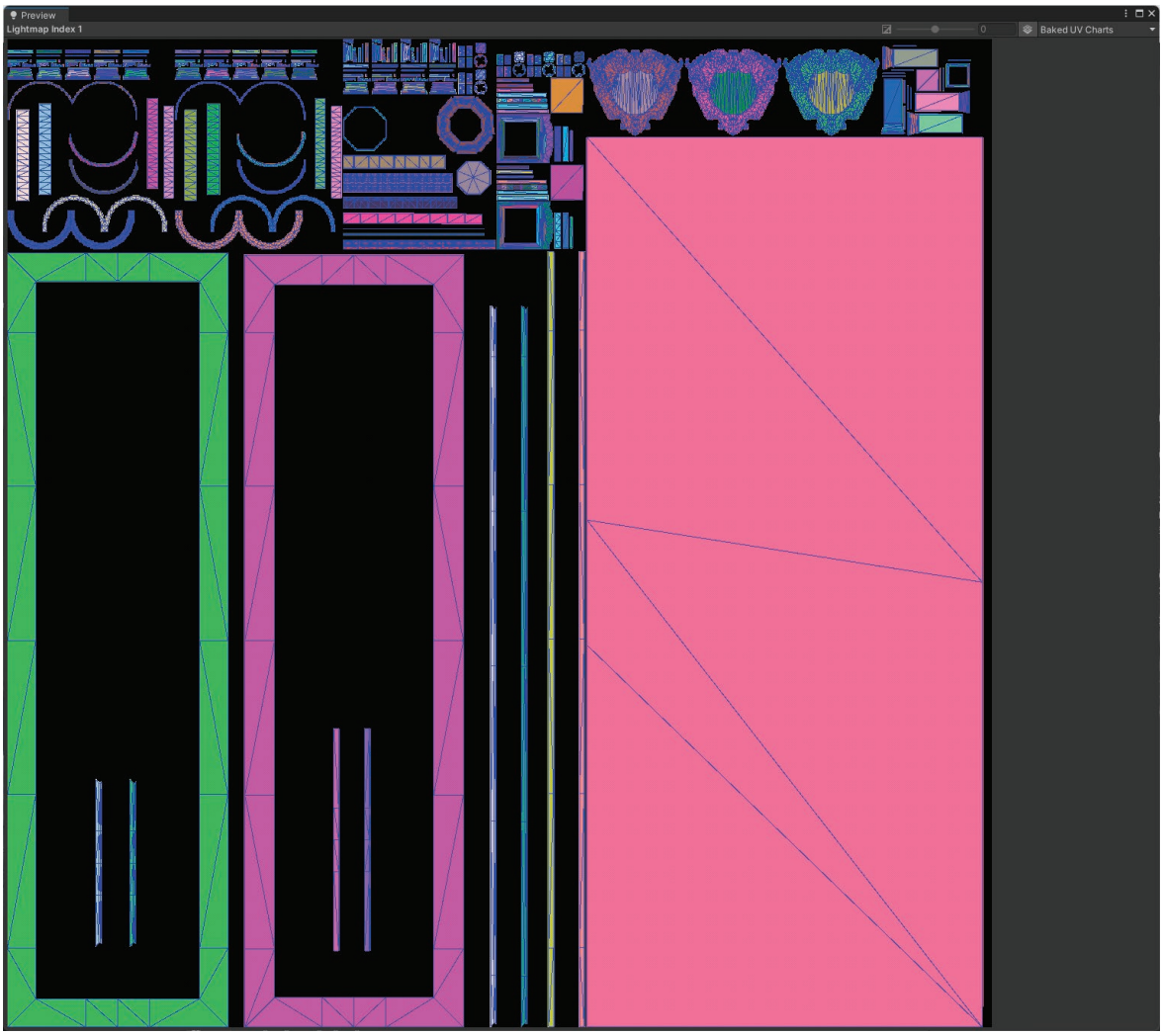
ライトマップはテクスチャであるため、Unity はそれらをシーンで正しく使用するために UV を必要とします。

インスタンスのグループ化とメッシュのスケーリングに違いがあるため、Unity は、ベイクしたグローバルイルミネーションとリアルタイムのグローバルイルミネーションにそれぞれ異なるライトマップ UV のセットを使用します。UV はメッシュ単位なので、同じメッシュのインスタンスはすべて同じ UV を使用します。

Unity はモデルのインポート時にこれらの UV を生成するか、モデル内の既存の UV セットを使用することができます。Unity は、ベイクしたライトマップの UV を Mesh.uv2 チャンネルに保存します。このチャンネルは TEXCOORD1 シェーダーのセマンティックにマップされ、一般に “UV1“ と呼ばれます。

Unity はリアルタイムライトマップの UV を再パックして、各チャートの境界に少量のパディングを持たせ、にじみ (特定の UV アイランドから隣りの UV アイランドへの光の漏れ) などのグラフィック関連の問題を減らします。

UV の計算は、インスタンスのスケールとライトマップの解像度に依存します。Unity は可能な限りこれを最適化し、同じメッシュ、スケール、ライトマップ解像度を持つ Mesh Rendere のコンポーネントが UV を共有できるようにします。



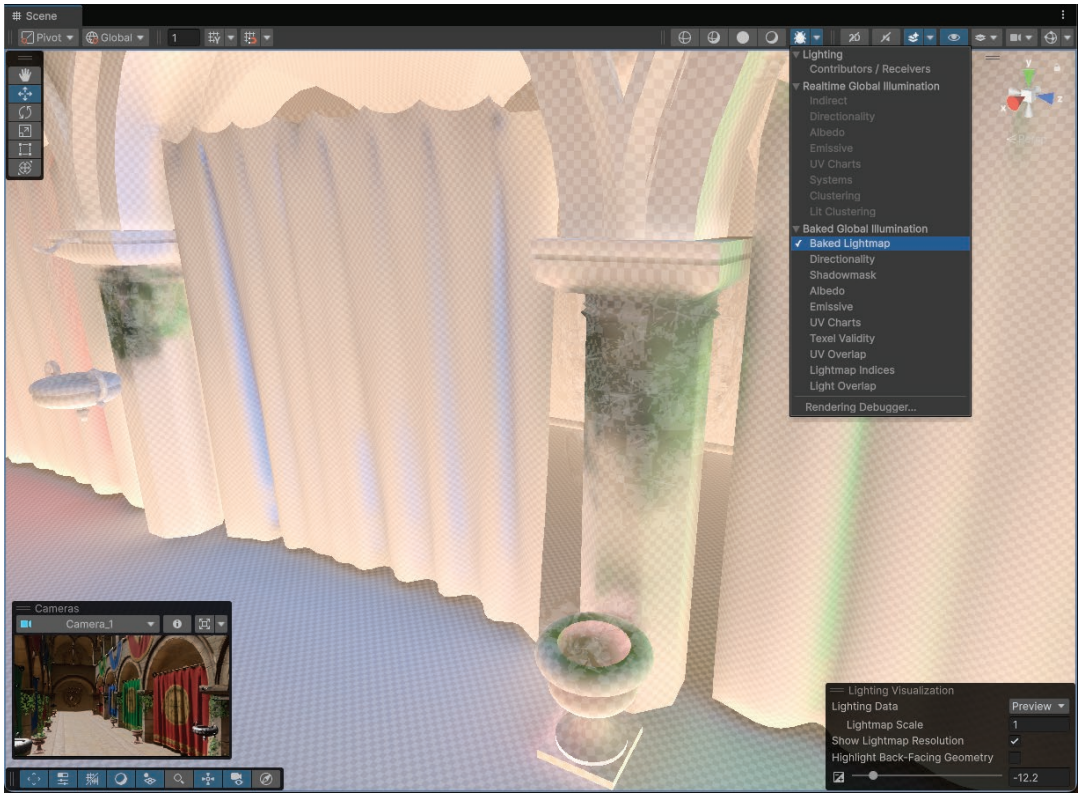
ライトマップを正しく表示するには UV が必要です。

ライトマップ UV の技術的な詳細については、ドキュメントのページ を参照してください。

ライティングのベイク処理における対話型プレビュー (Unity 6.1)

Unity 6.1 では、対話型の **GI Debug Preview Mode** がシーンビューに導入されています。この新機能により、ベイクしたライティングのリアルタイムプレビューが可能になり、シーンへの変更がグローバルイルミネーションにどのように影響するかを、完全な再ベイクを行わずに確認できます。

この機能はシーンビューの描画モードに直接統合されており、既存のベイク済みデータを変更することなく、対話的に更新情報を表示します。これを使用するには、ドロップダウンの **Baked Global Illumination** (ベイクしたグローバルイルミネーション) でデバッグドローモードを選択します。



GI Debug Preview Mode はシーンビューで利用できます。

この機能は、ベイクしたグローバルイルミネーション (GI) 向けの以前の **Auto Generate** (自動生成) システムに代わるものです。

動的グローバルイルミネーション

ベイクした GI により高品質な間接光が得られますが、設定に時間がかかり、計算コストが高くなる可能性があります。また、静的であるという大きな制約もあります。ベイクしたライトマップは、光の状態の変化、オブジェクトの動き、時間帯の遷移に対応できません。これを解決するため、Unity 6 HDRP では、リアルタイムまたはほぼリアルタイムでのライティング更新を可能にする動的グローバルイルミネーション (GI) 手法がいくつか用意されています。

スクリーンスペースグローバルイルミネーション (SSGI) では、画面の深度とカラーバッファを使用して、反射するディフューズライトを計算します。これにより、事前に計算されたライトマップを使用せずに、リアルタイムで間接光を得られます。

レイトレースグローバルイルミネーション (RTGI) は、ハードウェアアクセラレーションによるレイトレーシング (DXR) を使用して SSGI を拡張し、物理的により正確な光の反射を実現します。レイトレーシングを有効にすると、Inspector で利用可能なプロパティは RTGI 設定を反映して変化します。

ライトプローブ と **Adaptive Probe Volume (APV)** は、プローブを使用した間接光によりグローバルイルミネーションを近似します。APV はプローブを状況に合わせて配置し、リアルタイムまたはほぼリアルタイムで間接光を更新するので、ライティングの動的な変化を伴う大規模な環境に適しています。ライトプローブと APV の詳細については、次のセクション を参照してください。

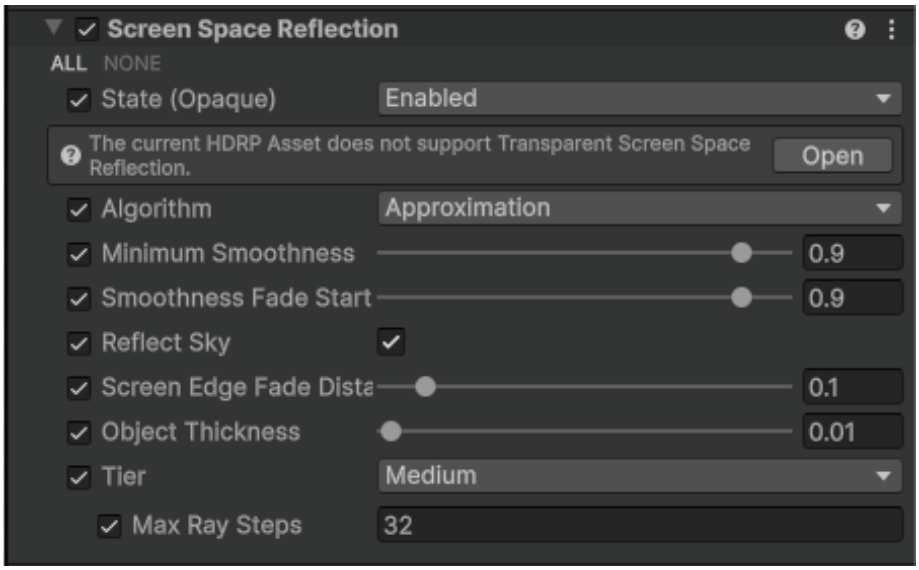
SSGI と RTGI の両方とも、ライトマップ と ライトプローブ のすべてのデータを置き換えます。カメラに影響を与えるボリュームで有効になっていると、ライトプローブとアンビエントプローブはゲームオブジェクトの間接光に寄与しなくなります。

プロジェクトのパフォーマンス要件と動的ライティングのニーズに最適な手法を選択してください。これらの手法は、プロジェクトのパフォーマンス要件や動的ライティングのニーズに応じて、ベイクしたライティングを補完または置き換えることができます。

スクリーンスペースグローバルイルミネーション (SSGI)

SSGI は、深度とカラーバッファを使用して光がサーフェスでどのように反射するかを推定することで、間接光を近似します。光の変化が激しく、ライトマップを事前に計算できない場合に便利です。

SSGI は画面上のデータに対してのみ機能し、画面外のオブジェクトはグローバルイルミネーションに寄与しないことに注意してください。



Screen Space Global Illumination (スクリーンスペースグローバルイルミネーション) を Volume オーバーライドとして追加します。

Volume システムで **Screen-Space Global Illumination** オーバーライドを追加します (**Lighting (ライティング) > Screen Space Global Illumination (スクリーンスペースグローバルイルミネーション)**)。**Frame Settings** と HDRP アセットの **Lighting** セクションでこの機能を有効にします。

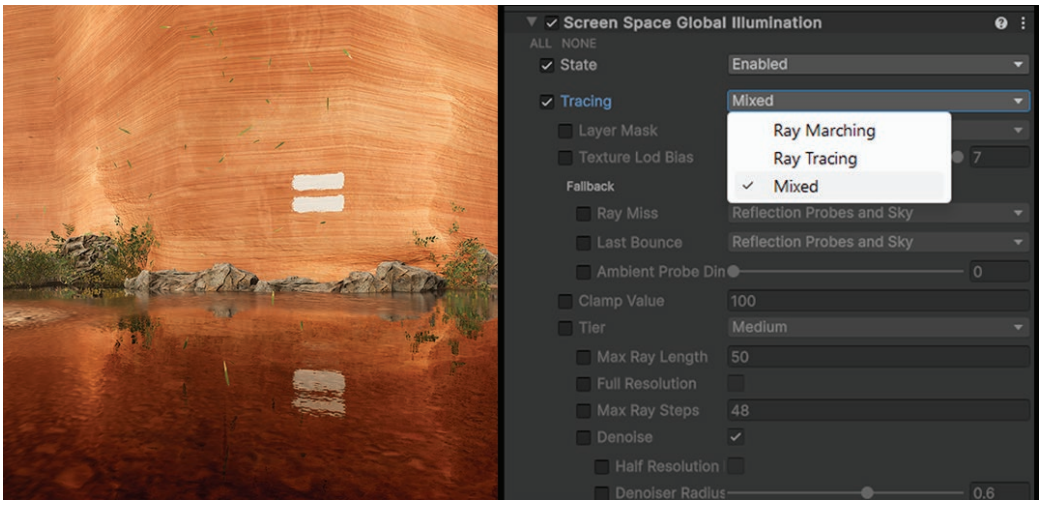
スクリーンスペースグローバルイルミネーションは、リアルな間接光を照射します。

トレースモード

SSGI とスクリーンスペースリフレクションはどちらも、Inspector に表示されるプロパティを決定する Tracing (トレース) モードを使用します。

- **Ray Marching:**このモードでは、スクリーンスペースにおける レイマーチング 手法を使用します。レイマーチングは、G バッファに表示されるスクリーンスペースオブジェクトのライティングを決定します。
- **Ray Tracing:**このモードはレイトレーシングを使用します。レイトレーシングは、スクリーンビューの外側、またはレイマーチングの制限を超えているオブジェクトのライティングを計算します。詳細については、「レイトレーシングとパストレーシング」(後述) を参照してください。
- **Mixed トレーシングモード:**これは、レイトレーシングとレイマーチングを組み合わせたものです。このモードでは、レイトレーシングだけでは不可能な、パーティクル、頂点アニメーション、デカールなどの追加エフェクトをキャプチャします。Mixed トレーシングモードでは、**Performance** モードと **Lit Shader Mode** を **Deferred** に設定する必要があります。これは、ディファードレンダリングでレンダリングされた G バッファ内のオブジェクトにのみ影響します。

  Unity 6 では、透明なスクリーンスペースリフレクションのための Mixed トレーシングモードにより、レイトレーシングによるリフレクションのみを使用するのではなく、両方のトレーシングモードを期待どおりに組み合わせるようになりました。例えば、このシーン では、崖表面のデフォームしないジオメトリでは見えない、葉のパーティクル、白いデカール、ゲームオブジェクトの反射を加えることができます。



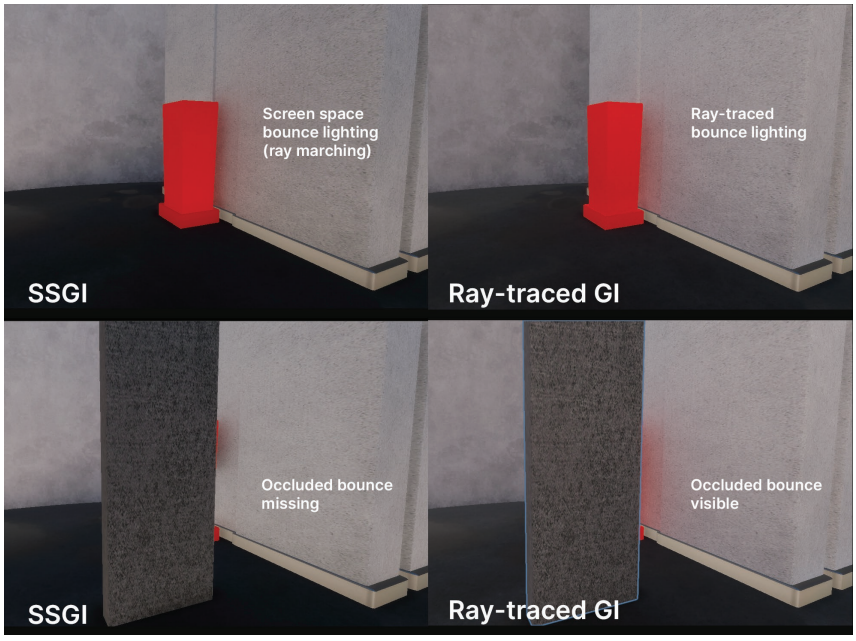
Mixed (ミックス) トレーシングモード

レイトレーシングによるグローバルイルミネーション (RTGI)

RTGI は、間接光の計算にハードウェアアクセラレーションによるレイトレーシングを使用します。これにより SSGI よりも高い精度が得られ、画面外のオブジェクトにも作用しますが、計算コストは高くなります。

レイトレーシングは、RTX 対応 GPU を搭載したプラットフォームでのハイエンドのリアルタイムレンダリングに最適です。

HDRP アセットで DXR のサポートを有効にし、HDRP 設定で **Ray-Traced Global Illumination** (レイトレーシングによるグローバルイルミネーション) を有効にします。



レイトレーシングによるグローバルイルミネーションは、画面外のオブジェクトに対して作用します。

詳細については、レイトレーシングの章 を参照してください。

Enlighten Realtime GI 廃止予定

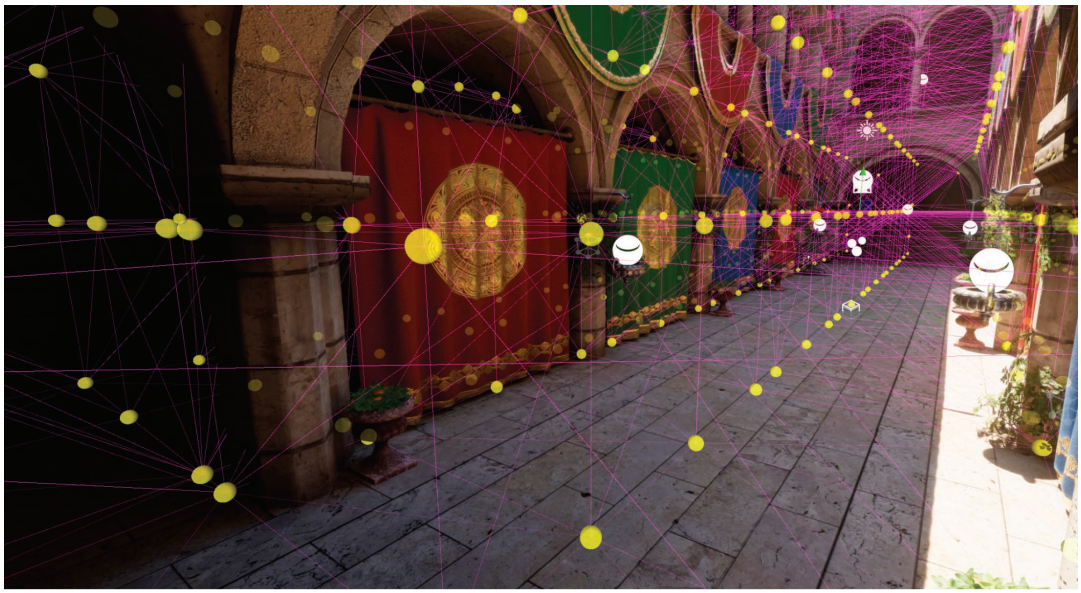
なお、Unity 6 は **Enlighten Realtime GI** (**Lighting (ライティング) > Realtime Global Illumination (リアルタイムグローバルイルミネーション)** 設定で利用可能) をサポートする最後のリリースです。詳細については、以前のフォーラム投稿 Update on Global Illumination 2021 を参照してください。

ライトプローブと Adaptive Probe Volume

ベイクした、またはリアルタイムのグローバルイルミネーションは、静的な環境にリアルなライティングを追加できます。しかし、動的なオブジェクトには、ライトプローブと呼ばれる別のライティングテクニックを使うことが可能です。

ライトマップと同様に、ライトプローブには、シーンのライティングに関する "ベイク" 情報が格納されます。違いは、ライトマップはシーン内のサーフェスに当たるライトに関する情報を格納しているのに対し、ライトプローブはシーン内の何もない空間を通り抜けるライトに関する情報を格納しているという点です。

ライトプローブを使用して、Level of Detail (LOD) システムを活用し、動くオブジェクトや静止した風景にライティングを適用できます。プローブは、間接的な反射光も含めた、高品質の光をキャプチャします。



ライトプローブは、何もないスペースのライトをサンプリングします。

ライトプローブと SSGI/RTGI の比較

カメラ、ライトプローブ、アンビエントプローブに影響する Volume オーバーライドでスクリーンスペースグローバルイルミネーションまたはレイトレーシンググローバルイルミネーションを有効にすると、そのボリューム内のゲームオブジェクトのライティングに関与しなくなります (詳細については、「動的グローバルイルミネーション」を参照してください)。

ただし Adaptive Probe Volume (APV) は、スクリーンスペースエフェクトでカバーされない領域の間接光情報を提供することで、SSGI と RTGI を補完できます。

Light Probe Group

Light Probe Group (ライトプローブグループ) コンポーネントを使用して、基本的なライトプローブの設定を確認できます。

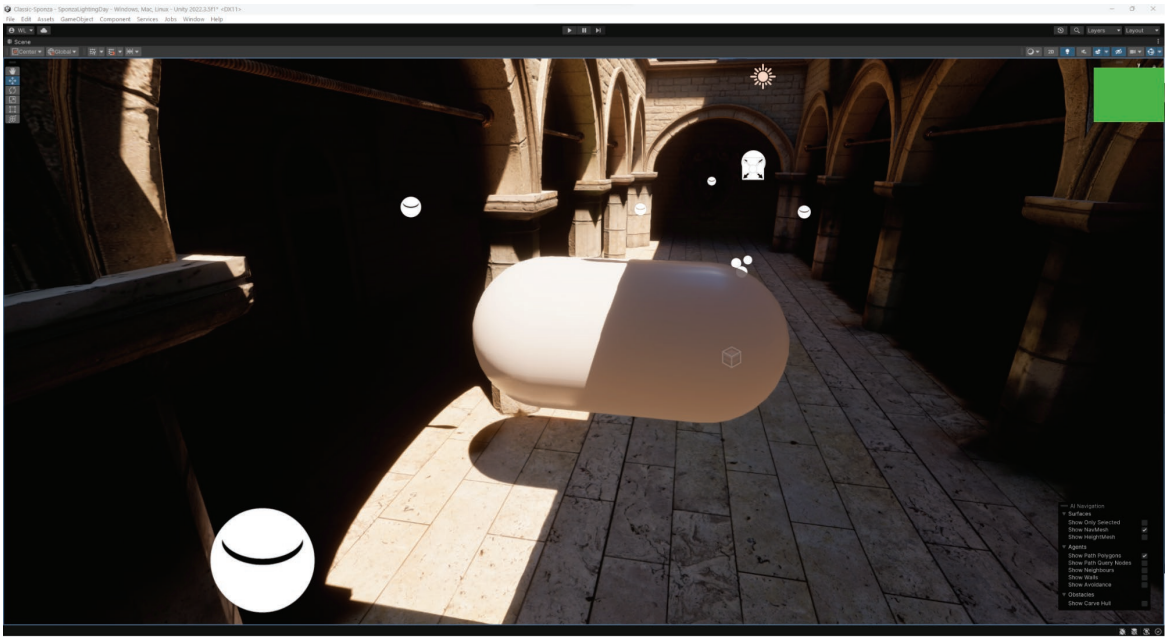
シーンにライトプローブを配置するには、空の ゲームオブジェクトに Light Probe Group コンポーネント (**Component > Rendering > Light Probe Group**) を追加します。次に、コンポーネント設定を使用して、プローブの位置を編集したり、プローブを追加したり、削除したりします。

動的オブジェクトが移動する可能性のある領域内のライティングが明らかに変化する場所 (出入り口近く、屋内と屋外の境界など) を中心に、グリッド状またはクラスター状にライトプローブを配置します。

ライトプローブを設置した後、シーンのアンビエントライトをキャプチャするためにベイクする必要があります。ベイクしたライトマップの場合と同様に、Lighting ウィンドウの **Generate Lighting** ボタンを使用します。

プローブライティングはランタイムのコストが比較的低く、事前計算も短時間で完了します。ライトマップはアンラップに時間がかかり、メモリを大量に消費する可能性がありますが、ライトプローブはそれらを必要としません。

しかし、ライトプローブで照らされたオブジェクトは、最も近い 4 つのプローブ間の結果をブレンドするため、ライティングが不正確になることがあります。下の画像では、ライトプローブを通してカプセルを動かすと、特にシーンの明るい部分と暗い部分の間を移行するときに、ライティングが不自然になるのがわかります。



ライトプローブは低コストである一方、結果が不正確になることがあります。

ライトプローブは計算が早いものの、手動配置が必要です。ライティング環境のサンプリングは反復プロセスであり、手間のかかる作業となります。このため、ライトプローブをシーン内に個別に配置するのではなく、Adaptive Probe Volume (後述) の使用を検討してください。

Light Probe Group の詳細については、ライトプローブのドキュメントページ を参照してください。

ランタイムのライトプローブの移動

Unity 6 には、ライトプローブを使った制作作業を効率化する新機能もいくつかあります。

新しい LightProbes API を使用すると、スクリプトを使用して、従来型ライトプローブの位置をランタイムに変更できます。これは、アプリケーションがモジュール環境またはプロシージャル生成された環境を使用している場合に便利です。

例えば、複数のシーンに分割され、ランタイムに加算的にロードされるゲーム世界では、場合によっては環境のさまざまな部分を適切な場所に移動させる必要があります。この場合、対応するライトプローブも再配置しなければいけません。そうしないと、ライティングが一致しないためです。

ただし、Light Probe Group の Transform の調整によってランタイムにライトプローブの位置を再配置するだけでは、上手くいきません。これはベイクの位置にしか影響しないからです。代わりに、新しい LightProbes API を使用すると、プローブデータをクローンし、プローブの位置を更新してから、スクリプトを介してプローブネットワークを再構築できます。この API には、複数のプローブを一度に効率的に移動させるためのパフォーマンスの最適化も備わっています。

小規模なシーンから環境を作成している場合、ライティングの一貫性を維持しつつライティングを事前にベイクし、ゲーム世界のモジュール化された部分をランタイムに再配置することができます。

実装例については、Unity Light Probes Runtime Movement Documentation (Unity ライトプローブランタイムの移動に関するドキュメント) を参照してください。

アダプティブプローブボリューム

Unity 6 の Adaptive Probe Volume (APV、アダプティブプローブボリューム) は、シーンジオメトリに基づいてライトプローブの配置を自動化するので、Light Probe Group (ライトプローブグループ) に手動で配置する必要がなくなりました。オブジェクトごとのプローブとは異なり、APV はピクセルごとのサンプリングを使用するため、ライティングの遷移がより滑らかになり、ゲームオブジェクト全体での一貫性が高まります。

APV はピクセル単位で機能するため、手動で配置したライトプローブに起因するライティングの問題の多くを回避できます。下の比較は、Light Probe Group (左) と APV (右) のこの違いを示しています。

Light Probe Group と APV

Light Probe Group では、各オブジェクトが 1 つの補間プローブをサンプリングするため、別々のパーツ間でライティングが一致しません。APV はピクセル単位のサンプリングを使用するため、ライティングの遷移がより滑らかになります。

以下の表は、Light Probe Group と APV の機能を比較したものです。

機能	ライトプローブグループ	Adaptive Probe Volume (APV)
プローブの選択	ゲームオブジェクト単位	ピクセル単位
ストリーミングでメモリ使用量を最適化	いいえ	はい
プローブを自動的に配置	いいえ	はい
異なるベイク間のブレンド	いいえ	はい
プローブの手動配置	はい	いいえ

APV の利点:

- **ピクセル単位のライティング:**APV　により、オブジェクト間の遷移がより滑らかになり、継ぎ目が少なくなります。
- **フォグ精度の向上:**APV　はボリュメトリックフォグのライティングのバリエーションをより正確に表現します。
- **プローブの自動配置:**APV のベイクでは、シーンの複雑さに基づいてプローブ密度を調整し、重要な領域のディテールを微調整します。
- **複数シーンのサポート:** Baking Set を使用すると、複数のシーンをまとめてベイクできます。
- **ストリーミングモード:**この機能は、大規模なオープンワールドのメモリ使用量を最適化します。
- **スカイオクルージョンの更新:**APV は、ランタイムに空からのライティングを動的に調整できます。
- **デバッグツール:**Rendering Debugger（レンダリングデバッガー）は、プローブのレイアウトと密度レベルを可視化し、トラブルシューティングを容易にします。

APV の制限:

- **プローブの固定配置:**ライトプローブの位置を手動で調整することはできません。このために壁や境界付近でアーティファクトが発生する可能性があります。
- **Light Probe Group からの変換不可:**既存の Light Probe Group を APV に変換できません。

この APV ボリュームレンダリングを、手動で配置したライトプローブと比較します。多くのシーンで、ライトプローブを数秒で設定できます。

プローブボリュームはライトプローブのレンダリングを向上させます。

HDRP Quality の設定または HDRP アセットの **Lighting (ライティング) > Light Probe Lighting** で、APV を有効にします。**Adaptive Probe Volumes** と **Light Probe Groups** はどちらかしか有効にできません。

APV は、64 個 (4 × 4 × 4) のプローブで構成される "ブリック" を使用して、プローブをシーン全体に自動的に分散させます。シーン内のジオメトリ密度に適応する Volume ベースのシステムを使用しています。

これらのプローブ間の間隔はさまざまです (1、3、9、または 27 ユニットで、これによりブリック全体のサイズが決定されます)。ジオメトリの密度が高いエリアでは、HDRP は密に詰まったブリックを使用するため、より解像度の高いライティングデータが得られます。密度が低い場所では、プローブの間隔も離れています。これにより、必要なところにリソースを集中させられます。

ベイクしたプローブボリューム内の個々のプローブを可視化すると、下のようになります。

APV 内のライトプローブの間隔は変動します。

Sponza Palace とは?

グローバルイルミネーションレンダリングのリファレンスモデルとして広く使われている Sponza Atrium 3D コンピュータグラフィックスモデルは、16 世紀のクロアチアの宮殿をもとに作られています。HDRP でリマスターされたバージョンは GitHub リポジトリにあります。

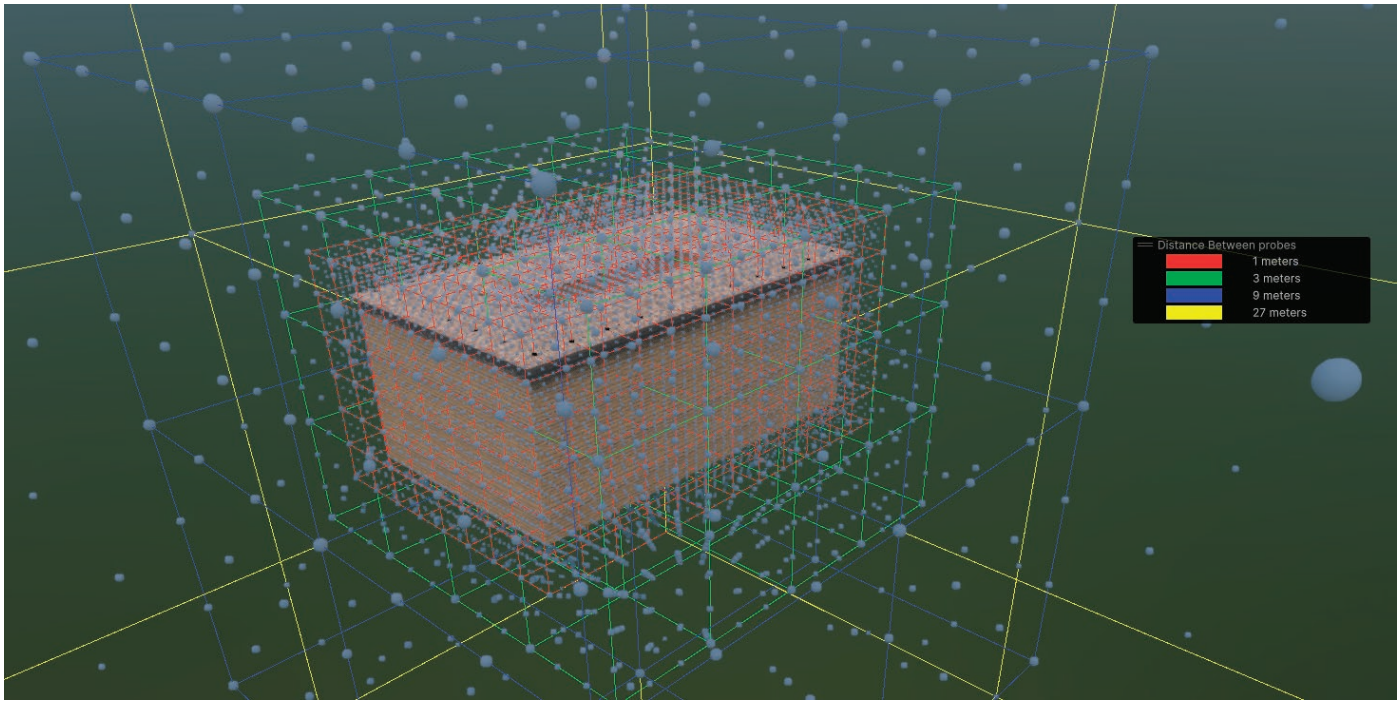
Adaptive Probe Volume の使用

Hierarchy ウィンドウで右クリックして、**Light (ライト) > Adaptive Probe Volume (アダプティブプローブボリューム)** (APV) を選択します。

Mode を **Global** (グローバル) に設定し、デフォルトを使用するか、特定の **Subdivision Override** を選択します。**Override Probe Spacing** で新しい値範囲を選択します。

Bake Probe Volumes をクリックしてボリュームをベイクします。その後、ジオメトリに基づいてプローブがシーンに配置され、ジオメトリが多い場所に、より密にパッキングされます。

ライトプローブ自体を可視化するには、**Analysis (分析) > Rendering Debugger (レンダリングデバッガー)** を開きます。**Probe Volumes** を選択し、**Display Probes** を有効にします。異なる解像度を表示するには、**Display Bricks** を選択します。



Rendering Debugger を使用して APV を可視化します。

なお、APV　内のライトプローブの位置を個別に手動で調整することはできません。つまり、そのプローブが壁や光の境界と常に完全に一致しているとは限りません。ただし、プローブの配置と密度をより適切に制御するために、サブディビジョンの設定が異なる複数の APV を使用できます。

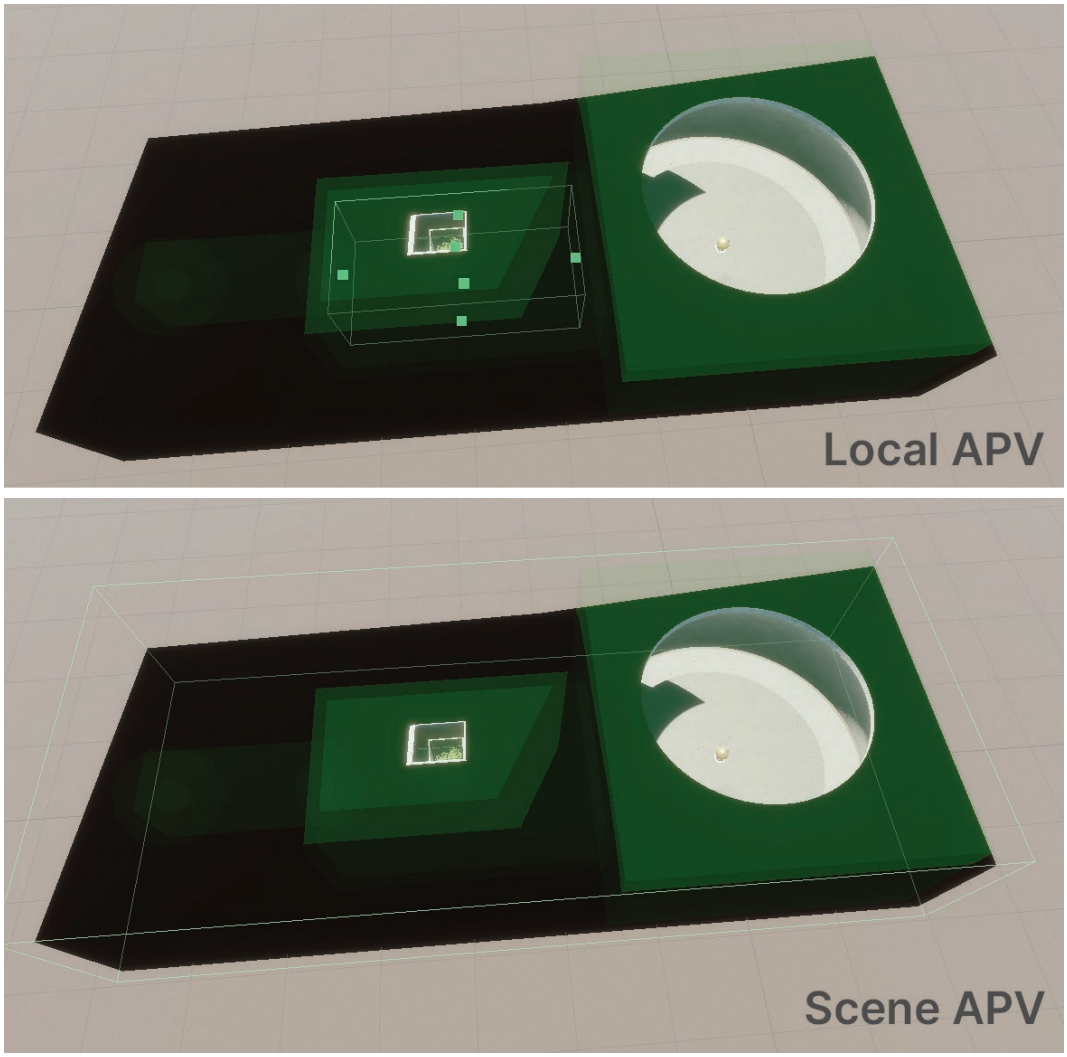
HDRP Sample テンプレートシーンも、同様の設定を使用しています。シーン内に 2 つのプローブボリュームがあります。APV　はモードを使用して、HDRP　がボリュームの大きさを決定する方法と、どのレンダラーがボリュームに影響を与えるかを制御します。

— **Global:**APV は、Baking Set または Contribute Global Illumination でマークされたシーン内のすべてのレンダラーを含むように自動的にサイズ変更します。HDRP は、ライティングを保存または再生成するたびにこのボリュームサイズを再計算します。

— **Scene:**APV は、現在のシーン内のすべてのレンダラーが含まれるように自動的にサイズ変更します。

— **Local:**APV　は自動ではサイズ変更しません。代わりに、Size　を使用してボリュームを手動で設定します。

Sample テンプレートでは、レベル全体を囲む 1 つの APV の **Mode** は **Scene** に設定され、2 つ目の部屋の別の APV の **Mode** は **Local** に設定されています。

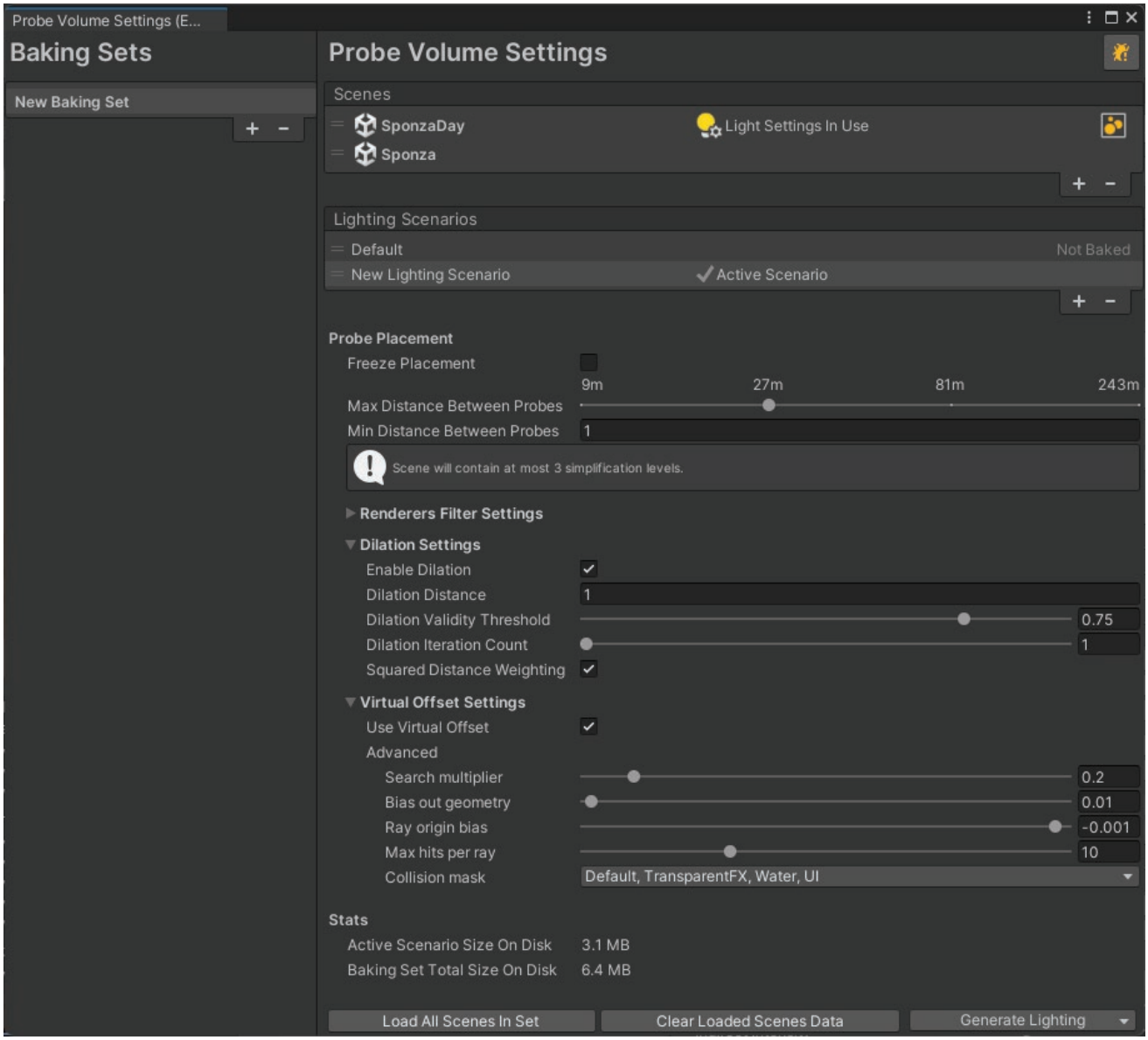


正確に制御するには、シーン内で複数の APV を使用します。

次のいずれかでプローブボリュームをベイクします。

— **Adaptive Probe Volume コンポーネント**:**Bake Probe Volumes**を選択して、特定のゲームオブジェクトからの APV のみをベイクします。

— **Lighting ウィンドウ**:**Generate Lighting** を選択して、リフレクションプローブやその他のベイクしたライティングとともに APV をベイクします。



プローブボリュームの設定

場合によっては、APV に完全に切り替え、ライトマップを完全に削除し、すべてのシーンオブジェクトに単一の一貫したライティングシステムを使用することで、ワークフローを簡素化できます。

HDRP は効率化のため、ベイクしたデータを複数に分割します。複数の Lighting Scenario を使用する場合、プローブの配置とジオメトリがベイク間で同じである限り、プローブボリュームはディスク上のベイクされたデータを複製しません。

ベイクしたライトマップ (左) と APV のみ (右) の比較

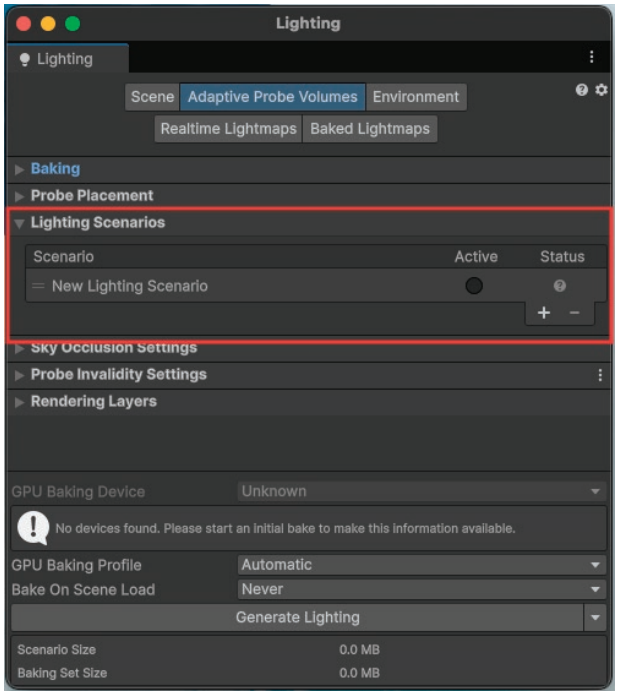
プローブボリュームは、ベイクしたライトマップのニュアンスをすべて捉えることはできないかもしれませんが、スクリーンスペースアンビエントオクルージョン などのリアルタイムエフェクトと組み合わせて、細部を補うことができます。これは、ライトマップに依存せずに間接光を実現するための方法の 1 つです。

プローブボリュームの設定とプロパティの完全版リストを参照してください。また、ベイクした GI と APV を適用する方法については、Four techniques to light environments in Unity (Unity で環境のライティングを行う 4 つの手法) をぜひ参照してください(自動保翻訳機能推奨)。

ライティングシナリオアセット

APV は、**Lighting Scenario** (ライティングシナリオ) アセットを使用して間接光データを切り替えることもできます。各 Lighting Scenario には、ベイクされたライティング状態が格納されます。異なるライティング設定を異なる Lighting Scenario にベイクし、ランタイムに HDRP が使用する設定を変更することができます。Lighting Scenario は、シーンまたは Baking Set のいずれかと一致させることができます。

例えば、昼用と夜用の 2 つのライティングシナリオを作成できます。スクリプトを使用して、ランタイムに 2 つのシナリオを切り替えたりブレンドしたりできます。エディターでは、Rendering Debugger を使用して効果をプレビューできます。



Lighting ウィンドウで Lighting Scenario を適用します。

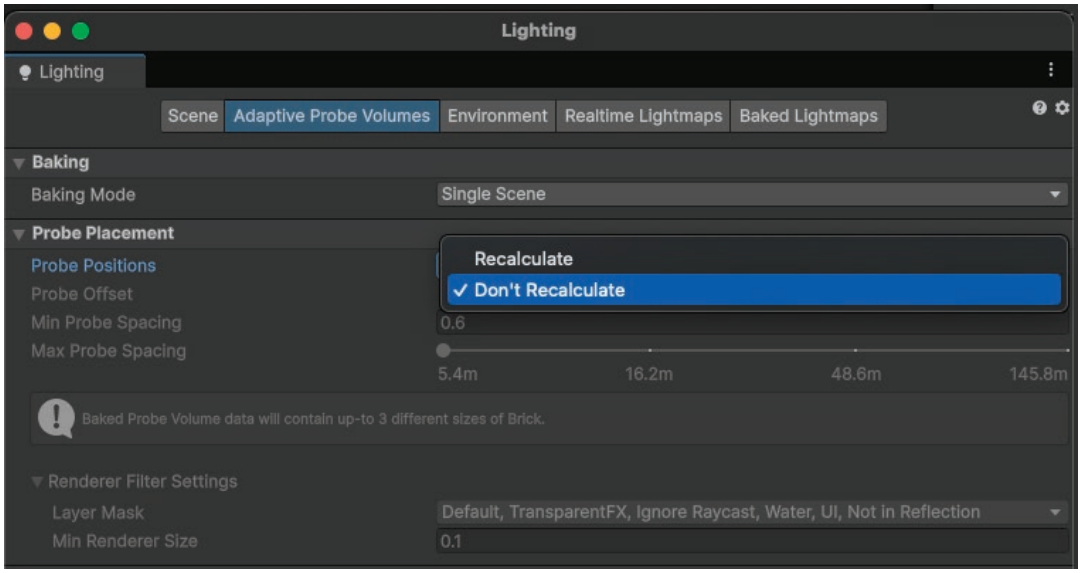
ライティングを瞬時に切り替えるのではなく、**Scenario Blending** を使用して 2 つの Lighting Scenario を徐々に切り替え、より自然に変化させることができます。これにより、急激な変化を回避し、より没入感のある昼夜のサイクルやライティング状態の変化が可能になります。

同じ環境で異なる Lighting Scenario を使用。

Lighting Scenario アセットを使用するには、アクティブな HDRP アセットに移動し、**Lighting >**

Light Probe Lighting > Lighting Scenarios を有効にします。次に、ベイク結果を格納する新しい Lighting Scenario アセットを作成します。

1. **Lighting** ウィンドウの **Adaptive Probe Volumes** パネルを開きます。
2. **Lighting Scenarios** セクションで、**Add** (+) ボタンを選択して、Lighting Scenario アセットを追加します。
3. **Lighting** ウィンドウで、**Adaptive Probe Volume** タブの下にある **Probe Positions** が **Don't Recalculate** に設定されていることを確認します。これにより、Unity はプローブの位置を変更せずにライティングのみをリベイクします。プローブの位置が変更されると、以前にベイクされたシナリオが無効になる可能性があるためです。



Lighting Scenario を使用するときは、プローブの位置を再計算しないでください。

リストから Lighting Scenario を選択して有効にし、**Generate Lighting** を選択します。HDRP は、有効な Lighting Scenario にベイク結果を保存します。

ライトマップを使用していない場合は、**Generate Lighting** の隣にあるドロップダウンボタンを使用して、**Bake Probe Volumes** を選択することもできます。

ProbeReferenceVolume API を使用してランタイムで現在使用されている Lighting Scenario を設定し、BlendLightingScenario API を使用してそれらをブレンドします。

Rendering Debugger (**Window > Analysis > Rendering Debugger**) を使用して、Lighting Scenario 間の遷移をプレビューすることもできます。Scenario Blend Target を Lighting Scenario に設定し、Scenario Blending Factor を使用してブレンド効果を確認します。

Scenario Blending Factor は、ベイクした 2 つのライティング状態間の補間を制御します。これは、動的な時間帯の効果、屋内と屋外の遷移、同一シーン内の異なるライティングの雰囲気に使用できます。Scenario Blending によって、Lighting Scenario の変更時に突然 "パッ" と切り替わることなく、滑らかで徐々に遷移させることができます。

Scenario Blending の例は、Time Ghost: Environment アセットで確認できます。ここでは、MonoBehaviour によって異なる Lighting Scenario がブレンドされ、1 日の時間経過におけるスムーズな遷移をシミュレートしています。

Lighting Scenario マネージャー

ランタイムにアクティブな Lighting Scenario を変更すると、HDRP はライトプローブ内の間接光データのみを変更します。ジオメトリの移動、ライトの修正、または直接光の変更には、スクリプトを使用する必要がある場合があります。

参考として、Time Ghost: Environment サンプルの ScenarioManager スクリプトを参照してください。

APV の問題を修正する

APV はプローブを自動的にグリッドに配置するため、配置によってレンダリングエラーが発生することがあります。

プローブがジオメトリの内側にあり、そのサンプリングレイがライティングされていない裏面に当たると、無効なプローブが発生し、正確なライティングが妨げられます。結果として、本来明るいはずの部分が暗くなったり、その逆になったりする場合があります。

この例では、3D Sample シーンの床下に配置された無効なプローブが、カプセルの下部に暗いアーティファクトを発生させています。

無効なプローブが、暗いアーティファクトを発生させています。

エディターには、テクニカルアーティストがこれらの問題を特定して修正するためのツールがいくつか用意されています。

— **Rendering Debugger:Probes Volumes > Debug Probe Sampling** に移動して、特定のピクセルに影響を与えているプローブを確認します。

— **プローブ無効設定:Lighting** ウィンドウの **Adaptive Probe Volumes** パネルに移動します。これらの設定を使用して、各プローブのキャプチャポイントをオフセットし、サンプリングを改善します。

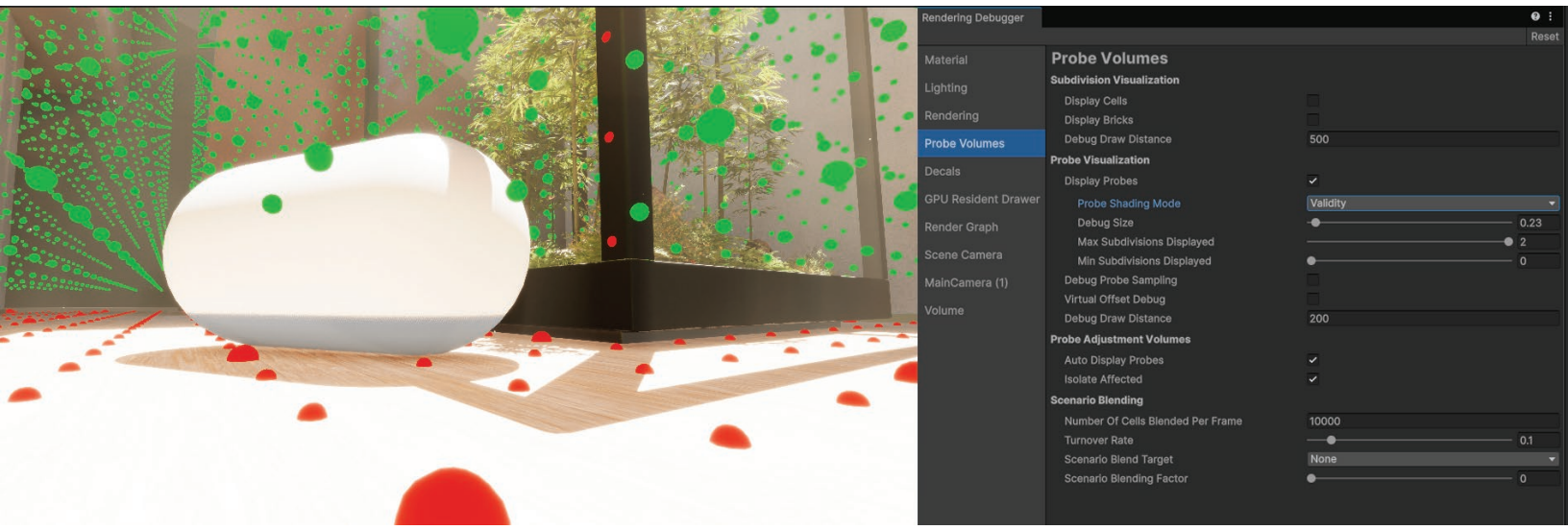
APV に関する、トラブルシューティングが必要なよくある問題と解決策をいくつか見てみましょう。

無効なプローブの修正

よくある問題の 1 つは、ジオメトリ内にプローブが生成されることです。オブジェクト内の裏面をサンプリングするプローブは、ライティングが正しくない可能性があり、無効と見なされます。

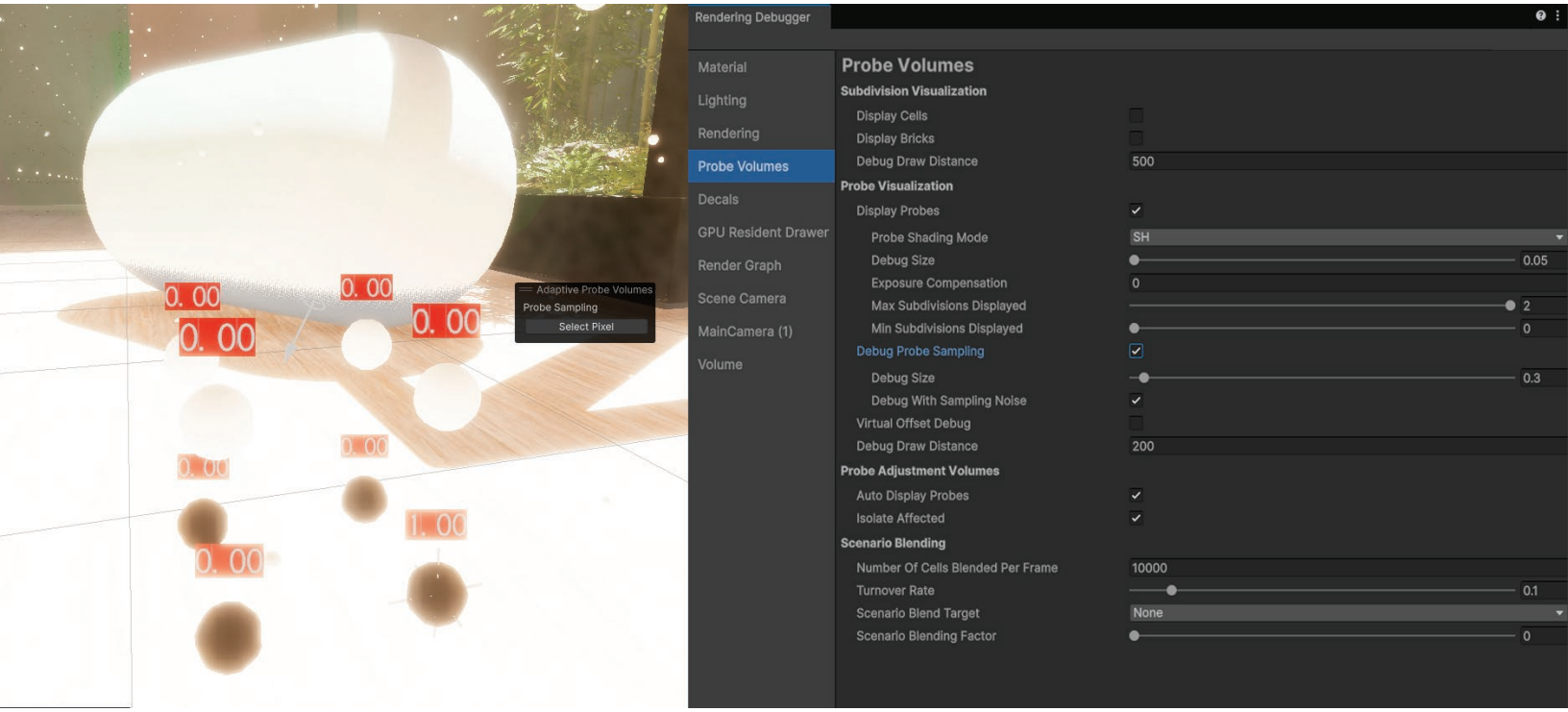
トラブルシューティングには、**Rendering Debugger** (**Window > Analysis > Rendering Debugger > Probe Volumes**) を使用します。**Display Probes** オプションを有効にし、**Probe Shading Mode** を **Validity** に設定して、無効なプローブを可視化します。

上記の例では、ゲームビューに APV の個々のプローブが表示されます。有効なプローブは緑、無効なプローブは赤で表示されます。



Rendering Debugger (レンダリングデバッガー) に無効なプローブが赤で表示されます。

特定のアーティファクトの原因となっている可能性のあるものを分離するには、**Debug Probe Sampling** を有効にします。シーンビューで **Select Probes** を使用し、カプセルの裏側をクリックします。このデバッグビューには、特定のピクセルに影響を与えているプローブが表示されます。



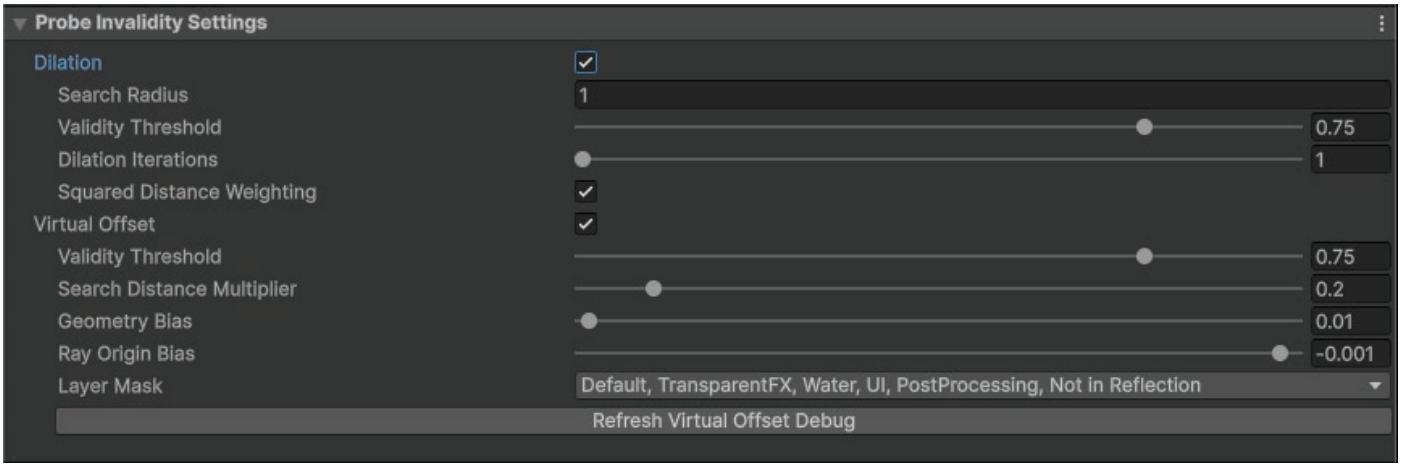
Debug Probe Sampling オプションでは、問題の原因となっているプローブが表示されます。

Probe Invalidity Settings (**Lighting** ウィンドウの **Adaptive Probe Volumes** パネル) を調整して、無効なプローブによる暗い領域を修正します。

— **Virtual Offset** は、光線を照射してジオメトリの外側にある有効な位置を探し、各プローブのキャプチャ位置をずらします。**Validity Threshold**、**Search Distance Multiplier**、**Layer Mask** などの設定を調整して、プローブの配置を改善します。**Geometry Bias** はプローブを周囲の表面から離し、**Ray Origin Bias** はプローブレイの開始位置を調整して、配置精度を高めます。

— **Dilation** 設定は、近くの有効なプローブのライティングデータを流用して、欠落しているデータを埋めることで、無効なプローブを修正するのに役立ちます。**Search Distance**、**Validity Threshold**、**Iterations**、およびその他の拡張設定を使用してこのプロセスを微調整し、プローブの検証における厳密さを制御します。**Squared Distance Weighting** は、流用したライティングデータをより自然にブレンドできます。

設定を試し、暗い筋が消えるまで Prove Volume をベイクします。Virtual Offset と Dilation はどちらもベイク処理にのみ影響し、ランタイムのパフォーマンスには影響しません。



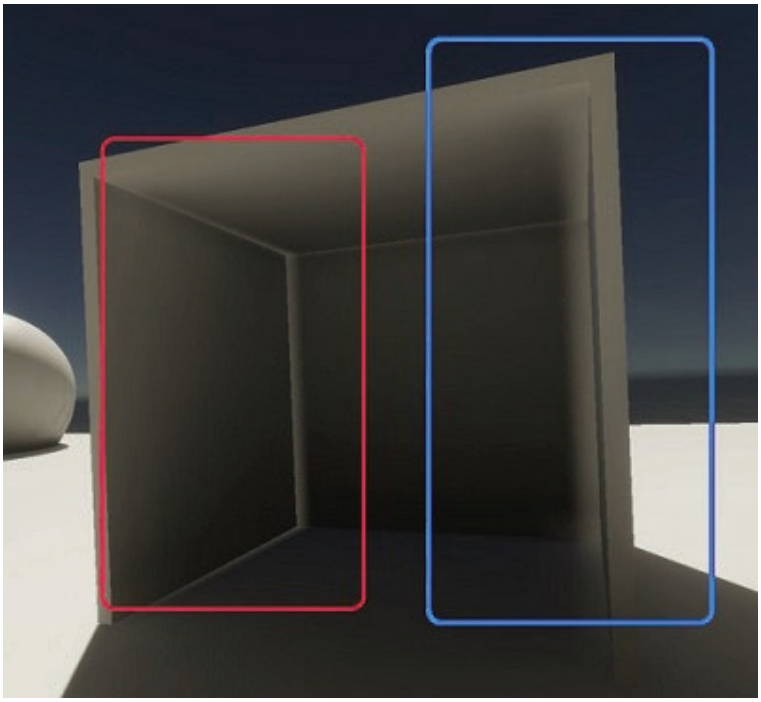
Probe Invalidity Settings を使用して、無効なプローブによる暗い領域を修正します。

Virtual Offset と Dilation の設定を調整した後は、アーティファクトがなくなっています。

ライトリークの修正

無効なプローブによる暗い筋と同様に、"ライトリーク" は、(ライトプローブが壁の反対側にあるなど) ジオメトリからは見えないライトプローブからの光をジオメトリが受け取ったときに発生する可能性があるアーティファクトです。

これは、環境内の APV 密度と壁厚が一致しない場合に発生し、本来は不透明である壁を外部のライトが透過したように補間されて見える現象です。



ライトリークはアーティファクトとして現れます。

ライトリークは明るすぎたり暗すぎたりする領域で生じ、壁や天井の隅などによく現れます。APV はライトプローブを規則的なグリッドに配置するため、プローブが壁に沿わなかったり、異なるライティングエリアの境界に配置されないことがあります。

ライトリークを修正するには、以下の方法を使用します。

- プローブをより適切に配置するために、壁を厚くします。壁の幅が局所的な ブリック のプローブ間距離に近づくように壁を調整します。
- Adaptive Probe Volumes Options オーバーライド (後述) を使用してサンプリング位置を調整します。
- Rendering Layer Masks (後述) を有効にして、オブジェクトが誤ったレイヤーでプローブを使用しないようにします。
- Baking Set プロパティを変更して、プローブ間隔と検索距離を最適化します。

Probe Adjustment Volume を使用する

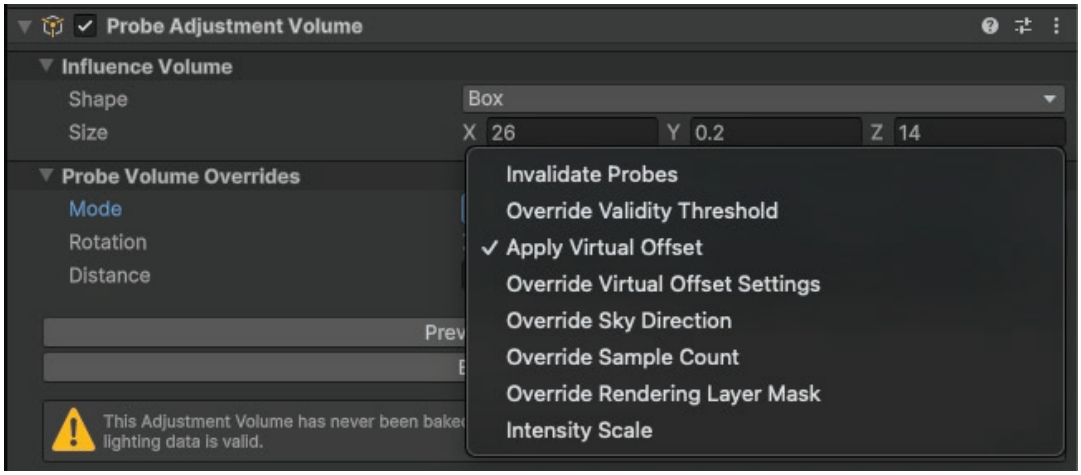
Volume システムには、**Probe Adjustment Volume Option オーバーライド** があります。このオーバーライドにより、ライトプローブと APV を局所的に制御できるようになり、シーンの特定領域におけるプローブベースのライティングを調整できます。

Influence Volume (インフルエンスボリューム) 設定を使用して、影響を与える **Box** (ボックス) または **Sphere** (スフィア) を定義します。次に、強度、サンプリング、空の寄与、プローブの有効性を調整する **モード** を選択します。

- **Invalidate Probes:** プローブが無効としてマークされ、ライティングの計算に関与しなくなります。

- **Override Validity Threshold:**プローブの有効性判定のしきい値を変更し、無効なプローブによるアーティファクトを減らします。
- **Apply Virtual Offset:**Lighting ウィンドウのローカル仮想オフセットの Rotation と Distance がオーバーライドされます。
- **Override Virtual Offset Settings:Validity Threshold**、プローブを近くのサーフェスから離す **Geometry Bias**、プローブレイの開始点を調整する **Ray Origin Bias** の Lighting ウィンドウ設定がオーバーライドされます。
- **Intensity Scale:**ボリューム内の明るさを調整します。
- **Override Sky Direction:**プローブが空の特定の方向を使用するようにします。
- **Override Layer Mask:**影響を受けるプローブを選択するために別の Layer Mask を使用できます。

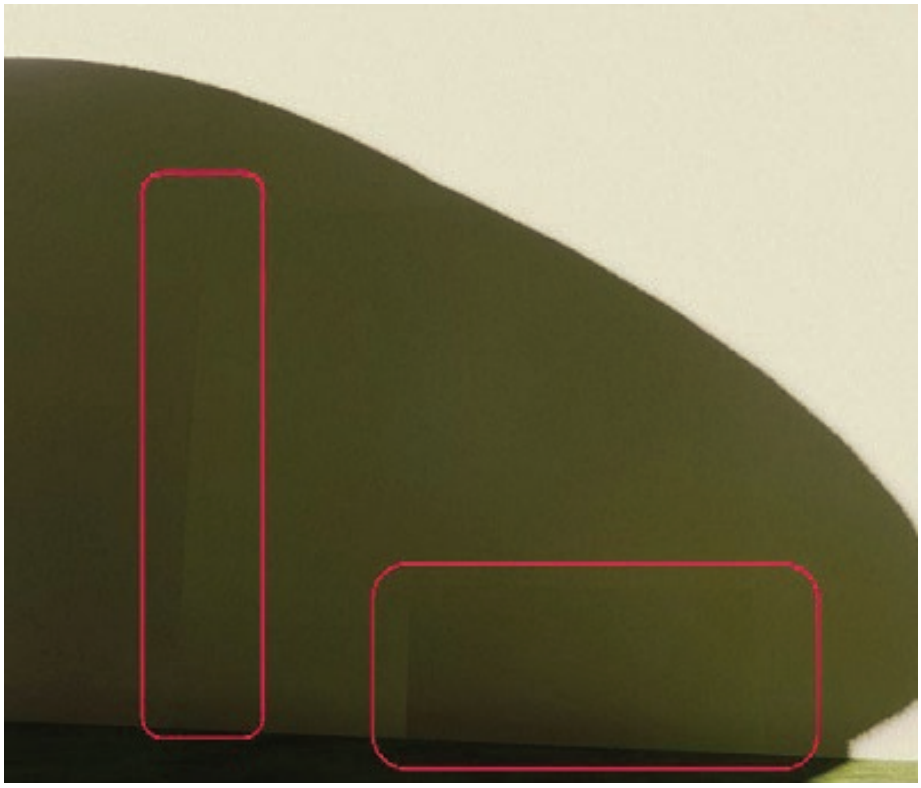
ボリュームは、カメラがボリュームの近くまたは内部にある場合にのみシーンに影響します。Understand volumes (ボリュームを理解する) および Probe Volumes Options Override reference (プローブボリュームオプションオーバーライドのリファレンス) で、**Probe Volumes Options** (プローブボリュームオプション) 設定の詳細を参照してください。



このオーバーライドにより、ライトプローブと APV を局所的に制御できます。

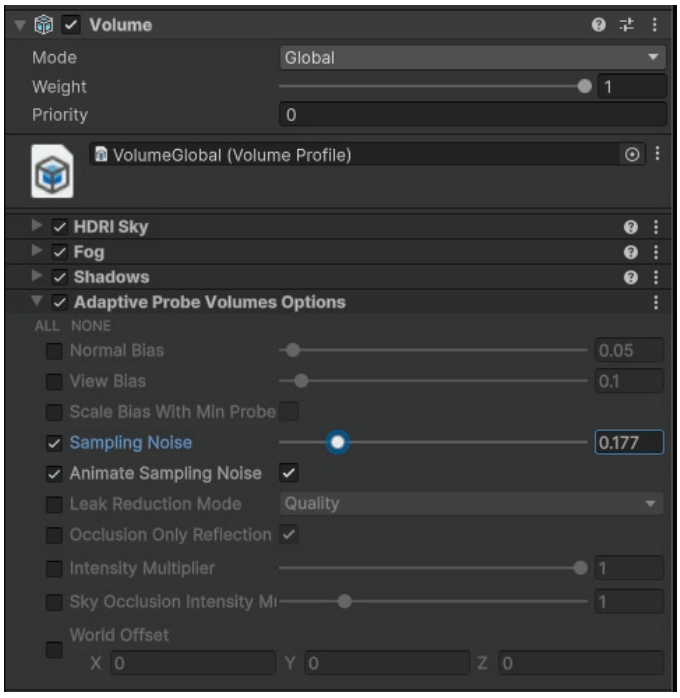
継ぎ目の修正

隣接する 2 つのプローブブリックの密度が異なると、継ぎ目が現れ、ライティングの遷移が突然発生します。HDRP は継ぎ目を自動的に修正しますが、それでも解決しない場合は、Adaptive Probe Volumes Options オーバーライドで **Sampling Noise** を有効にして、遷移を緩和できます。



プローブブリック間の継ぎ目の例。

Baking Set のプロパティやプローブ密度を調整することで、ライティング結果をさらに調整できます。APV の問題の大半はベイク中に発生するため、ライティングの最終決定を行う前に調整することで、シーンのレンダリング効率を最大化できます。



Sampling Noise を調整して継ぎ目を修正します。

Rendering Layer Mask

APV の問題を修正するもう 1 つの方法は、Rendering Layer Mask (レンダリングレイヤーマスク) を使用することです。Rendering Layer Mask は、関連するプローブからオブジェクトがライティングをサンプリングするためのフィルターとして機能し、ライトリークを減らします。APV は最大 4 つの Rendering Layer Mask をサポートします。

ライトベイク中、近くのオブジェクトに基づいて各プローブに Rendering Layer Mask が自動的に割り当てられます。特定の Rendering Layer Mask (例:"Interior") が設定されたオブジェクトの近くのプローブは、そのレイヤーを継承します。

ランタイムでは、オブジェクトは Rendering Layer Mask を共有するプローブからのみライティングをサンプリングします。一致するマスクが見つからない場合、周辺にある有効なすべてのプローブからサンプリングされます。

ライトリークの一般的な原因は、ライティングが大きく異なる内部プローブと外部プローブの間でピクセルがブレンドされることです。これを防ぐために、内部と外部に別々の Rendering Layer Mask を作成できます。

この機能を使用するには、HDRP アセットで **Light Layers** を有効にする必要があります。

APV の Rendering Layer Mask を有効にするには、次の手順を実行します。

1. **Mesh Renderer** コンポーネントでレンダラーに **Rendering Layer Masks** を割り当てます。
2. **Lighting** ウィンドウを開き、**Adaptive Probe Volumes** > **Rendering Layer Masks** に移動します。
3. リストに最大 4 つの Rendering Layer Mask を割り当てます。ベイク時に、ライトプローブは近くのオブジェクトに基づいてマスクに割り当てられます。

HDRP では、Rendering Debugger を使用して、どのレイヤーがオブジェクトに割り当てられているかを可視化できます。

- Material タブに移動します。
- **Material** フィールドを **Common > Rendering Layers** に設定します。

APV におけるスカイオクルージョン

APV のスカイオクルージョンは、影の領域や覆われた領域 (トンネル、部屋、オーバーハングの下など) での空の光の影響を減らすことで、間接光を動的に調整します。これにより、よりリアルな間接光を作成でき、空の影響をあまり受けない領域における過度の明るさを防ぐことができます。

アンビエントプローブ はランタイムに更新できるため、昼から夜への滑らかな切り替えなど、リアルタイムでのライティングの遷移が可能になります。これにより、APV のみを使用して昼夜のサイクルをシミュレートできます。

スカイオクルージョンの有効化

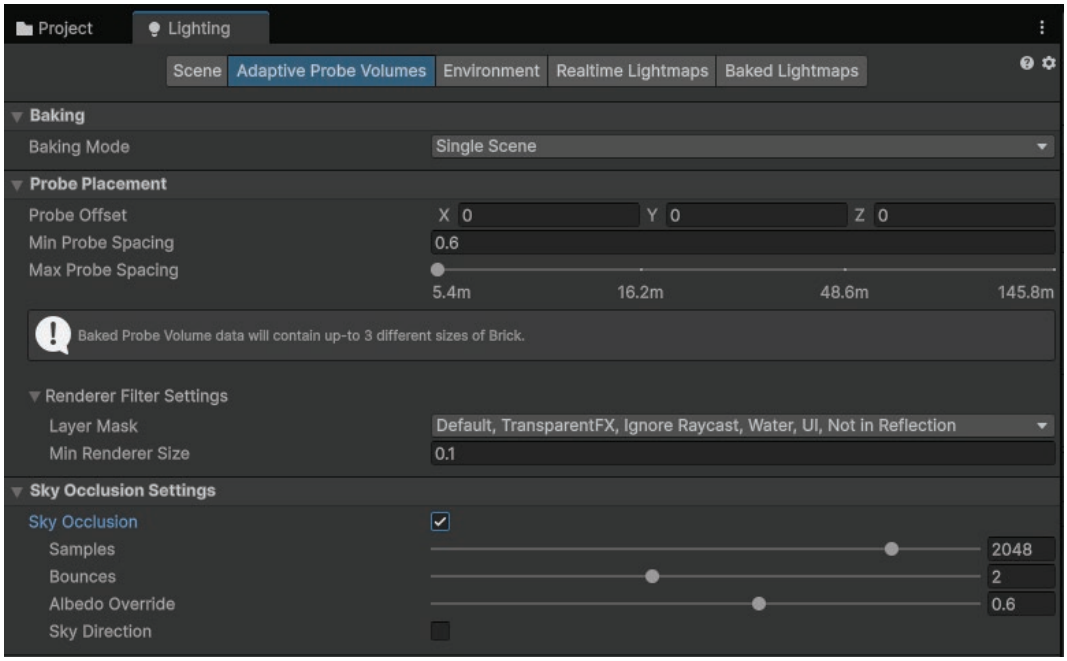
スカイオクルージョンを有効にするには

— Lighting ウィンドウで **Progressive GPU Lightmapper** を有効にします。Progressive CPU を使用している場合、この機能はサポートされません。

— Adaptive Probe Volumes パネルで **Sky Occlusion** (スカイオクルージョン) を有効にし、ボリュームを再ベイクします。

これを有効にすると、空からの間接的な光の受け取り量を示す静的スカイオクルージョン値が、各プローブに追加でベイクされます。これはランタイムにアンビエントプローブの空の色と組み合わされ、空が変化すると動的に更新されます。

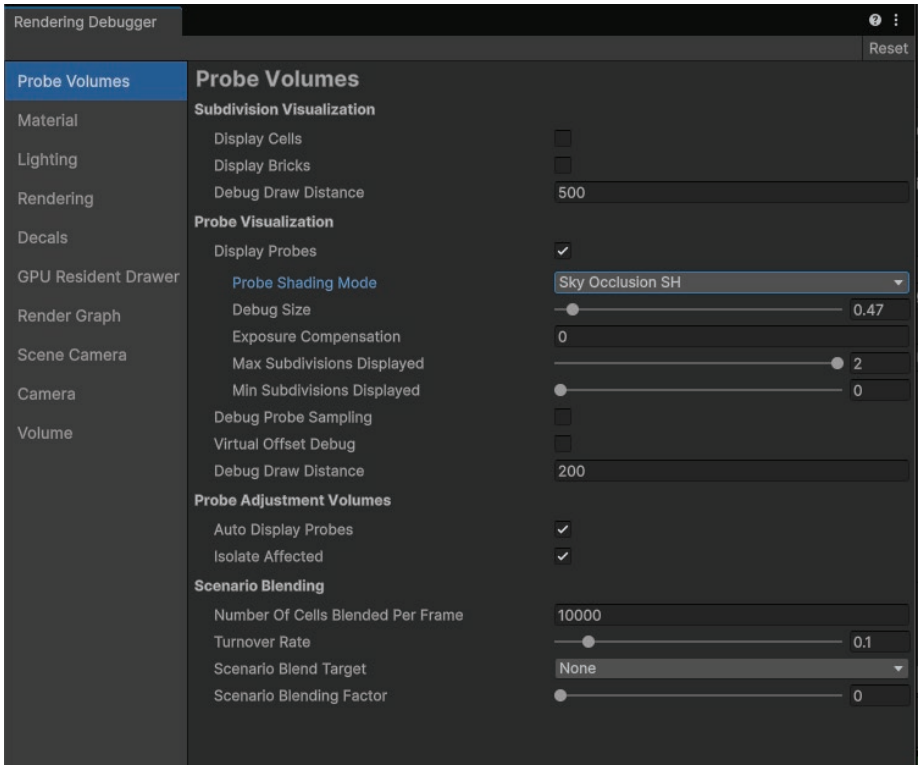
ランタイムにアンビエントプローブが確実に更新されるようにするには、**Volume Profile Asset** を開き、**Visual Environment > Sky セクション**に移動し、**Ambient Mode** を **Dynamic** に設定します。



Lighting ウィンドウで Sky Occlusion オプションを有効にします。

スカイオクルージョンのデバッグ

スカイオクルージョンが APV ライティングにどのような影響を与えるかを **Rendering Debugger** で可視化できます。**Probe Volumes** セクションで、**Display Probes** を有効にし、**Probe Shading Mode** を **Sky Occlusion SH に設定します。**



Rendering Debugger (レンダリングデバッガー) でスカイオクルージョンをデバッグします。

下の画像で**スカイオクルージョン**を有効にした場合 (左) と無効にした場合 (右) を比較してください。スカイオクルージョンを有効にすると、開放領域のプローブは明るく見え、シーンのジオメトリによって遮られたプローブは暗くなり、空の間接的なライティングが適切に減衰されます。スカイオクルージョンを無効にすると、すべてのプローブで空のライティングが均一になり、影の領域が不自然に明るく見えます。

スカイオクルージョン設定を有効にした場合 (左) としない場合 (右)

この違いは、空から遮られた領域で最も顕著です。スカイオクルージョンがベイクされると、アンビエントプローブ の変更に基づいてシーンのライティングが更新されます。HDRP では、Skybox (スカイボックス) モードではなく、Color (カラー) モードまたは Gradient (グラデーション) モードを使用している場合のみ、アンビエントプローブがリアルタイムで更新されます。

空の方向

デフォルトでは、アンビエントプローブをサンプリングする際に、空の方向としてオブジェクトの表面法線が使用されます。これは、空の光が他のオブジェクトに反射する閉鎖的な空間では不正確になることがあります。

Adaptive Probe Volumes パネルで **Sky Direction** (空の方向) を有効にすると、プローブごとに正確な空のライティング方向を計算して保存できます。この計算では、プローブから空への直接の視線がない場所や、窓などの特定の方向から光が差し込む場合に、反射光を考慮することができます。これにより、ベイク時間とメモリの増加と引き換えに、ライティングの精度が向上します。

APV データロードの最適化

ランタイムでの APV データのロードを最適化するには、APV データをストリーミングするか、AssetBundles/Addressables からロードします。

APV データのストリーミング

ストリーミングでは、カメラの動きに合わせて必要なデータだけをロードすることで、大規模な APV ライティングが可能になります。これにより、メモリ使用量を減らし、利用可能な CPU/GPU メモリよりも大きな APV データをベイクできます。HDRP は StreamingAssets フォルダーからデータをストリーミングし、カメラの視錐台内のセルのみをロードします。

APV ストリーミングを有効にするには

1. **Edit > Project Settings > Quality > HDRP** に移動します。
2. **Quality Level** を選択し、**Lighting > Light Probe Lighting** を展開します。

ここで、以下の 2 種類のストリーミングを有効にできます。

- **Enable GPU Streaming** を選択して、CPU メモリから GPU メモリにストリーミングします。
- **Enable Disk Streaming** を選択して、ディスクから CPU メモリにストリーミングします。先に **Enable GPU Streaming** を有効にする必要があります。

ストリーミング設定は、HDRP Asset Settings でさらに詳細に設定できます。

HDRP でロードして使用する最小単位はセルで、これは Adaptive Probe Volume 内の最大の ブリック と同じサイズです。ライトプローブの密度を調整 することで、APV 内のセルのサイズに影響を与えることができます。Rendering Debugger を使用して APV セルを可視化し、ストリーミングをデバッグします。

AssetBundles または Addressables から APV データをロードする

ランタイムに必要な APV データのみをロードするには、Streaming Asset ではなく AssetBundle または Addressables に保存します。

1. AssetBundle-Based APV Loading を有効にし、**Project Settings > Graphics > Pipeline Specific Settings > HDRP に移動します。**
2. **Probe Volume Disable Streaming Assets** を有効にします (ディスクストリーミングを無効にします)。
3. ライティングをベイクします。
4. シーンを AssetBundle に加えます。

注意:Streaming Assets を無効にすると、一部のシーンがロードされていない場合でも、Baking Set のすべてのライティングデータがメモリに保持されるため、メモリ使用量が増加する可能性があります。

ランタイムでのライトプローブのベイク

Unity 6 では、ライトプローブをベイクするための新しい API が導入され、従来の方法と比較して制御性、効率性、柔軟性が向上しています。一度にベイクするプローブの数を制御できるようになり、処理時間とメモリ使用量のバランスを取ることができます。

この API では、**OpenCL** をプローブのベイクに活用した高性能な **GPU アクセラレーションバックエンド**が採用されています。このシステムは、必要なライティングデータをベイク前にシーンから直接抽出します。プロシージャルジオメトリには対応していません。

詳細については、以下のドキュメントページを参照してください。

— RadeonRaysContext

— RadeonRaysProbeIntegrator

— RadeonRaysProbePostProcessor

Adaptive Probe Volumes の詳細については、GDC 2023 のセッション Adaptive Probe Volumes による効率的でインパクトのあるライティング(自動翻訳機能推奨) を参照してください。Adaptive Probe Volume のドキュメント も合わせて参照してください。

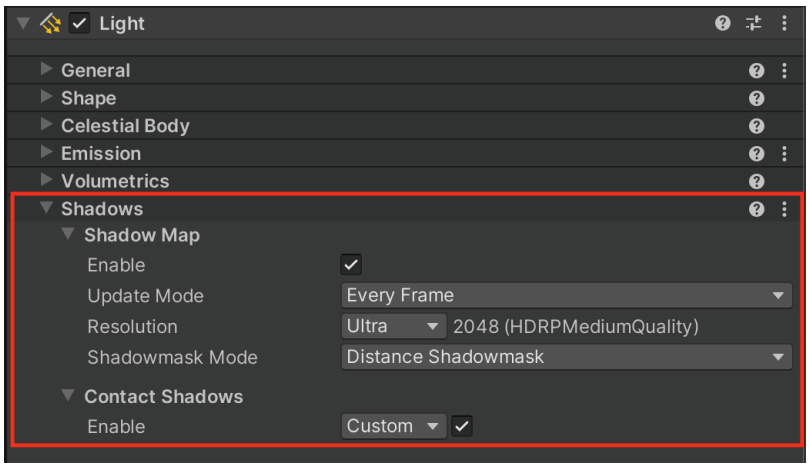
Shadows

シーン内に適切に配置されたシャドウは、ライティングそのものと同じくらい趣を生み出し、シーンに奥行きや立体感を加えることができます。HDRP　には、シャドウを微調整し、レンダリングが単調になるのを防ぐための機能が多数あります。

シャドウマップ

シャドウは、**シャドウマッピング**　という手法でレンダリングされます。この手法では、ライトの視点からの深度情報がテクスチャに保存されます。

Light コンポーネントの Shadows サブセクションで、シャドウマッピングの **Update Mode** と **Resolution** を変更できます。解像度と更新頻度の設定値が高くなるほど、リソースの消費が大きくなります。

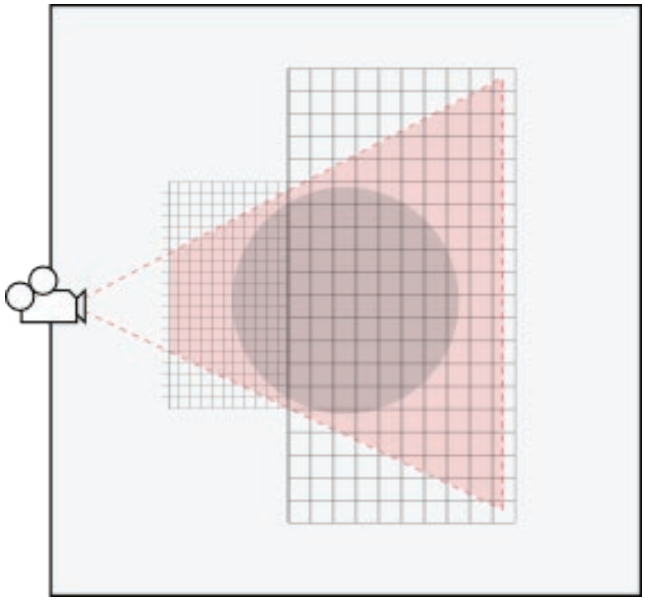


ライトごとのシャドウ設定

Shadow Cascades

ディレクショナルライトの場合は、シャドウマップがシーンの大部分を覆うため、透視エイリアシングと呼ばれる問題が生じることがあります。カメラから近い位置にあるシャドウマップのピクセルは、遠くにあるものよりも、ギザギザで粗くなります。

透視エイリアシングと粗いシャドウ

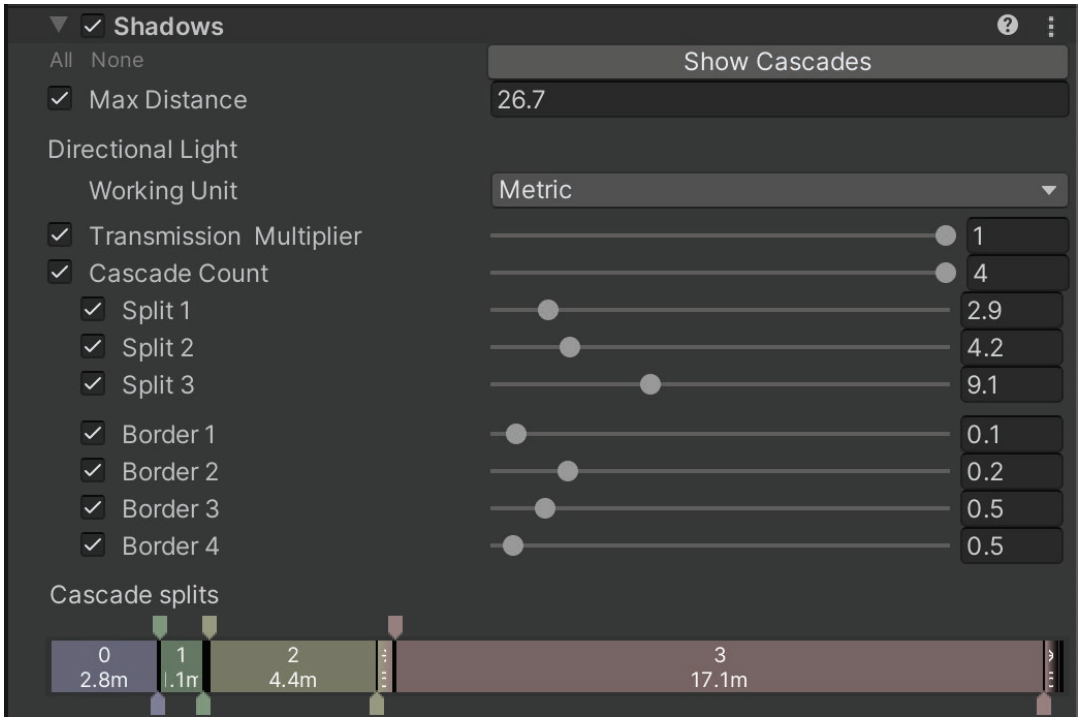


シャドウカスケードでは、カメラの錐台を複数のゾーンに分割します。各ゾーンにそれぞれシャドウマップがあります。

Unity では、カスケードシャドウマップ を使用してこの問題を解決しています。この手法では、カメラの錐台を、それぞれにシャドウマップがある複数のゾーンに分割します。これにより、透視エイリアシングの効果が低減されます。

シャドウカスケードによって透視エイリアシングが低減されます。

HDRP では、**Shadows** オーバーライドで、シャドウカスケードをより詳細に制御できます。ボリュームごとのカスケード設定を使用して、それぞれのカスケードの始点と終点を微調整できます。



Shadows オーバーライドはシャドウカスケードを調整できます。

Show Cascades ボタンを切り替えて、カスケードの分割をより簡単に可視化できます。ある程度の調整を加えることで、透視エイリアシングを最小限に抑えることができます。

Show Cascades オプションをクリックすると、カスケードの分割を可視化できます。

コンタクトシャドウ

シャドウマップは、特に 2 つのメッシュサーフェスが結合する目立ちやすいエッジで、小さなディテールを取り込めないことがよくあります。HDRP では、**Contact Shadows** (コンタクトシャドウ) のオーバーライドを使用して、こうしたコンタクトシャドウを生成できます。

コンタクトシャドウはスクリーンスペースの効果の一種で、フレーム内の情報を基に計算を行います。フレーム外のオブジェクトは、コンタクトシャドウには影響しません。コンタクトシャドウは、画面上のフットプリントが小さいシャドウのディテールに使用します。

Frame Settings で **Contact Shadows** を有効にします。**Lighting Quality Settings** の Pipeline Asset で、**Sample Count** を調整することもできます。

Contact Shadows のオーバーライド

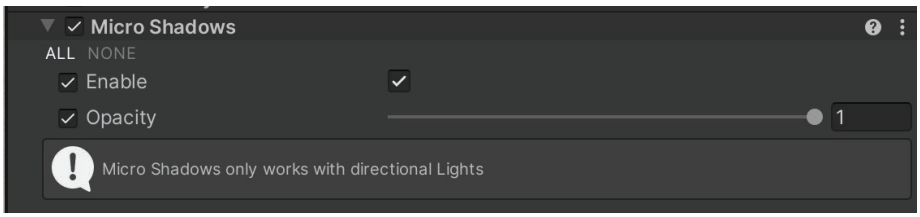
コンタクトシャドウ

シャドウの厚み、品質、フェードの設定を調整します。

この機能と Terrain や SpeedTree との相性は改善されています。詳細は ブログ にて確認してください。

マイクロシャドウ

HDRP では、小さなシャドウのディテールをマテリアルに広げることができます。**Micro Shadows** (マイクロシャドウ) のオーバーライドは、法線マップとアンビエントオクルージョンマップを使用して、メッシュのジオメトリそのものを使用せずに、非常に細かい表面の影をレンダリングします。



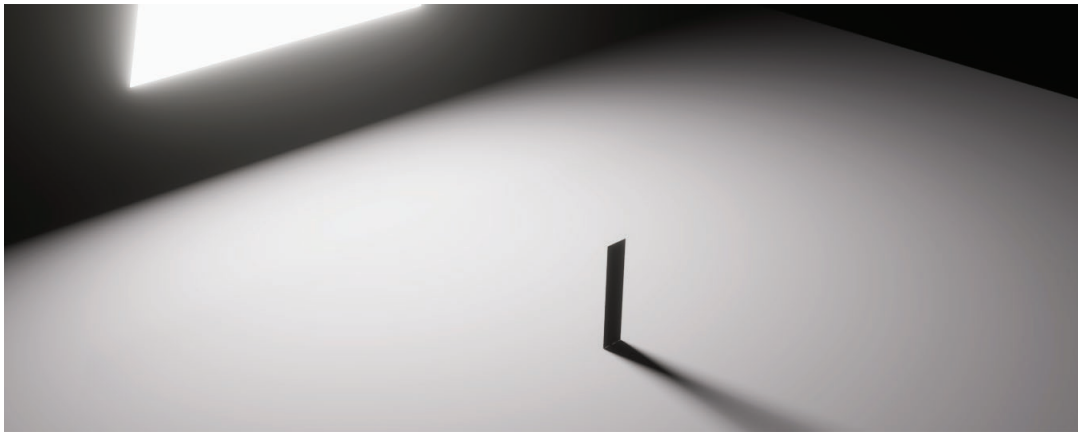
Micro Shadows のオーバーライド

Micro Shadows のオーバーライドをシーン内のボリュームに加え、Opacity (不透明度) を調整します。Micro Shadows を利用できるのはディレクショナルライトのみです。

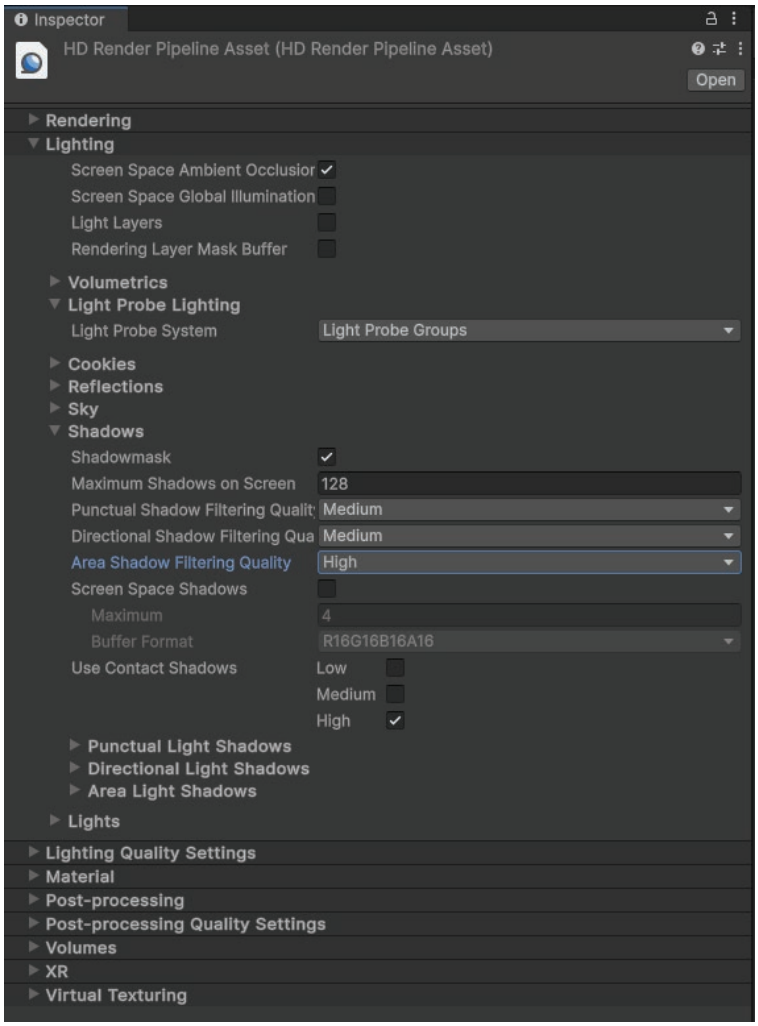
Micro Shadows によって、草葉の重なりのコントラストが強められています。

エリアライトソフトシャドウ

HDRP アセットの **Area Shadow Filtering Quality** を High に設定すると、エリアライトつきのソフトシャドウが改善されます。



エリアライトによって、ソフトシャドウが改善されています。



Area Shadow Filtering Quality

リフレクション

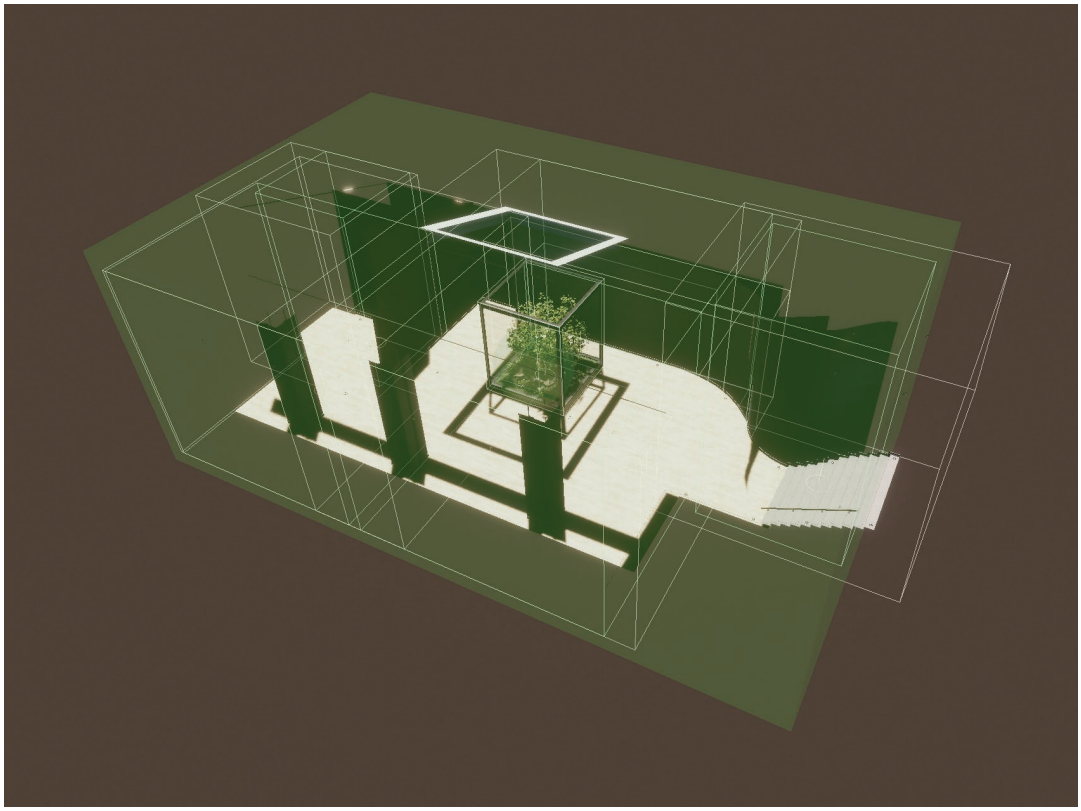
リフレクションは、ゲームオブジェクトと周囲の環境を統合するのに役立ちます。通常、リフレクションは、滑らかで光沢のあるサーフェスに関連付けますが、滑らかではないマテリアルの場合でも、PBR ワークフローで適切なリフレクションを受ける必要があります。HDRP では、次のようなさまざまな手法でリフレクションを生成できます。

— スクリーンスペースリフレクション

— リフレクションプローブ

— スカイリフレクション

各リフレクションタイプは、リソース消費が多くなる場合があるため、ユースケースに合わせて最適な手法を選択してください。複数のリフレクション手法がピクセルに適用される場合、各リフレクションタイプの効果がブレンドされます。Influence Volume という名前のバウンディングサーフェスを使用して 3D スペースを分割し、リフレクションが適用されるオブジェクトを指定してください。

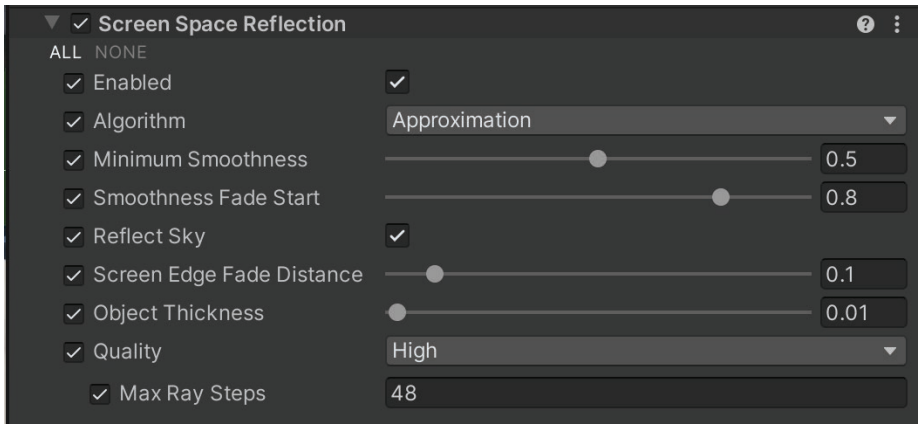


Influence Volume では、リフレクションプローブによってリフレクションを生成する場所を指定します。

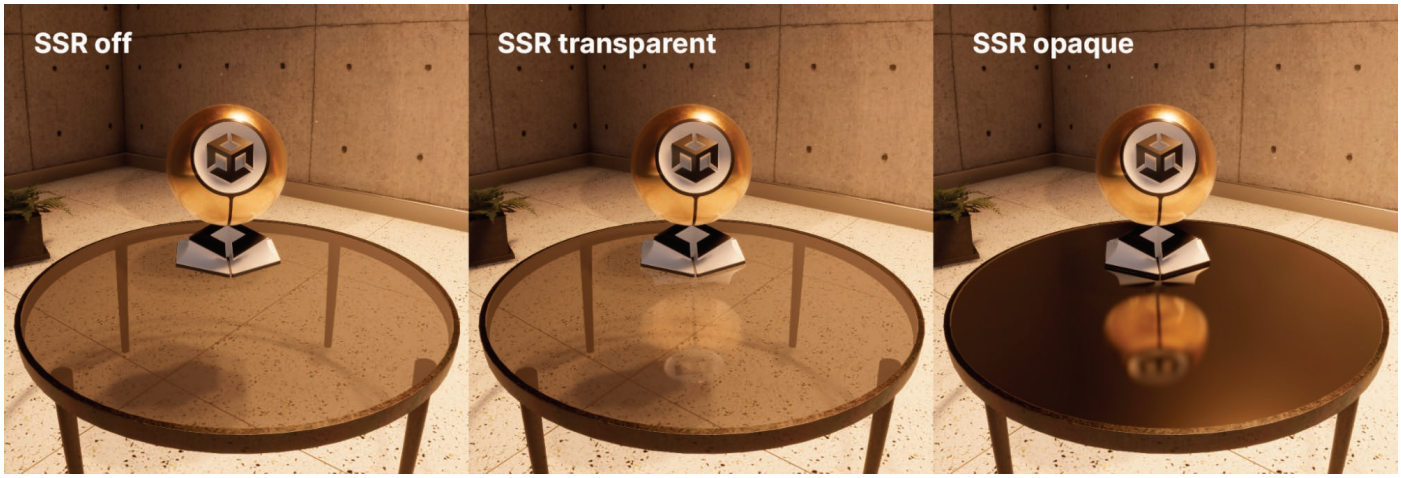
スクリーンスペースリフレクション

スクリーンスペースリフレクション（SSR）では、深度とカラーバッファを使用してリフレクションを計算します。そのため、反射するのはカメラビュー内のオブジェクトのみになり、画面上の位置によっては適切にレンダリングされない場合があります。光沢のある床や濡れた平面サーフェスが、スクリーンスペースリフレクションを適用する対象として適しています。

スクリーンスペースリフレクションでは、フレーム外のオブジェクトはすべて無視されるため、それが効果の制限となる場合があります。



スクリーンスペースリフレクションのオーバーライド



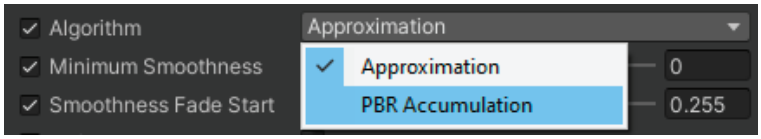
スクリーンスペースリフレクションは、透明および不透明のマテリアルに使用できます。

Lighting の **Frame Settings** (**HDRP Default Settings** またはカメラの **Custom Frame Settings**) で、**Screen Space Reflection** (スクリーンスペースリフレクション) 設定が有効になっていることを確認してください。続いて、Volume オブジェクトに **スクリーンスペースリフレクション** のオーバーライドを設定します。

SSR を表示するには、マテリアル表面の Smoothness (滑らかさ) が **Minimum Smoothness** の値を超えている必要があります。比較的粗いマテリアルに SSR を適用したい場合は、この値を引き下げます。ただし、Minimum Smoothness のしきい値を下げると、計算コストが大きくなる場合があります。スクリーンスペースリフレクションのオーバーライドの効果をピクセルに適用できなかった場合、HDRP はリフレクションプローブを使用する方法にフォールバックします。

Quality ドロップダウンを使用して、既定数の **Max Ray Steps** を選択します。Max Ray Steps が多くなると、クオリティーは上がりますが、コストは大きくなります。すべての効果と同様、パフォーマンスとビジュアルクオリティーのバランスをとることが大事です。

注意: 蓄積を使用した物理ベースのアルゴリズムか、精度の低い近似アルゴリズム (デフォルト) のいずれかを選択できます。



SSR を表示するには、マテリアル表面の Smoothness (滑らかさ) が Minimum Smoothness の値を超えている必要があります。

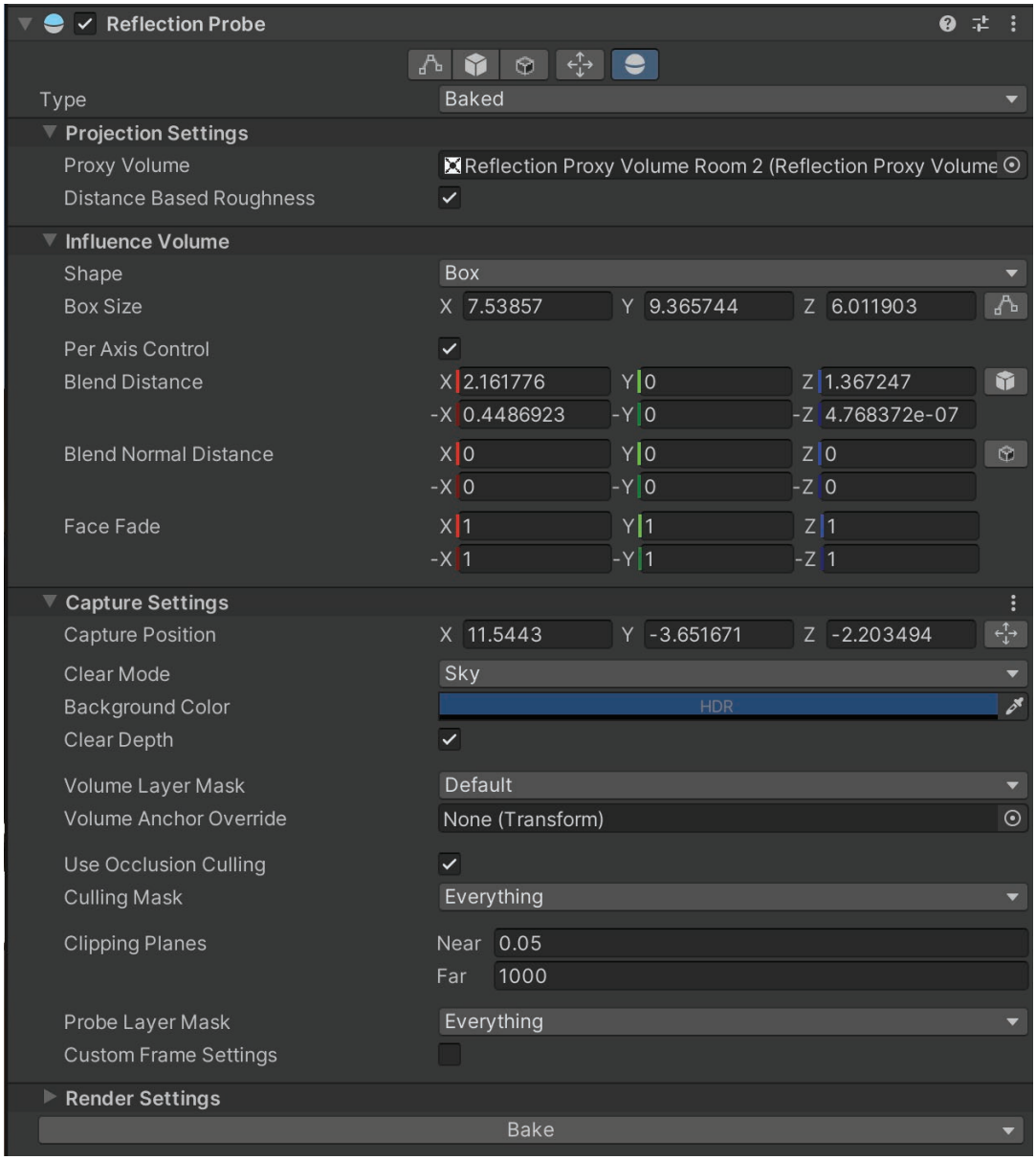
リフレクションプローブ

リフレクションプローブを使用すると、画像ベースの手法でリフレクションが生成されます。プローブは、周囲全方向の球面ビューを取り込み、その結果をキューブマップのテクスチャーに保管します。シェーダーは、そのキューブマップを使用してリフレクションを再現します。

各シーンに複数のプローブを適用し、結果をブレンドできます。以降、ローカライズされたリフレクションが環境内のカメラの動きに合わせて変化します。
各プローブの **Type** を **Baked** または **Real-time** に設定します。

— Baked のプローブは、静的環境でキューブマップのテクスチャを 1 回だけ処理します。

— Real-time のプローブは、ランタイムにエディターではなくプレイヤー内にキューブマップを作成します。そのためリフレクションが静的オブジェクトに限定されなくなりますが、リアルタイム更新によってリソース消費が多くなる場合があることに注意してください。



Reflection Probe コンポーネント

最適化のヒント

リアルタイムのリフレクションプローブを最適化するには、リフレクションの見栄えに大きく影響しないレンダリング機能を無効にします。これを行うには、カメラの設定をグローバルまたはプローブごとに **オーバーライド** します。更新をタイムスライスするスクリプトを作成しても良いでしょう。

Time Slicing プロパティは、Unity 2022 で Reflection Probe (リフレクションプローブ) コンポーネントに追加されました。このプロパティを表示するには、リフレクションプローブの **Type** を **Realtime** に設定します。この機能は、キューブマップ全体を一度に更新するのではなく、フレームごとに　1　つの面を更新するため、処理負荷が分散され、効率が向上します。その結果、パフォーマンスコストを抑えられます。

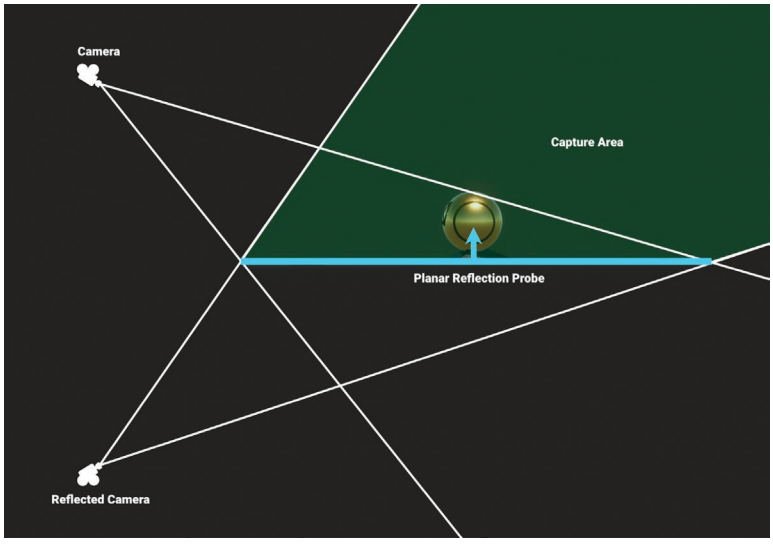
HDRP では、従来のキューブリフレクションプローブキャッシュ配列が、八面体投影の **2D テクスチャアトラスキャッシュ**　に置き換えられています。これにより、それぞれのリフレクションプローブの解像度を制御してメモリを節約することができます。

Influence Volume (インフルエンスボリューム) によって、ゲームオブジェクトにリフレクションが適用される 3D の境界を指定します。一方 **Capture Settings** では、リフレクションプローブによってキューブマップのスナップショットを取得する方法をカスタマイズできます。

Planar Reflection Probe コンポーネント

Planar Reflection Probe (平面リフレクションプローブ) コンポーネントでは、表面の滑らかさを考慮して、平らな表面の反射を再現できます。鏡や光沢のある床を表現するのには最適です。

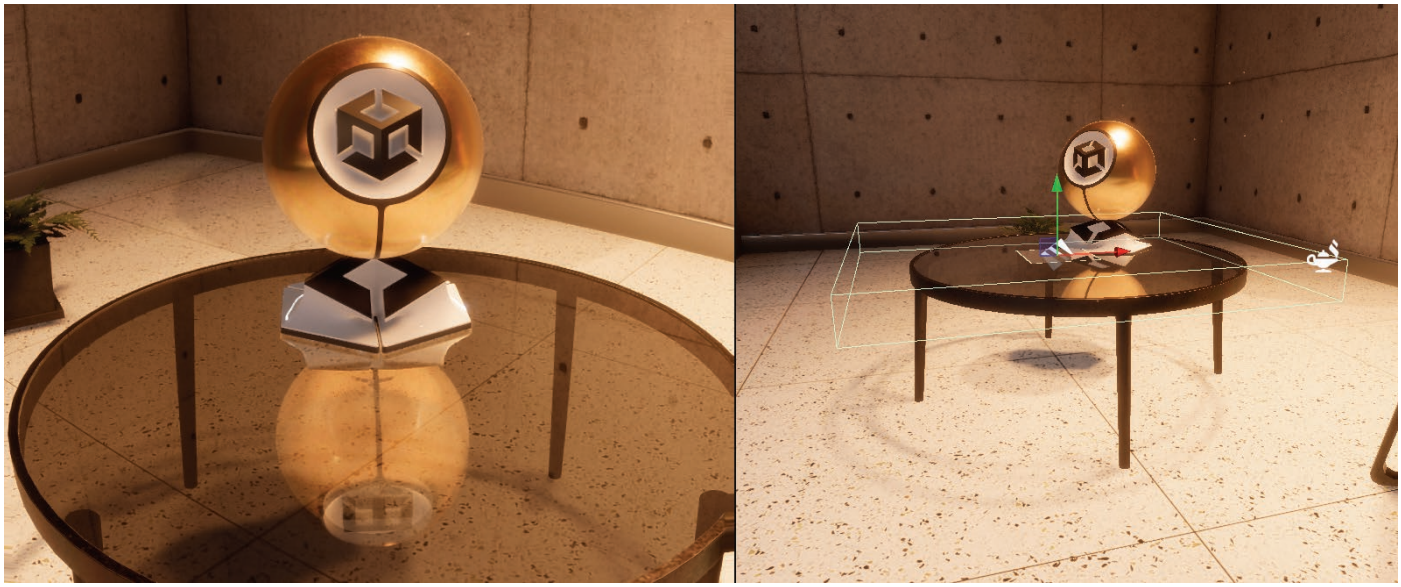
平面リフレクションプローブは標準的なリフレクションプローブと多くの共通点がありますが、2 種類のプローブの動作にはわずかな違いがあります。平面リフレクションプローブは、環境をキューブマップとして取り込むのではなく、プローブの鏡面で反射されたカメラのビューを再現します。



平面リフレクションプローブでは、平面にカメラを反射させることで、鏡面画像を取り込みます。

その後、生成された鏡像が 2D レンダーテクスチャとして保存されます。この画像が矩形のプローブの境界に描画され、平面リフレクションを生み出します。

平面リフレクションプローブは、平らなオブジェクトの反射を作成します。



スカイリフレクション

近傍のリフレクションプローブの影響を受けないオブジェクトは、空のリフレクションにフォールバックします。

リフレクションプローブは周囲の部屋を映し、空のリフレクションは Gradient Sky を反映します。

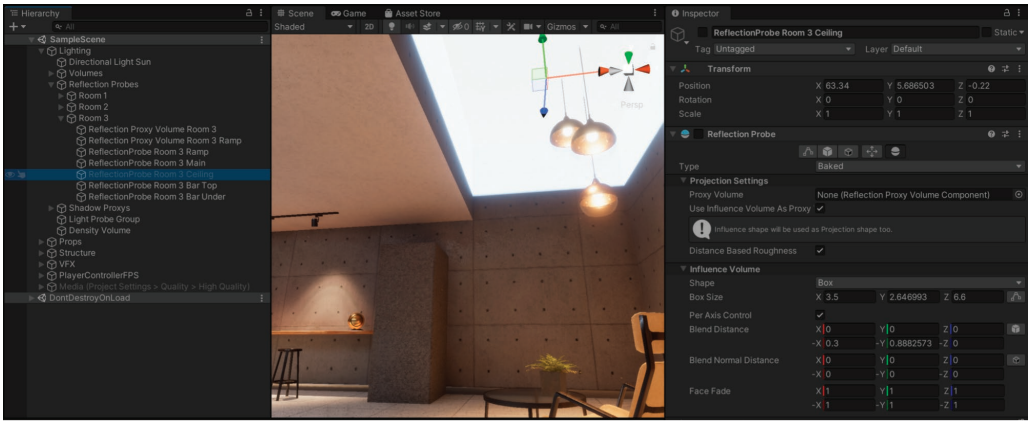
リフレクション階層

HDRP では質の高いリフレクションが得られるよう、各ピクセルの精度を最大限に高め、他の手法とブレンドすることができる手法を使用しています。HDRP では、加重優先度を基に、3 つのリフレクション方法 (SSR、リフレクションプローブ、空) をチェックします。リフレクションを評価する際のこのシーケンスは、**リフレクション階層**と呼ばれます。

1 つのピクセルで、ある手法によってリフレクションを特定できなかった場合は、その次の手法にフォールバックします。つまり、スクリーンスペースリフレクションはリフレクションプローブにフォールバックし、その後スカイリフレクションにフォールバックします。

リフレクションプローブに Influence Volume を適切に設定することが大事です。これを設定しないと、不適切なスカイリフレクションからライトが漏れる場合があります。

これは、SampleScene の Room 3 で顕著です。いずれかのリフレクションプローブを無効にするか、Influence Volume を変更すると、リフレクションが強制的に空にフォールバックされます。これにより、HDRI の明るい空が、強力なリフレクションによってシーンよりも過剰に強調されます。



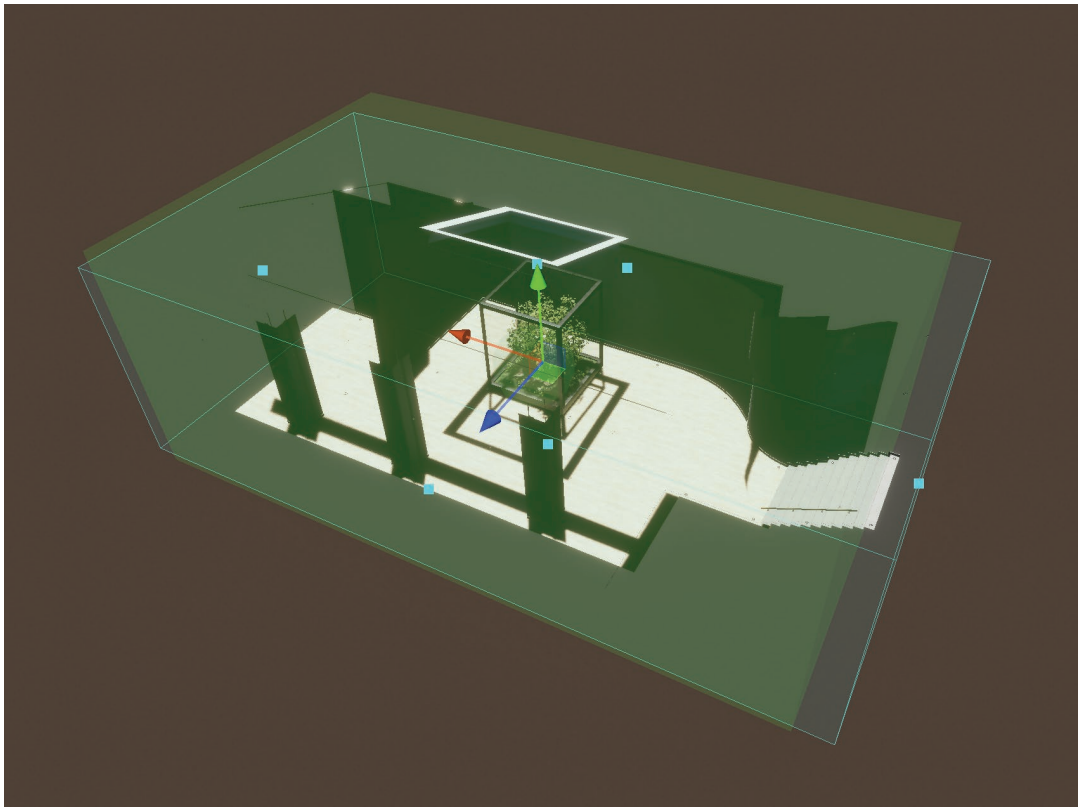
Room 3 の天井でリフレクションプローブを無効にすると、不要なライトリークが生じます。

リフレクション階層の決定に関する詳細は、HDRP のリフレクション に関するドキュメントページを参照してください。

Proxy Volume

リフレクションプローブのキャプチャポイントは固定されており、リフレクションプローブ近くのカメラ位置と一致することはほとんどないため、生成されたリフレクションで透視のシフトが目立つ場合があります。それにより、リフレクションが環境になじんでいないように見える場合があります。

Proxy Volume（プロキシボリューム）は、これを部分的に修正するのに役立ちます。カメラ位置を基に、Proxy Volume 内でより正確にリフレクションを再投影します。



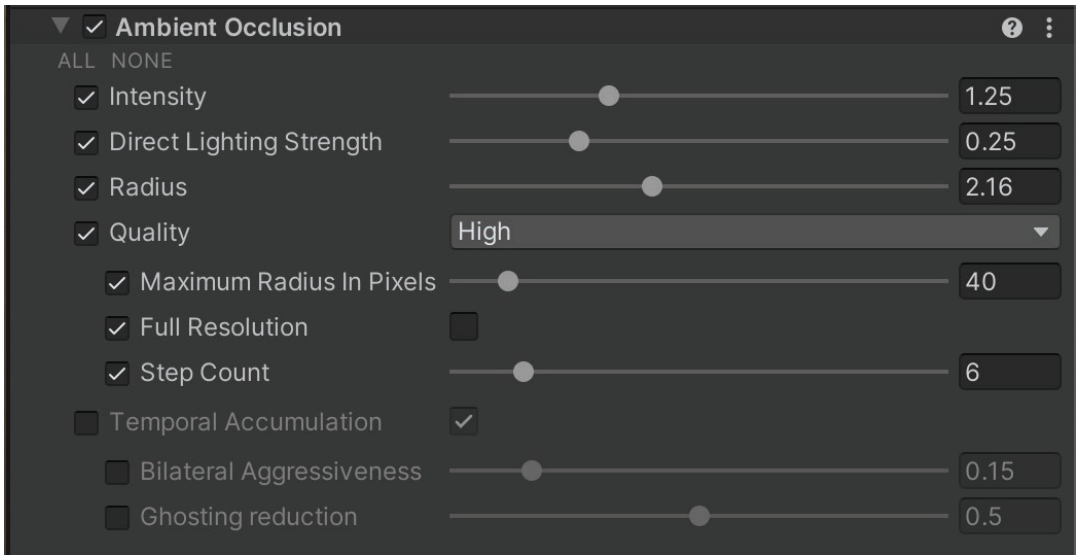
Proxy Volume では、ワールド空間での部屋の位置に合わせてキューブマップを再投影します。

リアルタイム ライティングエフェクト

HDRP は、ボリュームシステムを通じて、いくつかのリアルタイムのライティング効果を利用することもできます。ローカルまたはグローバルのボリュームを選択してから、**Add Override > Lighting で適切なエフェクトを追加します。**

スクリーンスペースアンビエントオクルージョン

アンビエントオクルージョンでは、近接した溝や穴、サーフェスの影のシミュレーションを行います。アンビエントライトが遮断されるエリアは、陰影が付いているように見えます。



アンビエントオクルージョンのオーバーライド

Sponza Atrium における Screen Space Ambient Occlusion の可視化

Unity のライトマッパーを使用して静的ジオメトリのアンビエントオクルージョンをベイクできますが、HDRP では、リアルタイム処理を行う **Screen Space Ambient Occlusion** (スクリーンスペースアンビエントオクルージョン) の オーバーライド も利用できます。これはスクリーンスペースの効果なので、フレーム内の情報のみが生成される効果に影響します。SSAO では、カメラの有効視野外のオブジェクトはすべて無視されます。

Frame Settings の **Lighting** で、**Screen Space Ambient Occlusion** を有効にします。続いて、ローカルまたはグローバルのボリュームで **Add Override** をクリックし、**Lighting > Ambient Occlusion** を選択します。

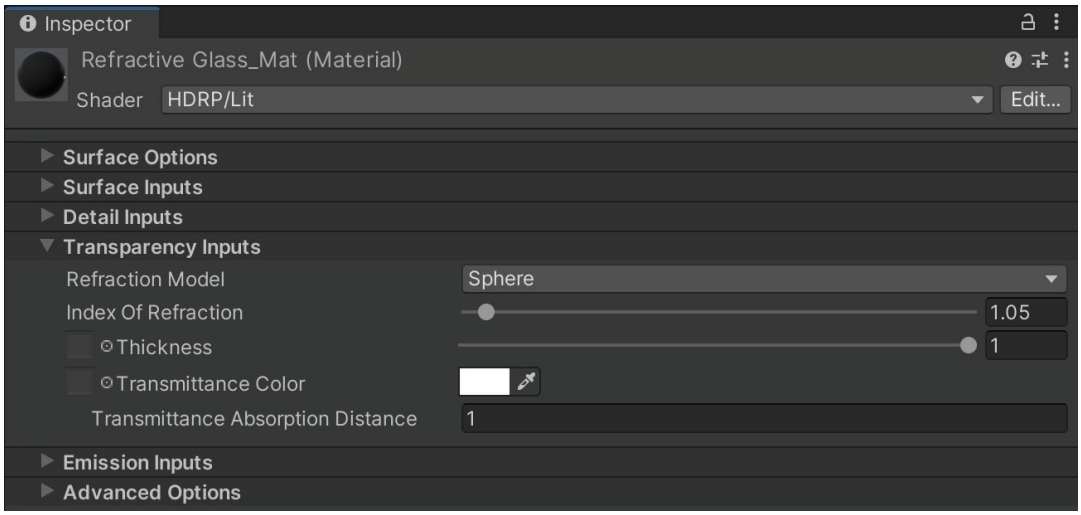
スクリーンスペース屈折

Screen Space Refraction (スクリーンスペース屈折) オーバーライドは、大気よりも密度が高い媒体を通過する光の動きのシミュレーションを行う際に役立ちます。HDRP のスクリーンスペース屈折では、深度とカラーバッファを使用して、ガラスなどの透明なマテリアルを透過する光の屈折を計算できます。

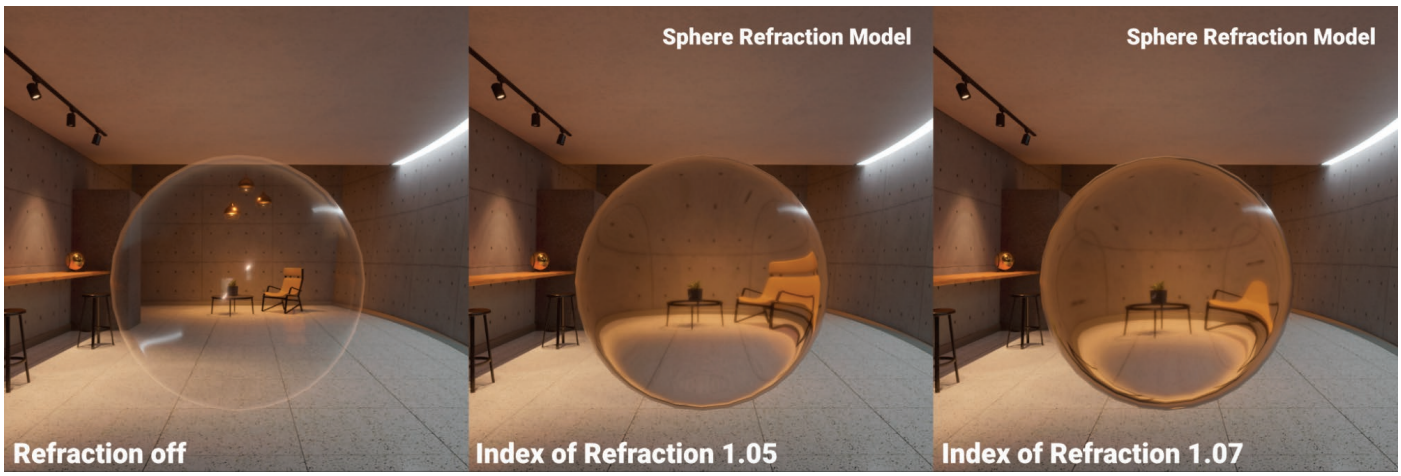
Screen Space Refraction のオーバーライド

HDRP/Lit シェーダーでこの効果を有効にするには、マテリアルの Surface Type が **Transparent** になっていることを確認します。

続いて、**Transparency Inputs** で **Refraction Model** と **Index of Refraction** を選択します。中空でないオブジェクトの場合、Refraction Model には **Sphere** を使用します。中空のオブジェクトについては **Thin** (泡など) または **Box** (わずかに厚みがあるもの) を選択します。



屈折を制御する Transparency Inputs



スクリーンスペース屈折

ライトレンダリングレイヤ

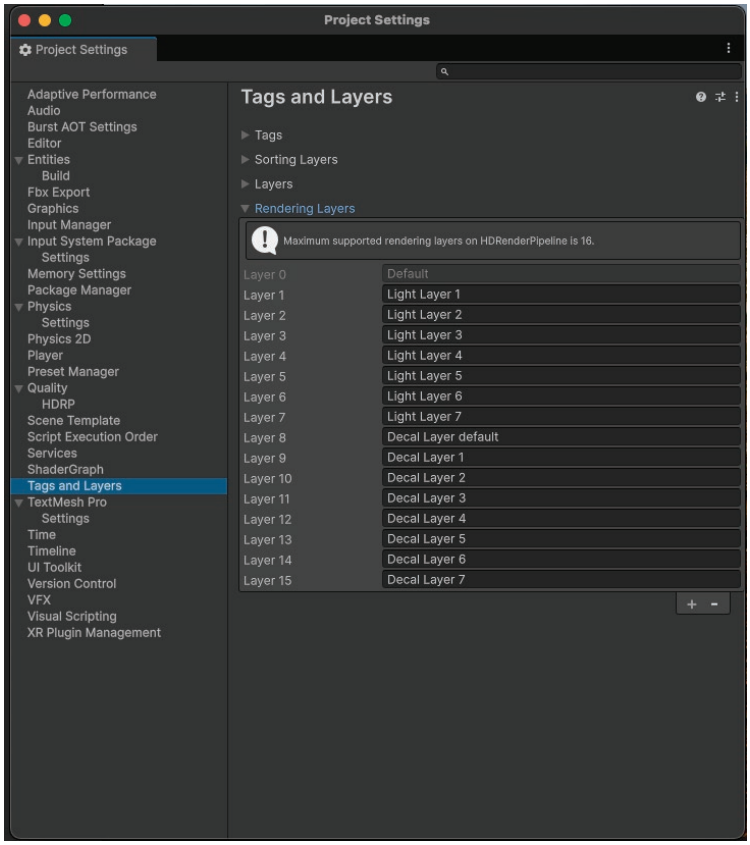
HDRP のレンダリングレイヤーを使用して、シーン内の特定のレンダラーに影響を与えるライト、デカール、効果を制御できます。レンダリングレイヤーは、特定のレンダラーにのみ影響するようにライトや効果を制限する LayersMasks です。

32 のうち 16 のレンダリングレイヤー

どのレンダラー (メッシュまたはオブジェクト) も最大 32 のレンダリングレイヤーに属することができますが、HDRP のライトとエフェクトが対応しているのは最初の 16 のレイヤーのみです。レンダラーは、スクリプトや整理する目的で 32 のレイヤーのどれに割り当てることもできますが、ライティングやデカールなどの HDRP のエフェクトに影響を与えるのは最初の 16 レイヤーだけです。

レンダリングレイヤーの設定

レンダリングレイヤーを作成して名前を付けるには、**Edit > Project Settings > Tags and Layers** に移動します。**Rendering Layers** を開き、管理しやすいようレイヤーにカスタム名を割り当てます。



Rendering Layers 機能を有効にします。

ライトのレンダリングレイヤーを有効にするには、**Edit > Project Settings > Graphics > Pipeline Specific Settings > HDRP** に移動します。**Frame Settings (Default Values)** の Lighting で Light Layers を有効にします。

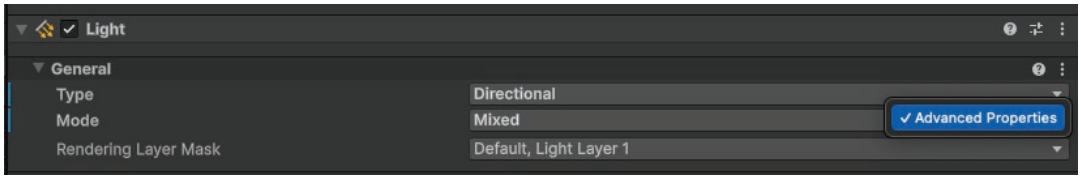
次に、**HDRP Asset** で **Light Layers** を有効にします。

レンダリングレイヤーをデカールで使用するには、**デカールレイヤー** のドキュメントを参照してください。

ライトでのレンダリングレイヤーの使用

Light Layers を有効にしたら、ライトと Mesh Renderer に割り当てることができます。

Inspector の Light にある **General** セクションのオプションメニュー (⋮) で **Advanced Properties** を有効にします。



Advanced Properties でレンダリングレイヤーを表示します。

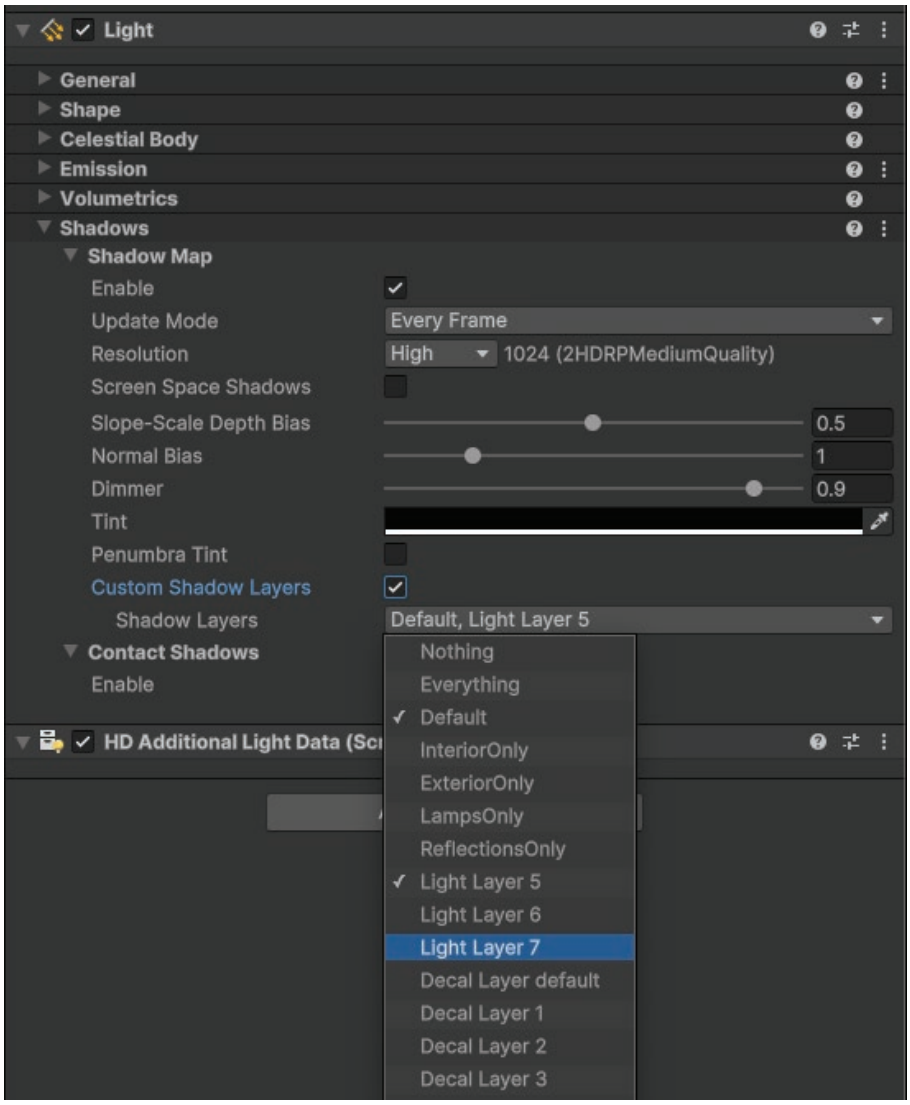
Rendering Layer Mask ドロップダウンで、ライトが影響する **Light Layers** を指定します。

Mesh Renderer や Terrain (地形) を割り当てるには、**Inspector** の **Rendering Layer Mask** ドロップダウンを使用します。ライトは、Mesh Renderer や Terrain が同じレンダリングレイヤーを使用している場合にのみ影響します。

シャドウでのレンダリングレイヤーの使用

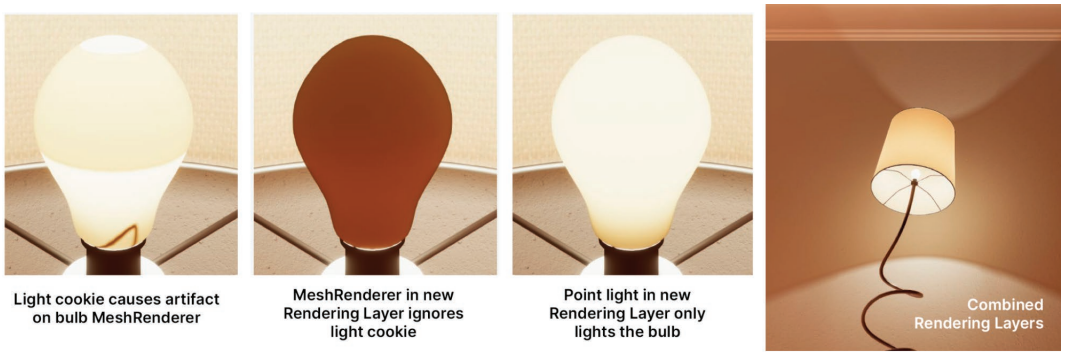
HDRP のデフォルトの動作では、ライトのレンダリングレイヤーを、そのライトのシャドウレンダリングに連動させます。つまり、ライトが照らすものはすべて、その同じレイヤーに基づいて影を落とすことになります。

シャドウをライティングから分離する必要がある場合は、Light の **Shadows** 設定で **Custom Shadow Layers** を有効にします。この機能を使用すると、シャドウ専用の別のレンダリングレイヤーを割り当てることができます。



レンダリングレイヤーを使用してライトの効果を分離します。

この例は、シャドウのレンダリングレイヤーの実際の使用方法を示しています。ここでは、ライトクッキーにより電球に不要なセルフシャドウイングが発生しています (左)。電球の Mesh Renderer を別のレンダリングレイヤーに割り当てることで、ライトとライトクッキーが電球に影響を与えることがなくなりました (中)。その後、電球と同じレンダリングレイヤーに別のポイントライトが割り当てられ、意図したライティングになっています (右)。



レンダリングレイヤーを使用してライティングとシャドウをカスタマイズします。

詳細については、この Unity における高品質な照明器具 についてのエキスパートガイドを参照してください。

レイトレーシングとパストレーシング

レイトレーシング は、従来のラスタライゼーションよりも訴求力のあるレンダリングを生成できる手法です。従来は計算コストが大きい処理でしたが、近年のハードウェアアクセラレーションの発達により、レイトレーシングのリアルタイム利用が可能になりました。

HDRP は、レイトレーシングとラスタライズレンダリングをブレンドしたハイブリッドレイトレーシングシステムを使用します。DirectX 12 をサポートしており、互換性のある NVIDIA RTX™ や AMD Radeon™ PRO GPU が必要です。

システム要件の一覧については、Getting started with ray tracing を参照してください。

Unity 6 の新機能

- **製品版レイトレーシング:**HDRP は DirectX 12 と特定のコンソールプラットフォームのレイトレーシングをサポートしています。詳細については、コンソール固有のドキュメントを参照してください。
- **強化されたレイトレーシング効果:**レイトレーシングによるアンビエントオクルージョン、グローバルイルミネーション、リフレクション、シャドウの改善により、より正確でリアルなライティングとシェーディングを実現しています。
- **立体角カリング**:立体角カリングを有効にして、小さく遠くにあるインスタンスを加速構造から除外できるようになりました。これは、HDRP Ray Tracing Volume 設定 と、EnableSolidAngleCulling によるスクリプティング API によって有効化できます。立体角カリングを有効にすると、レイトレーシングのパフォーマンスを簡単に向上させることができます。
- **インラインレイトレーシングとシェーダークエリ:**インラインレイトレーシングとシェーダーを介して、ハードウェアアクセラレーションによるレイクエリを活用できるようになりました。これにより、カスタムレイトレーシングシャドウなど、カスタムエフェクトオーサリングの新しい可能性が広がります。

— **デカールのサポート:**デカールが、レイトレーシングによるリフレクションとグローバルイルミネーションに完全に統合されました。例えば、大きな黄色のデカールは、周囲の表面を温かみのある反射光で染め、よりリアルなレンダリングを実現します。

Decal Projector を HDRP Path Tracer と併用できるようになり、水たまり、汚れ、弾痕などのマテリアルエフェクトを、下敷きとなる表面を変更することなく追加できるようになりました。

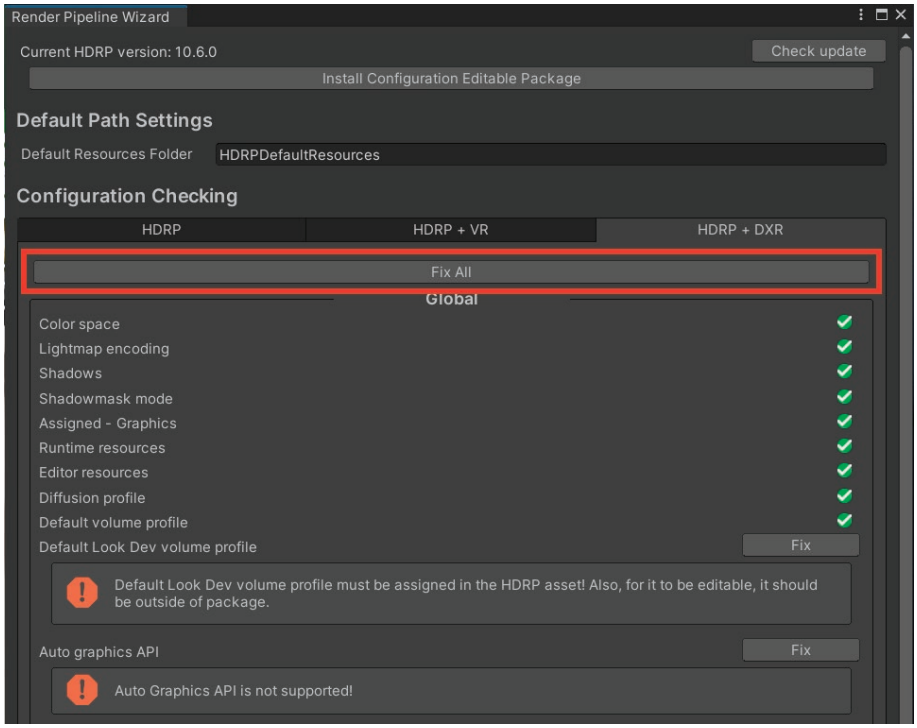
— **最適化された加速構造:**レイトレーシングでは、複雑なシーンのメモリ使用量が減り、リアルタイムパフォーマンスが向上しています。これは、新しい小さな BLAS アロケーター (小さなメッシュに対するメモリの無駄を削減)、BLAS 圧縮 (静的メッシュのメモリ使用量を軽減)、新しい メモリ最小化フラグなど、複数の最適化の結果です。

— **拡張されたハードウェアサポート:**完全なレイトレーシングハードウェアアクセラレーションは、NVIDIA GeForce RTX GPU、NVIDIA RTX PRO GPU、一部の AMD Radeon™ RX と PRO Series GPU、Windows PC 用の DirectX 12 と DXR 対応 GPU、および特定のコンソールプラットフォームで利用できます (詳細については、コンソール固有のドキュメントを参照してください)。

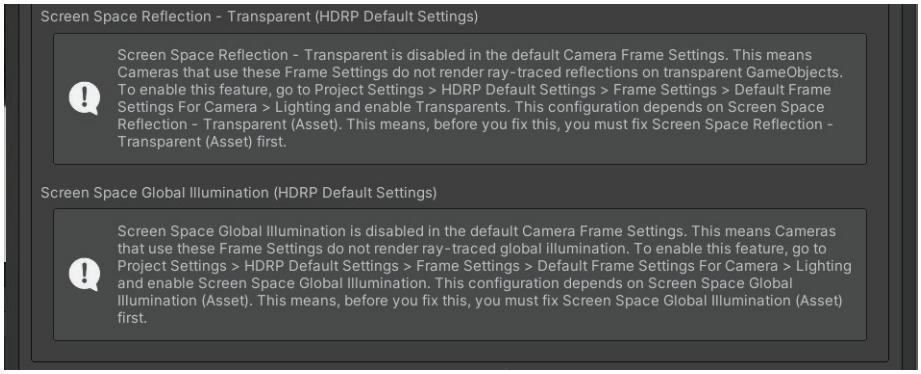
設定

レイトレーシングを有効にするには、HDRP プロジェクトのデフォルトのグラフィックス API を DirectX 12 に変更する必要があります。

Render Pipeline Wizard (レンダーパイプラインウィザード) を **Window > Rendering > HDRP Wizard** で開きます。**HDRP + DXR** タブの **Fix All** をクリックすると、エディターの再起動を促されます。



Render Pipeline Wizard でレイトレーシングを有効にします。



指示に従って無効になっている機能を修正します。

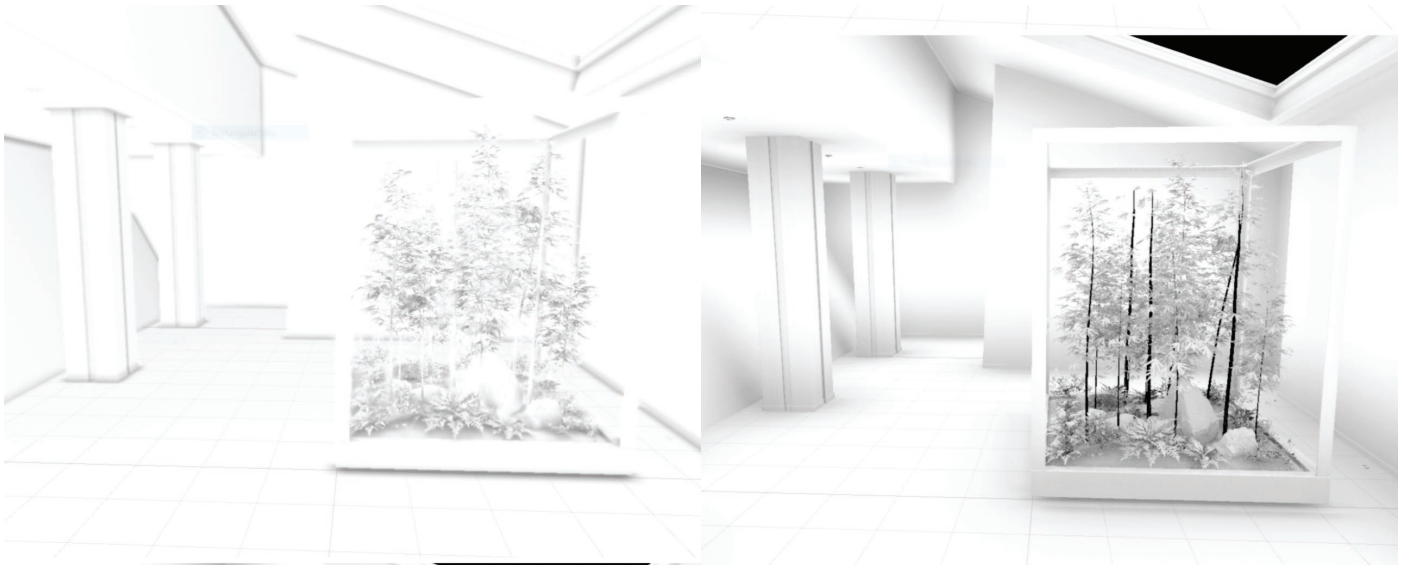
レイトレーシングを 手動で 設定することもできます。

プロジェクトでレイトレーシングを有効にしたら、**HDRP Global** または **Camera Frame Settings** でもレイトレーシングが有効になっていることを確認します。**Build Settings** で、対応している 64 ビットアーキテクチャを使用していることを確認します。また、**Edit > Rendering > Check Scene Content for HDRP Ray Tracing** から、シーンのオブジェクトを検証します。

オーバーライド

レイトレーシングの導入により、新しい Volume オーバーライドがいくつか追加され、HDRP の既存のオーバーライドが多数強化されています。

— **レイトレーシングによるアンビエントオクルージョン:**スクリーンスペースアンビエントオクルージョン (SSAO) に代わって、レイトレーシングによるアンビエントオクルージョンが導入されます (下の画像比較参照)。SSAO と異なり、レイトレーシングによるアンビエントオクルージョンでは、画面外のジオメトリを使用してオクルージョンを生成できます。これにより、フレームの端に向かうにつれて効果が消えたり、不正確になったりする問題が解消されます。



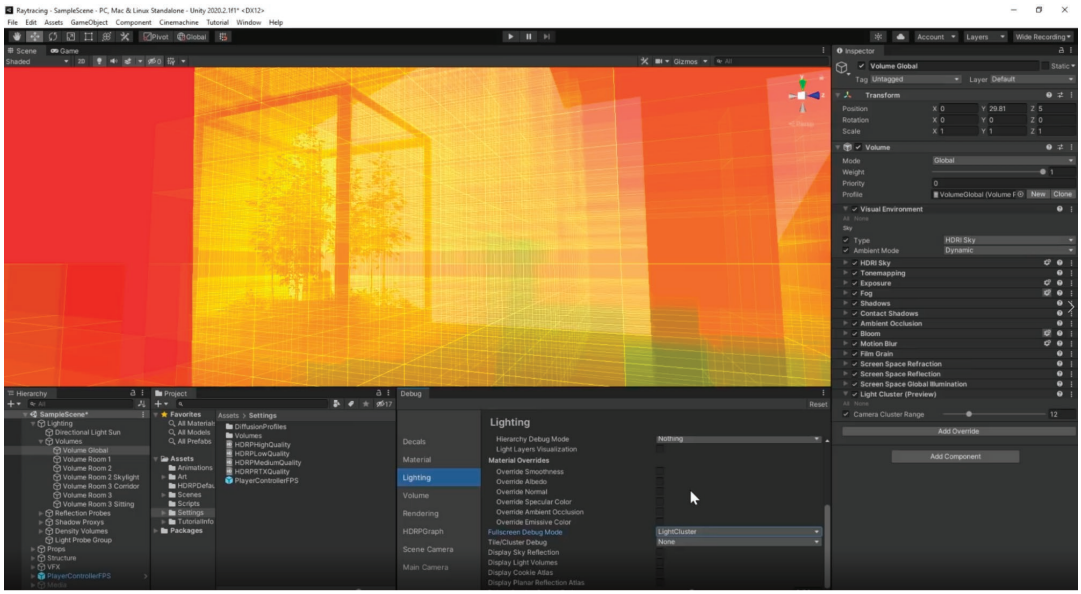
スクリーンスペースアンビエントオクルージョン (左) とレイトレーシングによるアンビエントオクルージョン (右) の比較

— **ライトクラスター:**レイトレーシングによるリフレクション、GI、SSS、および再帰的なレンダリングを使用する場合、アーティファクトを回避してパフォーマンスを最適化するために、ライトがライトクラスターに効率的に格納されるようにする必要があります。HDRP は、シーンをカメラ中心の座標軸に沿ったグリッドに分割します。

HDRP はこの構造を使い、光線がサーフェスに当たるたびに、ライティングに寄与する可能性のあるローカルライトのセットを決定します。その後、特定のエフェクトの反射光を計算します (レイトレーシングによるリフレクション、レイトレーシングによるグローバルイルミネーションなど)。

Camera Cluster Range の Volume オーバーライドを使用してこの構造の範囲を変更し、考慮する必要があるゲームオブジェクトやライトが確実に含まれるようにします。

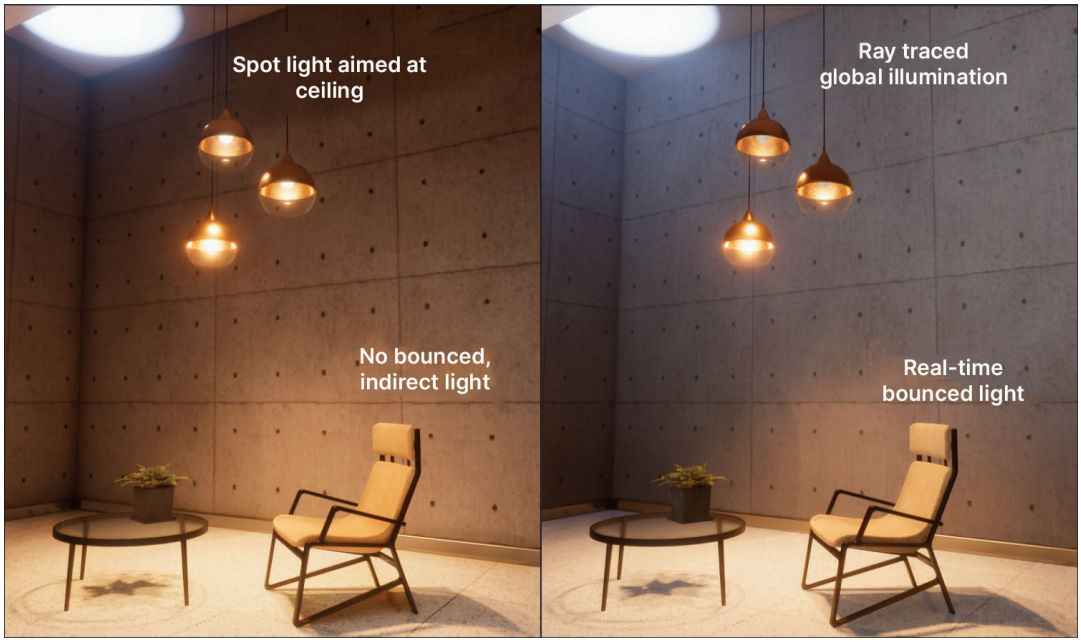
HDRP Debug モードは、**Windows > Analysis > Rendering Debugger > Lighting > Full Screen Debug mode** から使用できます。ライト数が HDRP アセットの **Maximum Lights per Cell** に達しているライトクラスターセルは赤でハイライトされるため、可視化に役立ちます。この設定を調整することで、不要なライトの漏れやアーティファクトを減らすことができます。



デバッグモードでのレイトレーシングライトクラスター

— **レイトレーシングによるグローバルイルミネーション:**これは、バウンスされた間接光をシミュレートする際、SSGI や ライトプローブの代わりに使えるものです。レイトレーシングによるグローバルイルミネーションはリアルタイムで計算され、ライトマップをベイクする長いオフライン処理を回避しつつも同等の結果を得ることができます。

複数のバウンスやサンプルが役に立つ複雑な室内環境には、Quality 設定を使用します。Performance モード (1 つのサンプルと 1 つの反射に限定) は、ライティングのほとんどがメインのディレクショナルライトによって行われる屋外の状況に適しています。

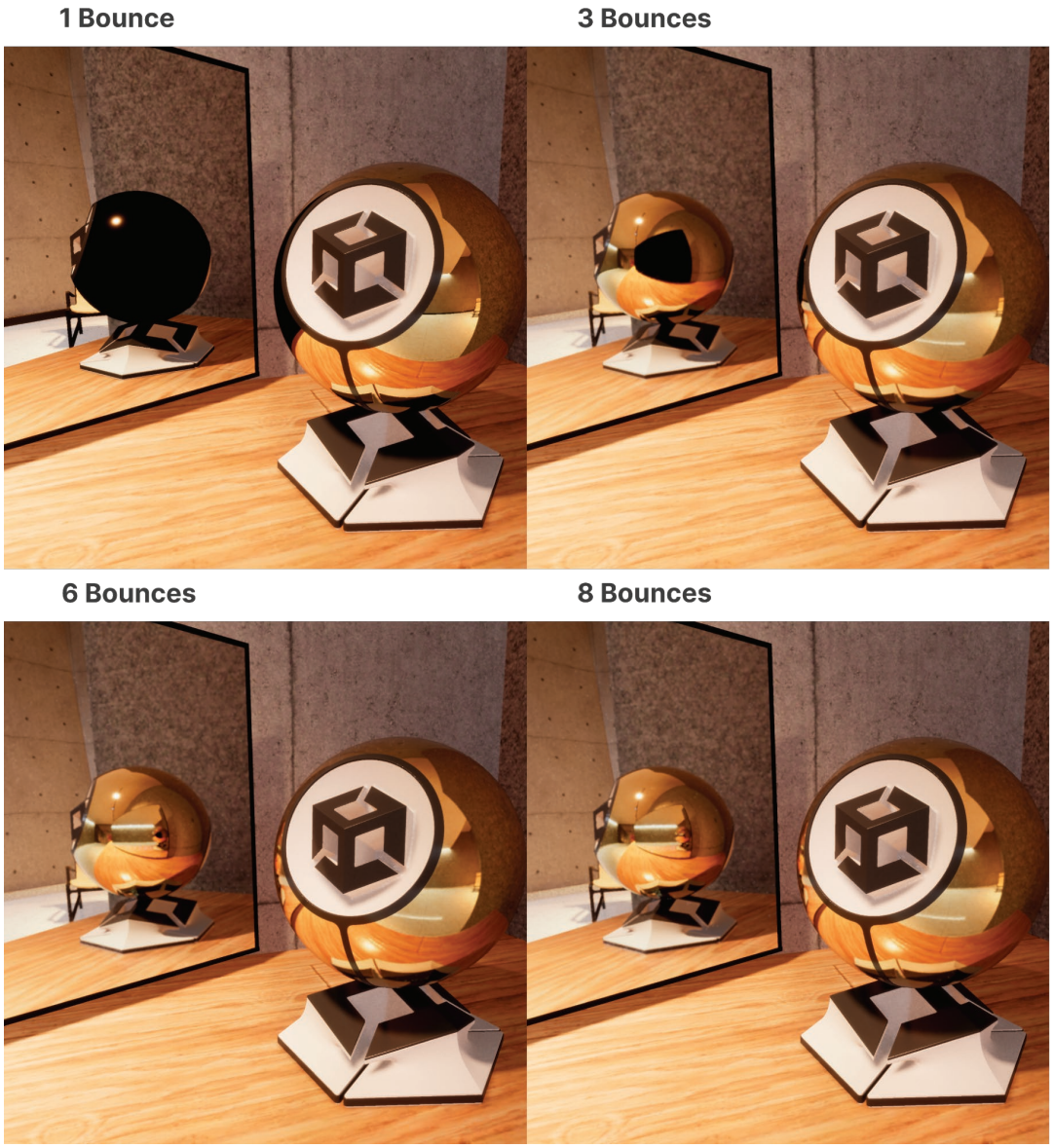


レイトレーシングによるグローバルイルミネーションによって、反射光がリアルタイムで表示されます。

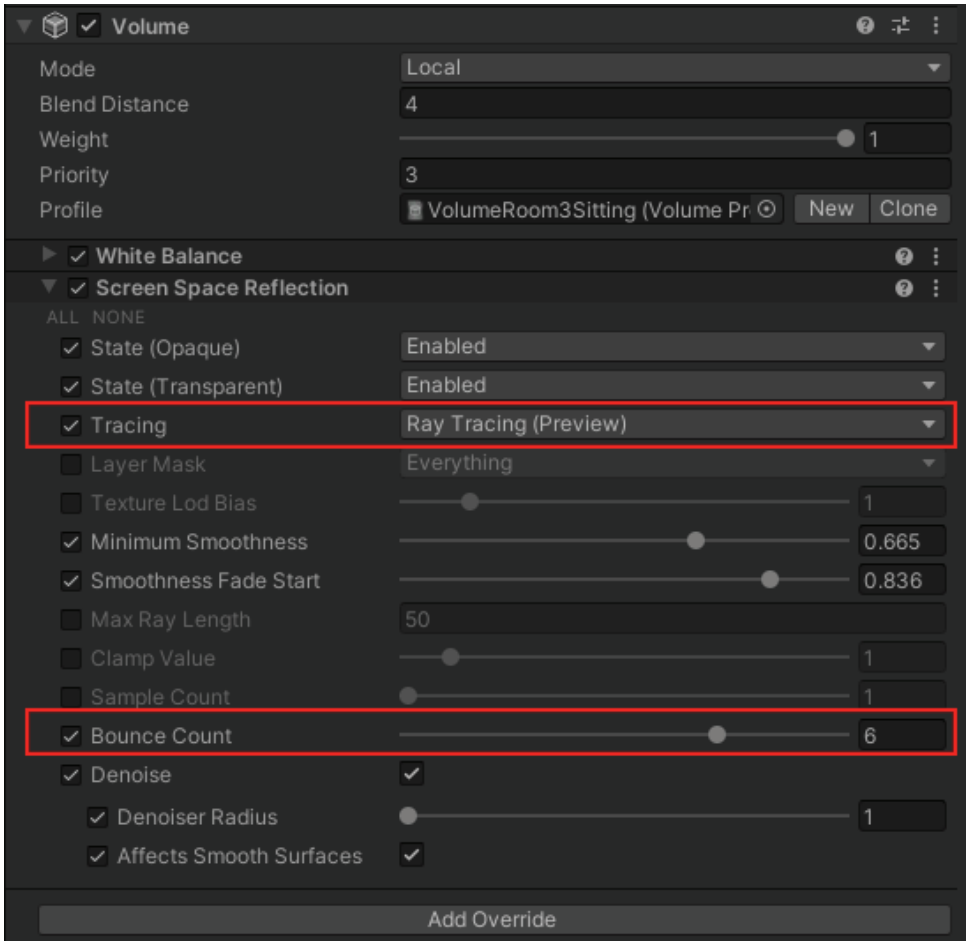
— **レイトレーシングによるリフレクション:** レイトレーシングによるリフレクションでは、リフレクションプローブやスクリーンスペースリフレクションよりも質の高いリフレクションを再現できます。画面外のメッシュは、生成されたリフレクションで適切に表示されます。

Minimum Smoothness と **Smoothness Fade Start** の値を調整すると、滑らかなサーフェスがレイトレーシングによるリフレクションを受け始めるしきい値を変更できます。必要に応じて **Bounces** を増やすこともできますが、パフォーマンスコストに注意してください。

以下の例は、合わせ鏡などの “無限の鏡“の 設定における、レイトレースバウンスの効果を示しています。この数枚の画像からは、1 回、3 回、6 回、8 回とバウンスの回数が増えるにつれて複雑になっていくリフレクションの進行を確認できます。バウンスの回数が増えるにつれてリフレクションが遠ざかっていくように見え、奥行き感が生まれています。



レイトレーシングによるリフレクションは、画面外のオブジェクトを含むこともあります。



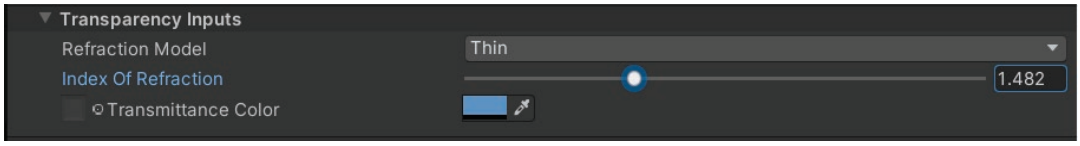
レイトレーシングは、スクリーンスペースリフレクションを改善することができます。

— **レイトレーシングによるシャドウ：**ディレクショナル、ポイント、スポット、矩形エリアの各ライトのレイトレーシングによるシャドウを、不透明なゲームオブジェクトのシャドウマップの代わりに使用することができます。ディレクショナルライトは、透明または半透明のゲームオブジェクトにレイトレーシングによるシャドウを落とすこともできます。

レイトレーシングは、現実世界の影と同様、キャスター (影をおとす物体) からの距離が遠くなるにつれて柔らかくなる自然なシャドウを作り出すことができます。

レイトレーシングされたシャドウはキャスターから離れるにつれてぼやけるため、シャドウマッピングとは異なる効果が得られます。ディレクショナルライト、ポイントライト、スポットライトも半透明のシャドウを生成できます。

HDRP のディレクショナルライトで、半透明の色付きシャドウを生成することも可能です。以下の画像では、ガラスのサーフェスによって、リアルな色合いのシャドウが床にキャストされています。



透明のシャドウキャスターには Transmittance Color を使用します。

ディレクショナルライトで、レイトレーシングによる色付きシャドウ (右) を生成できます。

HDRP のレイトレーシング機能について詳しくは、Activate ray tracing with HDRP / HDRP におけるレイトレーシングの有効化 (自動翻訳機能推奨) で確認してください。HDRP のマイクロサイトの レイトレーシングに関するドキュメント も参照してください。

インラインレイトレーシングのサポート (PC およびコンソール)

Unity レイトレーシング API が拡張され、Windows およびコンソールプラットフォームを対象にする場合、すべてのシェーダーステージでインラインレイトレーシングがサポートされるようになりました。

レイトレーシングパイプライン

HDRP のさまざまなレイトレーシングのエフェクト (シャドウやリフレクションなど) はすべて、Unity レイトレーシング API と従来の "レイトレーシングパイプライン" を使用して実装されています。

レイトレーシングパイプラインは、レイトレーシング加速構造　を作成することで機能します。これは、シーンのジオメトリを表し、GPU がレイの交点を効率的に判定するのに役立ちます。

レイがオブジェクトに当たると、"ヒットシェーダー" がシェーディングを計算します。レイが外れた場合、"ミスシェーダー"　が背景や環境を決定します。シェーダーテーブルはこれらのインタラクションを動的に管理し、それぞれの場合に適したシェーダーをバインドします。

このシステムは複雑なシーンには適していますが、処理のオーバーヘッドが増えるため、汎用のコンピュートシェーダーには適していません。シンプルなエフェクトに対して動的シェーダーテーブルを使用すると、不必要に複雑になることがあります。

インラインレイトレーシング

DXR 1.1 ではインラインレイトレーシングが導入され、従来のレイトレーシングパイプラインに依存せずに、コンピュートシェーダーとラスタライゼーションシェーダー内でハードウェアアクセラレーションによるレイクエリを実行できるようになりました。

これはハードウェアアクセラレーションによるレイトレーシングの代替手段となり、シェーダーとカスタムパイプラインエフェクトの作成において新しい可能性が広がります。

Unity 6 では、DXR 1.1 対応の Windows プラットフォームとコンソールを対象にした場合、この新機能がコンピュートシェーダーとラスタライゼーションシェーダーのステージ全体でレイトレーシング API によって完全にサポートされるようになりました。

シェーダーでのレイクエリの使用方法の詳細については、DXR1.1およびRayQueryオブジェクトのDirectX仕様を確認してください。また、インラインレイトレーシングサポート の詳細については、公式のディスカッションの投稿を参照してください。

インラインレイトレーシングは、これらのカスタムソフトシャドウを作成します。

パフォーマンス

レイトレーシングには、追加のパフォーマンスコストが必要になります。レイトレーシングが適用されたワールドの構築や更新を GPU 上で行うコストはワールドの複雑さと更新頻度に応じてスケールしますが、光線の送信と効果の適用にかかるコストは影響を受けるピクセルの数に比例してスケールします。

ここでは、見栄えとパフォーマンスのバランスを取るためのヒントをいくつか紹介します。

— **動的解像度:**レイトレーシングはピクセル数でスケールするため、低い解像度でレンダリングすればパフォーマンス負荷を軽減できます。最新のアップスケーラー (NVIDIA DLSS など) では、解像度を下げてレンダリングした後、視覚的忠実度の低下を最小限に抑えながら、高度な処理でフル解像度にアップスケールできます。

例えば、DLSS を有効にして、強制的に画面の解像度を 75% にできます。これによりフレーム時間が改善します。特に GPU 負荷の軽減により全体的なパフォーマンスが向上する CPU 依存のシナリオで有効です。

- **使用するエフェクトを絞る:**すべてのレイトレーシングの効果を常にアクティベートしておく必要はありません。ゲームアプリケーションのどこにデプロイするか選択できます。

  例えば、レーシングゲームでは、レイトレーシングによるリフレクションを加えることで、プレイヤーの車のレンダリングを改善できます。オープンワールドアドベンチャーゲームでは、レイトレーシングによるシャドウとグローバルイルミネーションが、より没入感の高い環境を構築するのに役立ちます。

- **必要な場合のみレイトレーシングを使用する:**レイトレーシングによるリフレクションやグローバルイルミネーションなど、複数のエフェクトを使用する場合はまず、Mixed トレーシングモード を試してください。このモードでは、最初に高速なレイマーチングを使用し、必要な場合にのみレイトレーシングにフォールバックします。

  コストのかかる長距離レイの計算をさらに軽減するには、Probe Volume、Sky Volume オーバーライド、リフレクションプローブなどの代替手法を使用します。これにより、すべてレイトレーシングを使用するより少ないコストで、効率的なライティングとリフレクションを実現できます。

- **評価を最適化する**:

  - 一部のエフェクトでは、Layer Mask を使用して、レイトレーシングの効果が不要なオブジェクトを評価しないようにします。
  - 品質設定を調整します。レイの長さ、サンプル数、解像度をカスタマイズして、パフォーマンスとノイズのバランスを取ります。レイを長くするとノイズを減らすためにより多くのサンプルが必要になります。サンプル数を減らすと GPU のコストは下がりますがノイズは増加します。
  - リフレクションについては、オブジェクトが多重反射を必要としない場合は、バウンス数を減らします。滑らかさのパラメーターを調整して、必要な箇所のみにレイトレーシングを適用します。
  - テクスチャを使用するレイトレーシングエフェクトの場合は、Texture Lod Bias を適用して低解像度の MIP レベルを取得します。これにより、GPU メモリ使用量を減らし、マテリアル評価時のテクスチャフェッチのコストを低減できます。その結果、見栄えに大きな影響を与えることなく、全体的なパフォーマンスを向上させることが可能になります。

- **一般的なレイトレーシング設定:**これらの設定からは、パフォーマンスと見栄えのバランス、レイの設定、加速構造の構築方法を調整することができます。

パフォーマンスを分析するには、再生モードで Rendering Debugger (**Window** > **Render Pipeline Debug**) を使用します。これにより、すべてのレイトレーシングエフェクトのコストを追跡するためのマーカーひと揃いが得られます。

CPU timings RT は、属性バインディングや C# の実行など、CPU でのレイトレーシングタスクの処理にかかる時間 (ミリ秒単位) を測定します。一方、**GPU timings RT** は、レイトレーシングされたエフェクトごとの GPU の実行時間を追跡します。

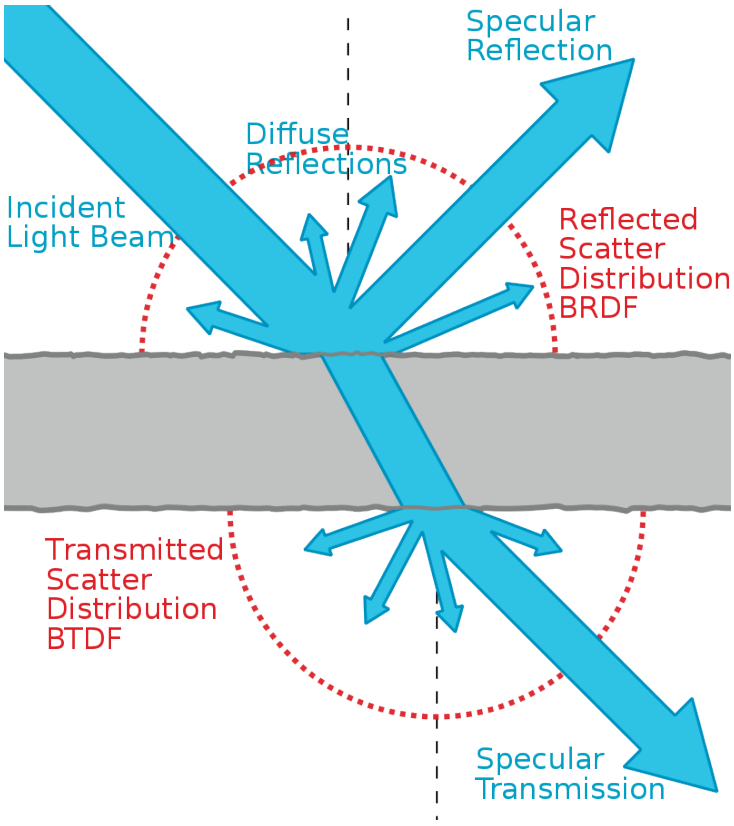
パストレーシング

パストレーシングはレイトレーシングの一種で、複数の表面での光線の反射を含め、各光線が辿り得る多くの経路のシミュレーションを行います。これにより、ライトとマテリアルの間の複雑な相互作用を捉えることができます。

パストレーシングも、レイトレーシングと同様に、カメラからの光線をシーンに照射することから始まります。

しかし、レイはサーフェスにぶつかった段階で止まるのではなく、様々なオブジェクトと相互作用しながら、シーン内で何度もバウンスし続けます。

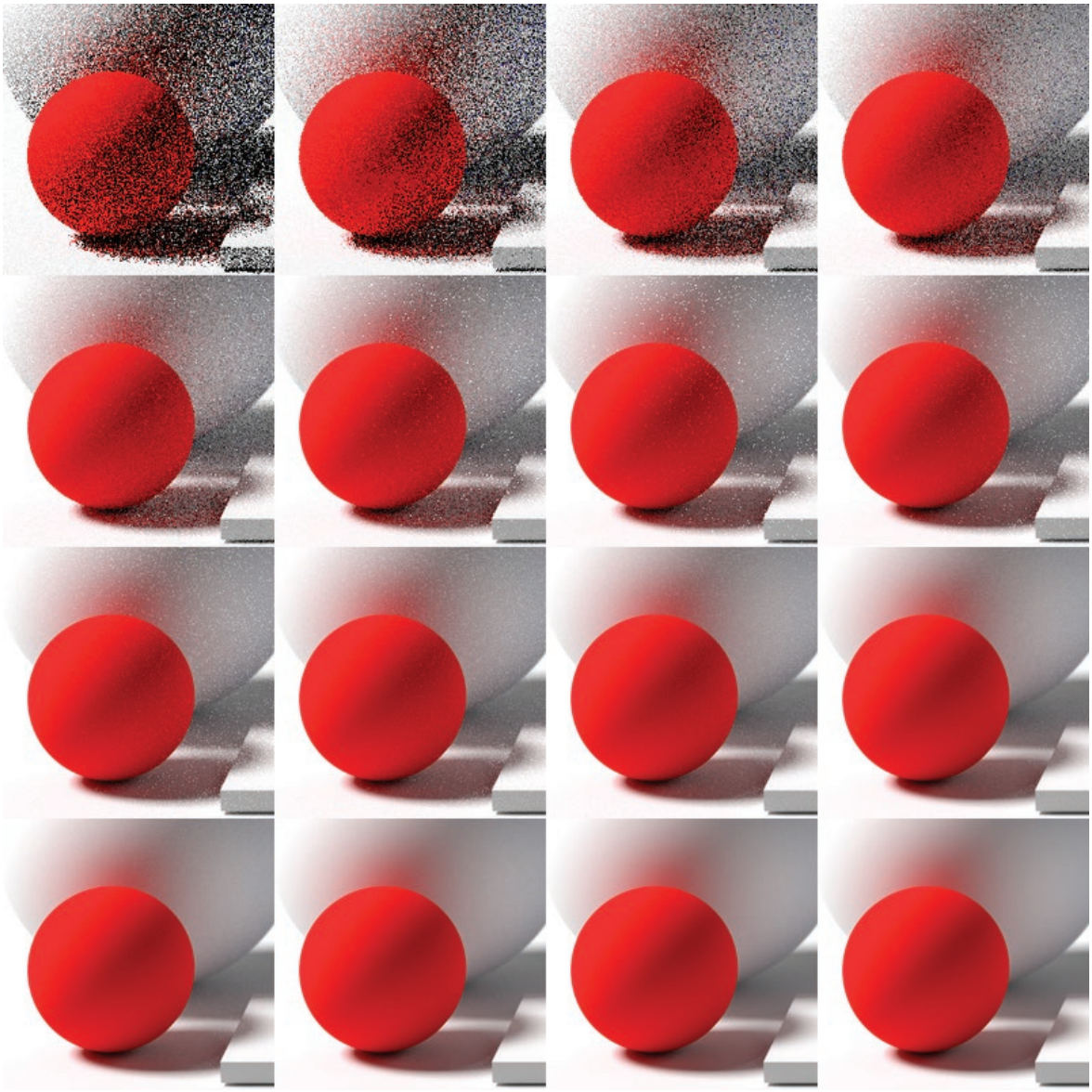
バウンスごとに、レンダラーは直接光と間接光の両方を考慮してライトの情報を収集します。これらすべてのバウンスから色情報が蓄積され、最終的なピクセルの色が決定されます。



パストレーシングスキャタリング。出典:Wikipedia

パストレーシングは、微かな乱反射やソフトシャドウなど、グローバルイルミネーションの効果をより自然な形で捉えます。この技術は、基本的なレイトレーシングに比べ、より物理的に正確でリアルな画像を生み出すことができます。

しかし、パストレーシングは複数のバウンスを計算するため、レイトレーシングよりも計算コストが嵩みます。特にリアルタイムのパフォーマンスにおいては、必要なサンプル数が少ないため、ノイズが問題になることもあります。



パストレーシングはより少ないサンプルでノイズを生成します。出典：Wikipedia

HDRP には現在、これを軽減するため、以下の新しいノイズ除去技術が搭載されています。

— NVIDIA Optix™ AI-accelerated denoiser

— Intel® Open Image Denoise (オプトインパッケージとして利用可能)

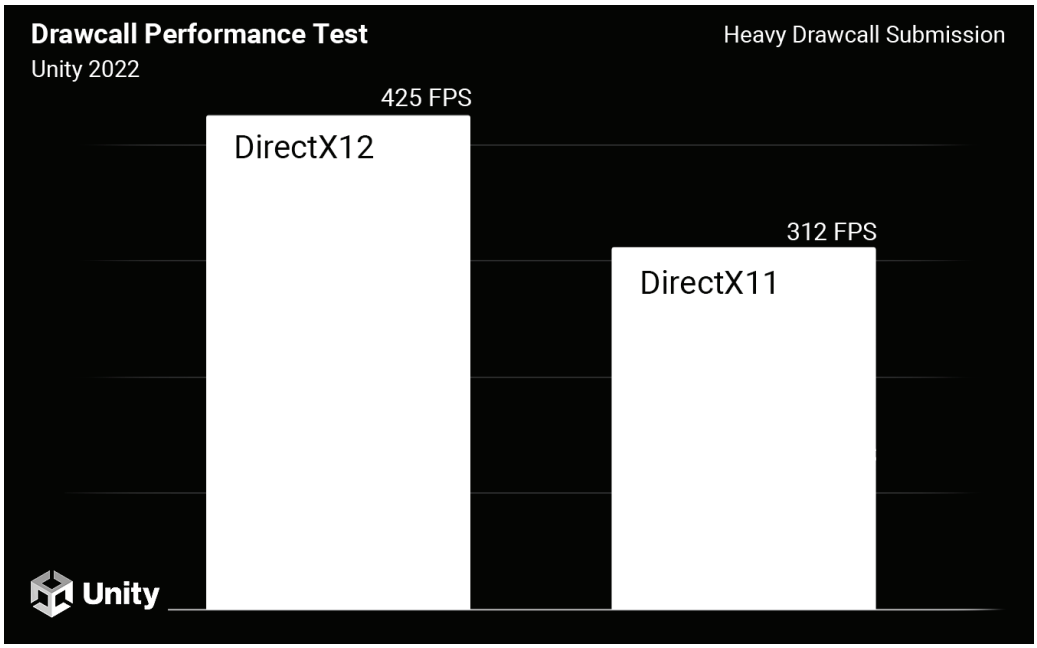
— また、HDRP は、パストレーシングを使用するマテリアルシェーダーに **AOV** (Arbitrary Output Variable、任意出力変数) を使用する設定を追加し、新たなパストレーシングのノイズ除去をサポートしています。この設定を有効にすると、HDRP はアルベドと法線の値を AOV に入力し、パストレーシングの結果の向上に役立てることができます。

DirectX 12

DirectX 12 (DX12) が Windows のデフォルトのグラフィックス API となり、必要に応じて DirectX 11 (DX11) にフォールバックします。Unity の DirectX 12 (DX12) グラフィックスバックエンドには、著しい改良が施されています。多くのドローコールが必要になる状況で、DX12 はスタンドアロンビルドにおいて優れた CPU 性能を発揮します。

ただし、DX12 のパフォーマンスが常に DX11 を上回るとは限りません。例えば、DX11 のドライバーは、DX12、Metal、または Vulkan よりも効率的にコンピュートシェーダーのディスパッチコールの順序を変更できます。GPU ワークロードが多いシーンや複雑なコンピュートシェーダーを使用するシーンに関しては、DX11 の方がより高いパフォーマンスを発揮する可能性があります。

エディターのパフォーマンス向上のため、DX12 ではエディターで ネイティブグラフィックスジョブ を実行するオプションが導入されました。このオプションはコマンドライン引数 **-force-gfx-jobs native** を使用してアクティベートできます。



Unity 2022 での DirectX 12 と DirectX 11 のパフォーマンステストの比較

— GPU 依存のプロジェクトの場合、DX11 をお選びください。

— CPU 依存の場合は DX12 を使用してください。

— エディターの性能を最大限に引き出したい場合は、エディターのネイティブグラフィックスジョブと DX12 を併用してみてください。

Unity 6 のサンプル

レイトレーシングの実際の動作を確認するのに最適な方法は、Unity 6 の Package Manager にある新しい Transparency サンプルです。このサンプルシーンでは、レイトレーシングによる透明度、リフレクション、ライティングのインタラクションを紹介しています。レイトレーシングエフェクトを有効にして、ガラス、水などの透明なマテリアルをよりリアルにレンダリングします。

Transparent サンプルでは、レイトレーシングエフェクトを紹介しています。

環境ライティング

現実世界では、光は物体の表面で反射し、大気中で散乱し、地面でも反射します。このような間接的な光は、シーン全体の明るさと色合いを決定づける役割を果たします。

HDRP では、環境ライティングがこの効果をシミュレートし、直接の光源がない場合でも、オブジェクトが空からの環境光を確実に受けるようにします。これは、ランダムな光子が大気と地球の間で跳ね返り、最終的に観測者に届くためです。

Unity では、空はカメラがフレームをレンダリングする前に描く背景の一種です。この種の背景は奥行き感を与え、環境を実際よりはるかに大きく見せます。また、空を利用して、シーン内にリアルな環境光を生成することもできます。

他の光源を無効にしている場合でも、SampleScene は Visual Environment の環境光に照らされます。

デフォルトの Lighting Data Asset

Unity 6 では、従来の SkyManager システムが新しい **デフォルトの Lighting Data Asset** (ライティングデータアセット) に置き換えられ、エディターに埋め込まれました。

新しいシーンを追加すると、このアセットが自動的に割り当てられます。Unity にビルトインされた Default Skybox から環境ライティングをキャプチャする非表示のアンビエントプローブとリフレクションプローブを使用して、デフォルトの環境ライティングが得られます。この新しいデフォルトアセットは Project ウィンドウに表示されません。

非表示プローブには、更新されない固定の計算済みデータが含まれていることに注意してください。Lighting ウィンドウで Skybox のマテリアルを変更しただけでは、これらのプローブは新しい Skybox をキャプチャしません。

デフォルトの Lighting Data Asset の使用を停止するには、Lighting ウィンドウで **Generate Lighting** または **Clear Baked Data** を選択します。その後、新しいデフォルトの Lighting Data Asset と新しいベイク済みリフレクションプローブ (アンビエントリフレクションプローブ) が作成され使用されます。これらのアセットは Project ウィンドウに表示され、Generate Lighting を選択すると更新されます。

以前は、SkyManager は環境ライティングがデフォルトでシーンに影響を与えることを保証できましたが、エディターとビルドされたプレイヤーの間で不整合が生じることもありました。新しいデフォルトの Lighting Data Asset により、ワークフローによる成果がより予測しやすくなります。

環境ライティングのみ：日光のディレクショナルライトが無効ですが、空には環境光が残っています。

太陽のキーとなるライトを追加することで、シーンの全般的なイルミネーションが決まります。環境光は、シャドウエリアを照らして、不自然に暗くならないようにするのに役立ちます。

直接的な日光と環境ライティングの組み合わせ

視覚的な環境

ほとんどの場合、デフォルトの空は、シーンのライティングや芸術的な目標に合ったカスタム設定に置き換えるものです。

HDRP では、**Visual Environment** (視覚的な環境) のオーバーライドを使用して、シーンの空と環境光を定義できます。

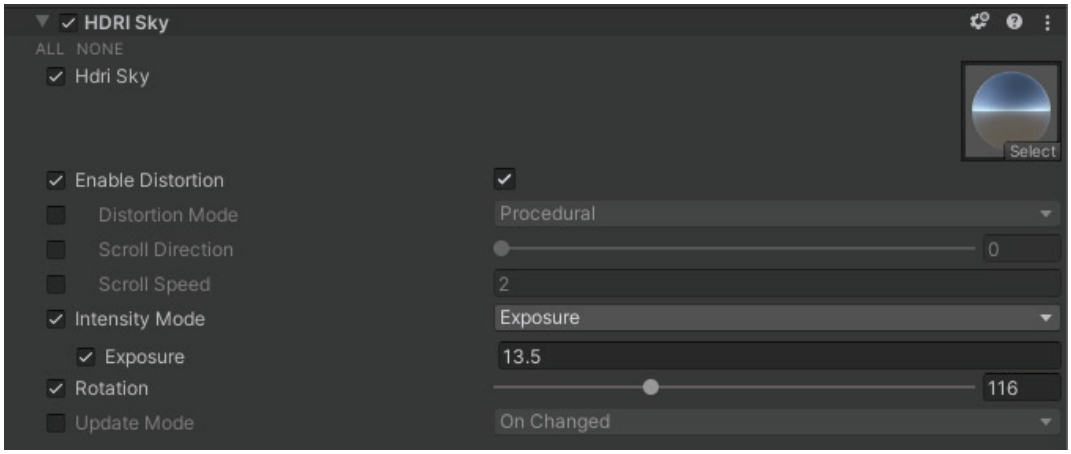
Ambient Mode:Dynamic を使用して、空のライティングを、Visual Environmen の **Sky > Type** に表示される現在のオーバーライドに設定します。または、**Ambient Mode:Static** を使用する場合は、**Lighting** ウィンドウの **Environment** タブの空の設定がデフォルトになります。

HDRP には、空を生成する手法が 3 種類あります。**Type** を **HDRI Sky**、**Gradient Sky**、または **Physically Based Sky** のいずれかに設定します。続いて、Sky メニューから適切なオーバーライドを追加します。

Visual Environment の空を適用すると、バーチャルワールド全体が、明るく照らされた巨大な球体で囲まれたような状態になります。この球体の色付きのポリゴンが、空、地平線、地面からの全般的なライトを提供します。

HDRI Sky

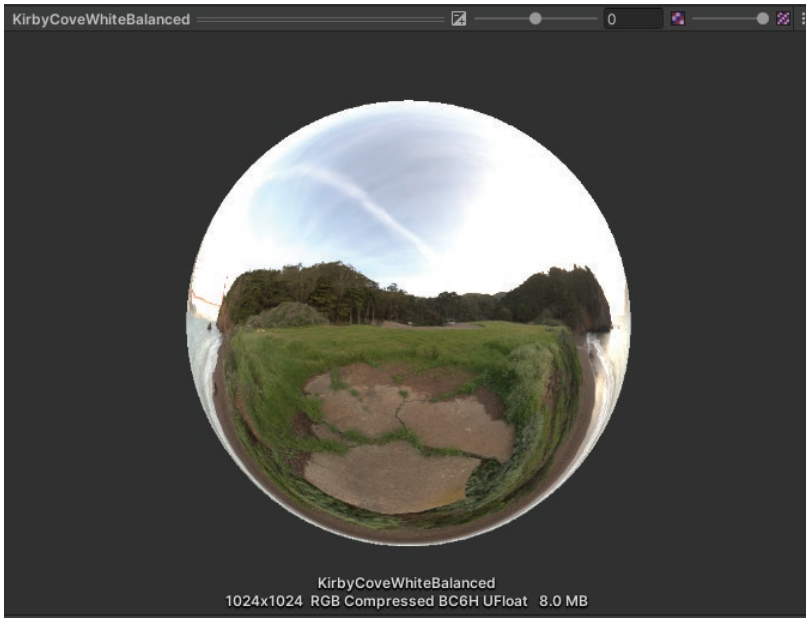
HDRI Sky のオーバーライドでは、ハイダイナミックレンジ（HDR）の写真から作成したキューブマップを使用して空を表現できます。HDRI については、無料のソースや低コストのソースがオンラインに多数あります。出発点としては、アセットストアにある Unity HDRI Pack が適しています。
挑戦したい方のために、自分で HDRI を撮影するためのガイド も用意しています。



HDRI Sky アセット

HDRIアセットをインポートしたら、**HDRISky**のオーバーライドを追加し、**HDRISky**アセットをロードします。**Distortion**、**Rotation**、**Update Mode** のオプションを調整することもできます。

空はイルミネーションのソースになるため、**IntensityMode**を指定してから、対応する**Exposure/Multiplier/Lux** の値を選択して環境ライティングの強度を制御します。強度と露出の値の例については、前述のライティングと露出のチートシートを参照してください。



キューブマップとして球体の内部に適用された HDRI Sky

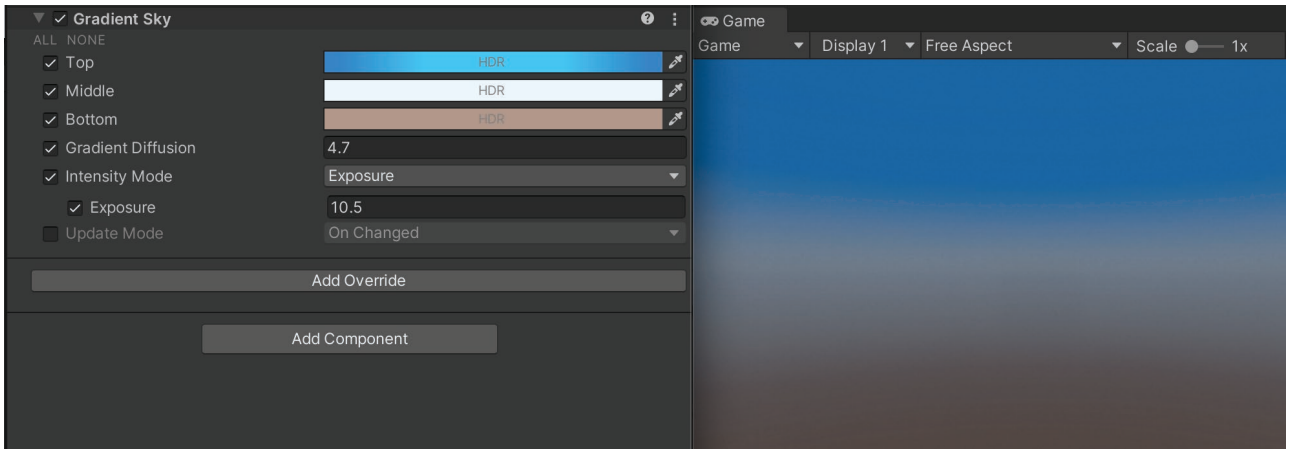
HDRI Sky のアニメーション

HDRI マップをプロシージャルまたはフローマップで歪ませることで、**HDRI Sky** をアニメーション化できます。これにより、静的な HDRI に風のエフェクトを加えたり、より特殊な VFX を作成したりできます。

Gradient Sky

Visual Environment のオーバーライドで **Gradient Sky** を選択すると、カラーランプを使用して背景の空を近似します。続いて、**Gradient Sky** のオーバーライドを追加します。**Top**、**Middle**、**Bottom** を使用して、グラデーションの色を指定します。

カラーランプを **Gradient Diffusion** とブレンドし、ライティングの強さの **Intensity** を調整します。

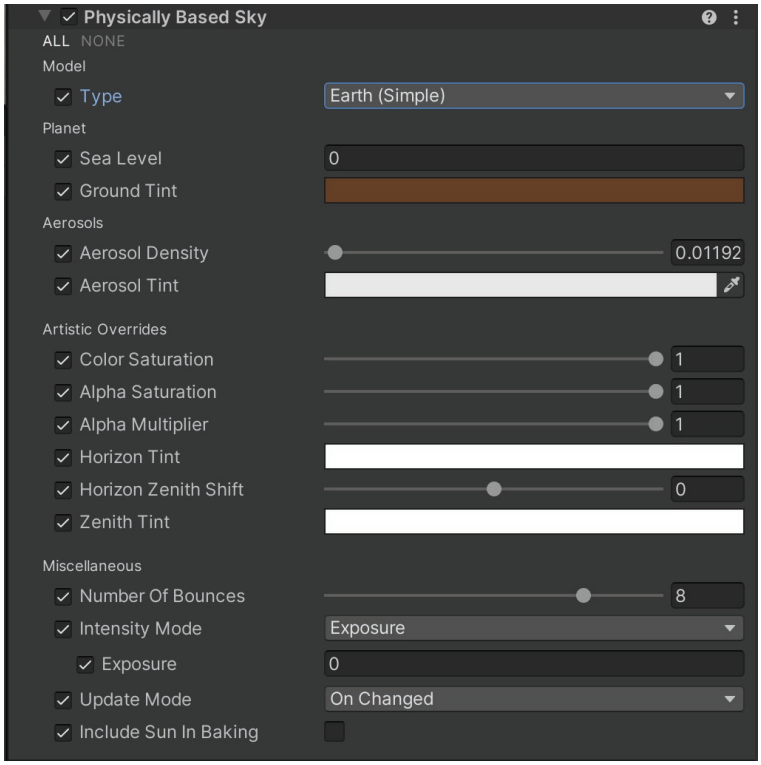


Gradient Sky では、Top、Middle、Bottom の色をブレンドします。

Physically Based Sky

グラデーションより大幅にリアルさを高めたい場合は、**Physically Based Sky** (物理ベースの空、PBR) のオーバーライドが有効です。

HDRP の Physically Based Sky オーバーライドは、空気とエアロゾルの微粒子で構成される 2 つの部分からなる大気をシミュレートします。これは、Volume システムを使用して、Physically Based Sky オーバーライド で調整できます。



Physically Based Sky のオーバーライド

このオーバーライドにより、ミー散乱 や レイリー散乱 などの現象を再現した空がプロシージャルに生成されます。この空では、大気を通って分散する光のシミュレーションが行われ、自然な空の配色が再現されます。PBR Sky では、正確なシミュレーションを行うためにディレクショナルライトが必要です。Physically Based Sky のオーバーライドは、環境ライティングソースとしても機能します。

『Time Ghost』の PBR Sky

色についてのヒント

地面の色は、オブジェクトがリフレクションプローブの影響を受けない実際の地面（地形など）の平均的な色に合わせて選んでください。

Unity 6 の PBR sky に関するアップデート

メモリ使用量の軽減とパフォーマンスの向上により、PBR の空がより効率的になりました。マルチスキャタリングでは、特定の事前計算ステップが不要になり、太陽が地平線下にあるときの光の散乱が改善されます。

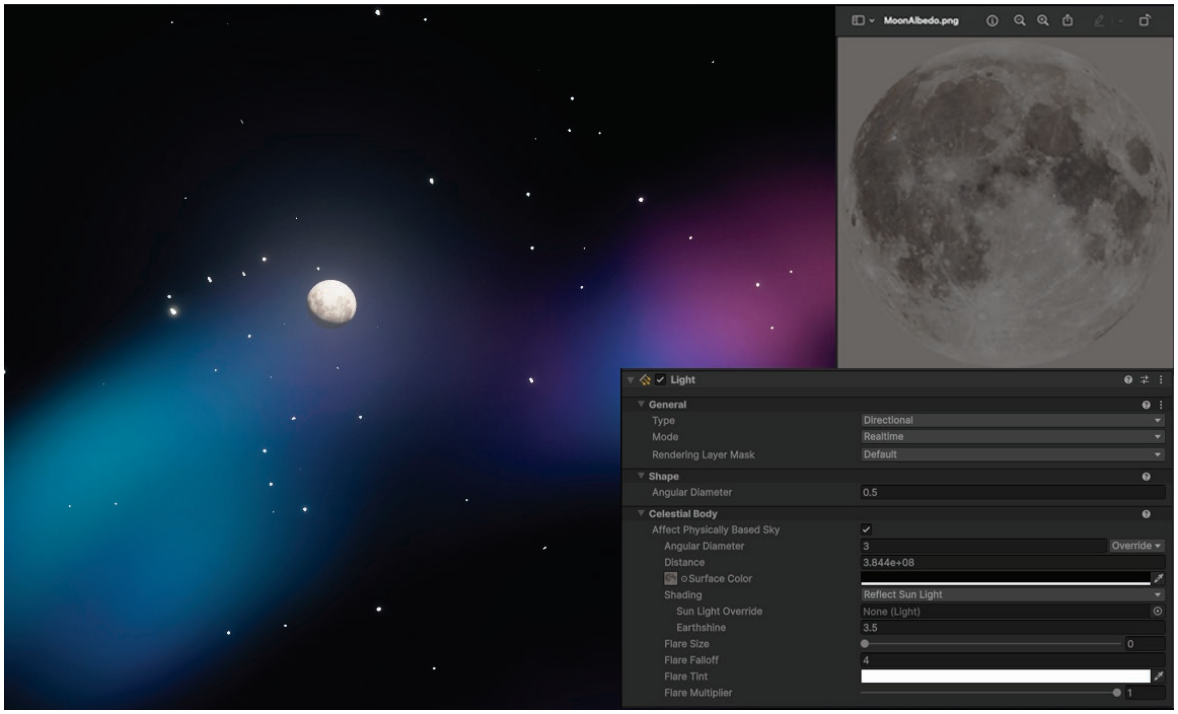
PBR Sky の大気モデルにオゾン層が加わりました。これは、夕暮れなど、太陽が地平線の近くにあり、太陽光が低い角度で厚い大気を通過する際に、微妙に紫がかることで確認できます。

オゾンは日没時に空を染めます。

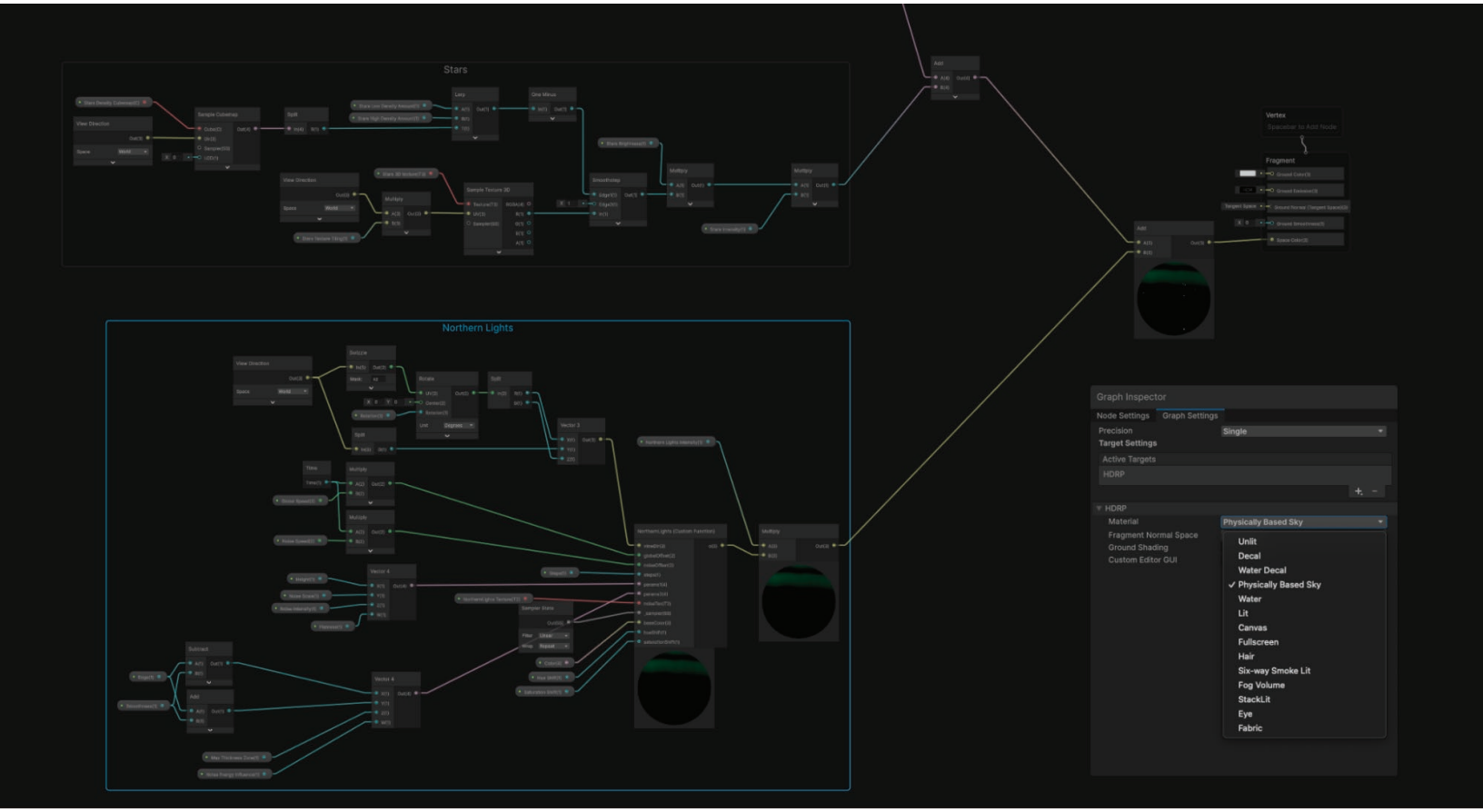
空気遠近法 では、大気中の微粒子による光の吸収をシミュレートできるようになり、遠くの山や雲などのオブジェクトを自然にフェードアウトさせることができます。

大気散乱は、雲をよりリアルにレンダリングします。

Unity 6 には、夜空のレンダリングを簡素化する新機能が備わっています。Shader Graph の **PBRSky マテリアルタイプ** では、プロシージャルスターなどのエフェクトを使用できます。また、新しいオプションにより、天体を月としてマークし、メインのディレクショナルライトからのライティングを確実に受けることができます。



天体で月を表すことができます。



Shader Graph は、プロシージャルスターやオーロラなどのエフェクトを可能にします。

大気散乱

パフォーマンスを最適化するために、大気散乱を空に適用し、高高度でのボリュメトリックフォグのコストを回避しながら、青い光が遠くのオブジェクトや雲に散乱する様子をシミュレートします。このシステムは、3 つの主要なインタラクションをモデル化します。

— **レイリー散乱** (空気分子) が空の青色を作り出します。

— **ミー散乱** (エアロゾル) は、もや、フォグ、公害のエフェクトをもたらします。

— **オゾン吸収** は、散乱に影響を与えることなく空のカラーグラデーションを変更します。

空のレンダリングが、遠くにあるオブジェクトや雲の大気散乱をシミュレートするようになり、環境光との相互作用が改善されました。これは時間帯の遷移に便利で、夕方から夜に移り変わる夕暮れ時に不自然に暗い雲が発生するのを防ぎます。

その結果、ライティングの遷移がより滑らかになり、フォグの距離に大きな値を設定しなくても空がより自然に見えるようになりました。さらに、エアロゾル散乱も改良され、よりリアルな大気の奥行きと一貫性のあるレンダリングが確保されています。

レンダリングスペース

Visual Environment の **Rendering Space** 設定によって、惑星と空のシミュレーション動作が決まります。

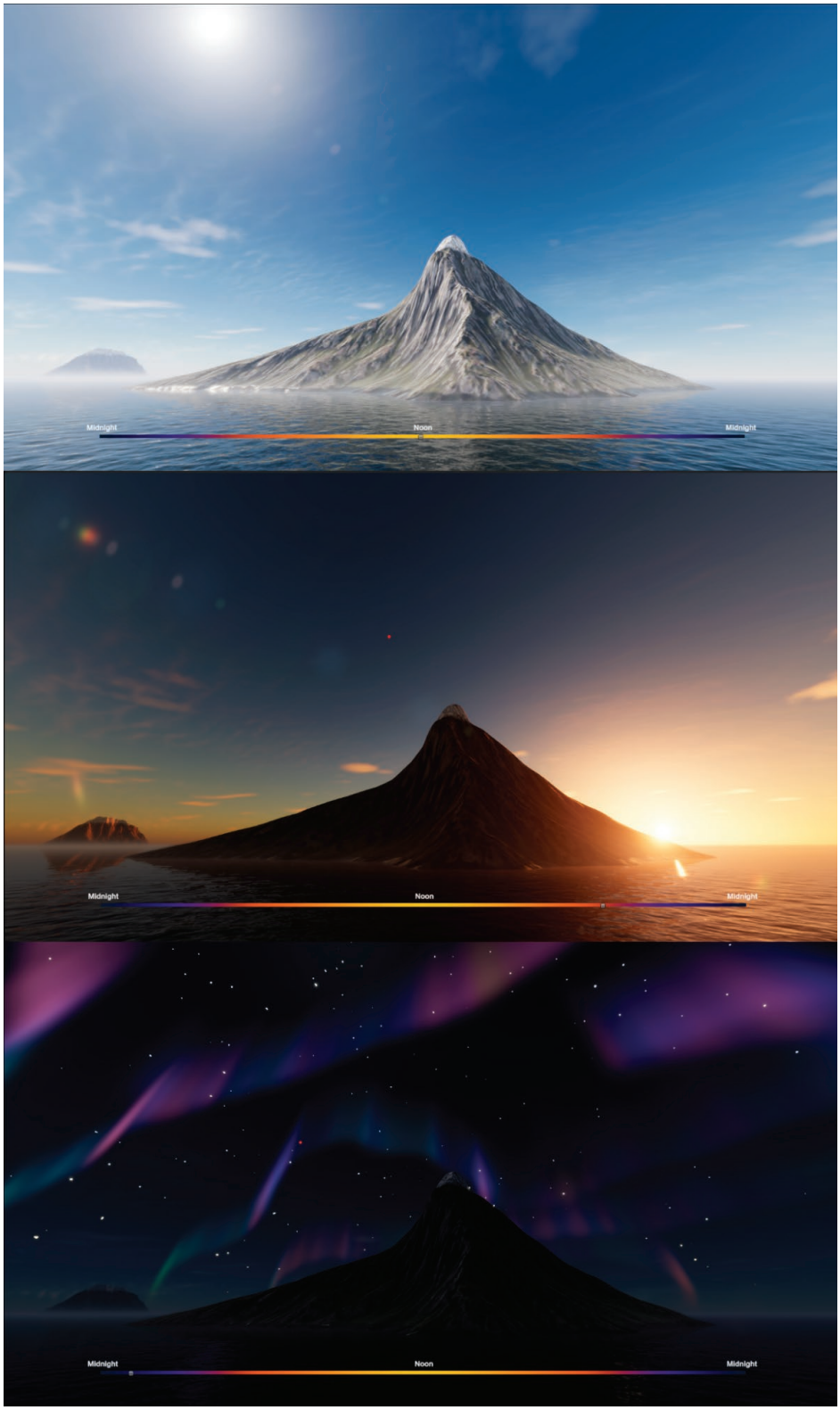
— **World モード**:惑星の表面はワールド原点に固定されたままで、その中心は Radius プロパティから得られます。大気散乱は、考えられるすべての光と視野角について事前に計算され、3D テクスチャに格納されて、ランタイムにリサンプリングされます。このモードでは、カメラは宇宙空間まで自由に移動できます。

— **Camera モード**:惑星はカメラと一緒に動き、地表は常にカメラの下にあります。このモードでは、シーン全体ではなく、特定の光のグループについて空の散乱を事前計算します。

なお、惑星表面は深度バッファでレンダリングされないため、レンズフレアが隠蔽されたり、モーションブラーで正しく動作したりすることはありません。

Environment Sample

これらの機能を実際に確認するには、**Package Manager** から **Environment Sample** をインポートします。時間帯システムを使用した空と雲のある島のシーンが含まれており、真夜中、正午、夕暮れの間を遷移します。



Environment Sample は空の設定を示しています。

Environment Sample では、HDRP のワールド構築ツールの地形と雲を使用しています。これについては、次の 2 つの章で説明します。

Terrain

ゲームの世界はプレイヤーの足元から始まります。Unity の Terrain Editor を使えば、規模にかかわらず、詳細かつ現実に忠実な風景を作ることができます。

地形オブジェクトは、地形図のようにスカルプトできる平面として始まります。高さの値をペイントすることで、山を高くしたり谷を削ったりすることができ、相互につながったタイルからバーチャルな世界を組み立てることができます。次に、プロシージャルまたはアーティストのブラシによって、テクスチャと植物で風景を仕上げます。

Terrain Tools パッケージは、追加のスカルプティングブラシや、地形アセットの編集や設定を自動化する一連のユーティリティによって、ビルトインのワークフローをさらに改善します。

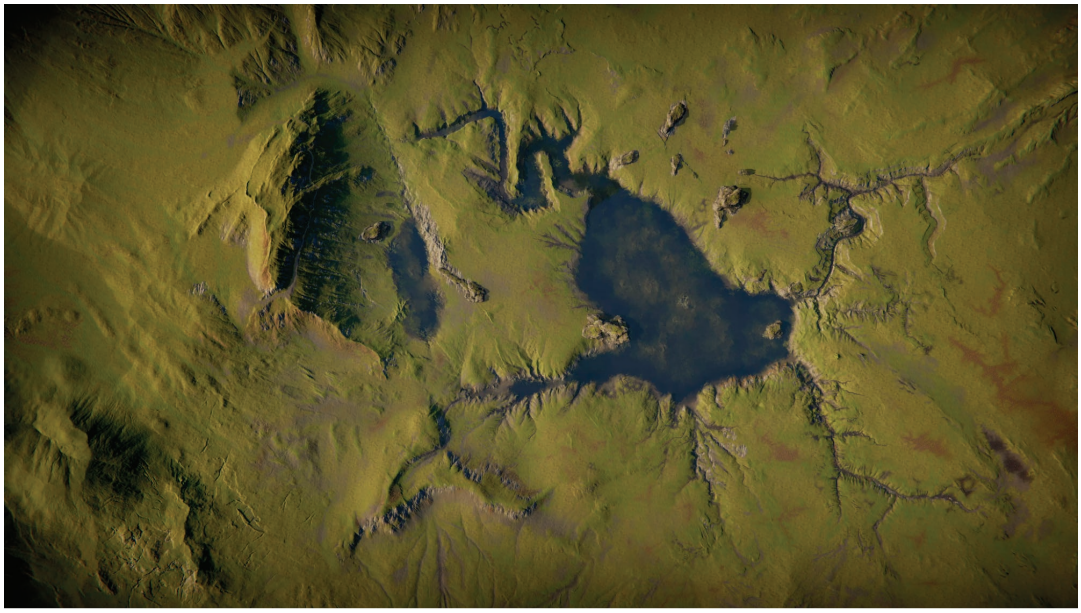
地形システムは、Cloud Layer Volume コンポーネントのオーバーライドと連動します。

特定のツールや手法を使用して地形を作成およびカスタマイズする方法を、Unity Learn のこの 一連のチュートリアル で確認してください。

地形の作成

Hierarchy で新しい地形を作成すると、Unity はシーンに大きな平面を追加します。この点のグリッドは、各頂点で変更することができます。

地形の形状は、グレースケールの**ハイトマップ**によって決定されます。このグレースケールのマップは、グリッド上の各頂点をどのように上げ下げするかを決定します。白は最も高い点、黒は最も低い点を表します。

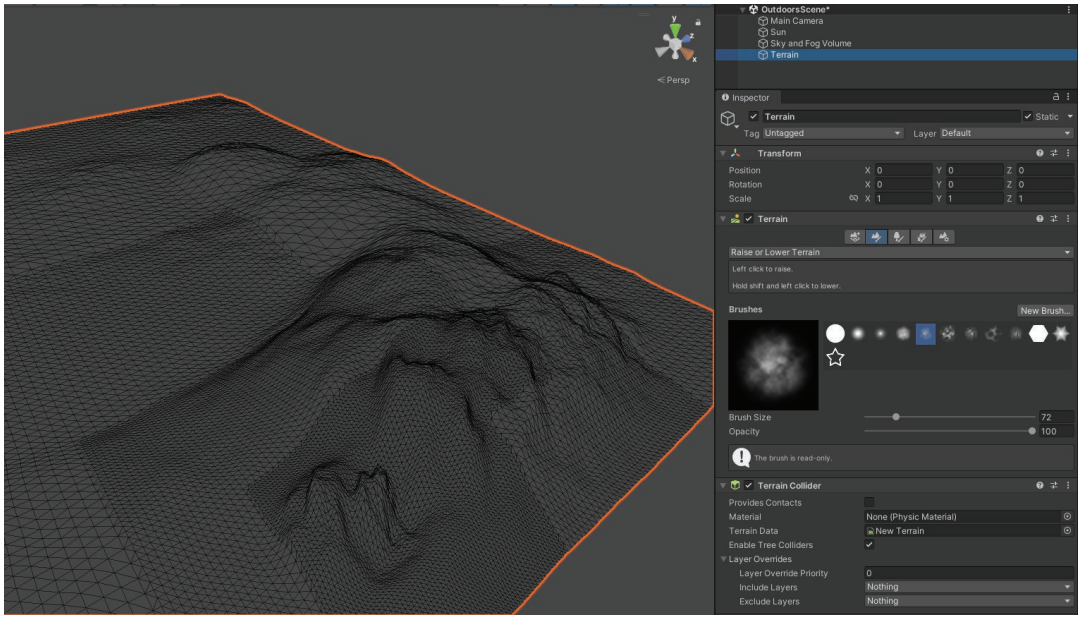


ハイトマップにペイントして地形を修正します。

スカルプティング

Terrain コンポーネントには、地形をスカルプトするためのさまざまなブラシが含まれています。これらを使って地形の一部を上下させたり、エリアを滑らかにしたり、特定の形状やパターンのカスタムブラシを作成することもできます。

Unity 6.1 では、ブラシが実際に使用されるまでフル解像度のブラシテクスチャを作成しないため、Terrain Inspector のパフォーマンスが向上しています。メモリ使用量が減り、Terrain Inspector がより高速に、そして効率的になっています。

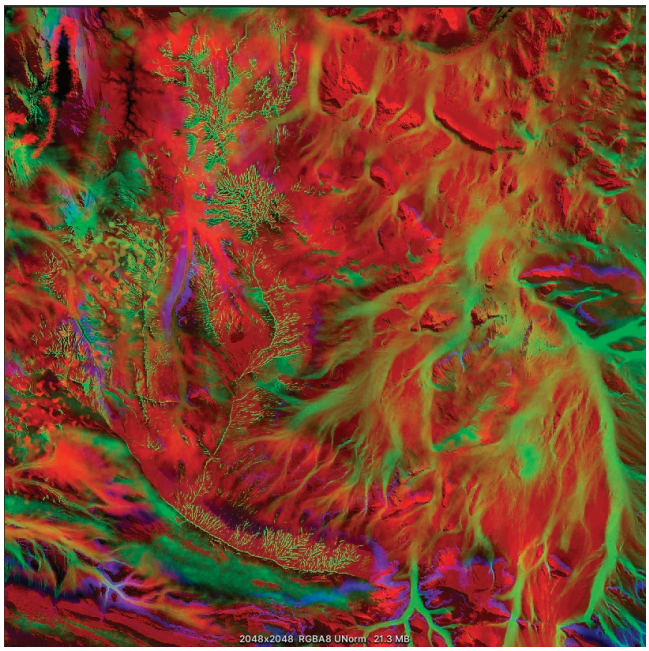


ブラシを使って地形をスカルプトします。

テクスチャリングとディテーリング

スプラットマップを使えば、地形にさまざまなテクスチャを描くことができます。これは画像編集ソフトのレイヤーのようなもので、テクスチャを定義してブレンドすることができます。

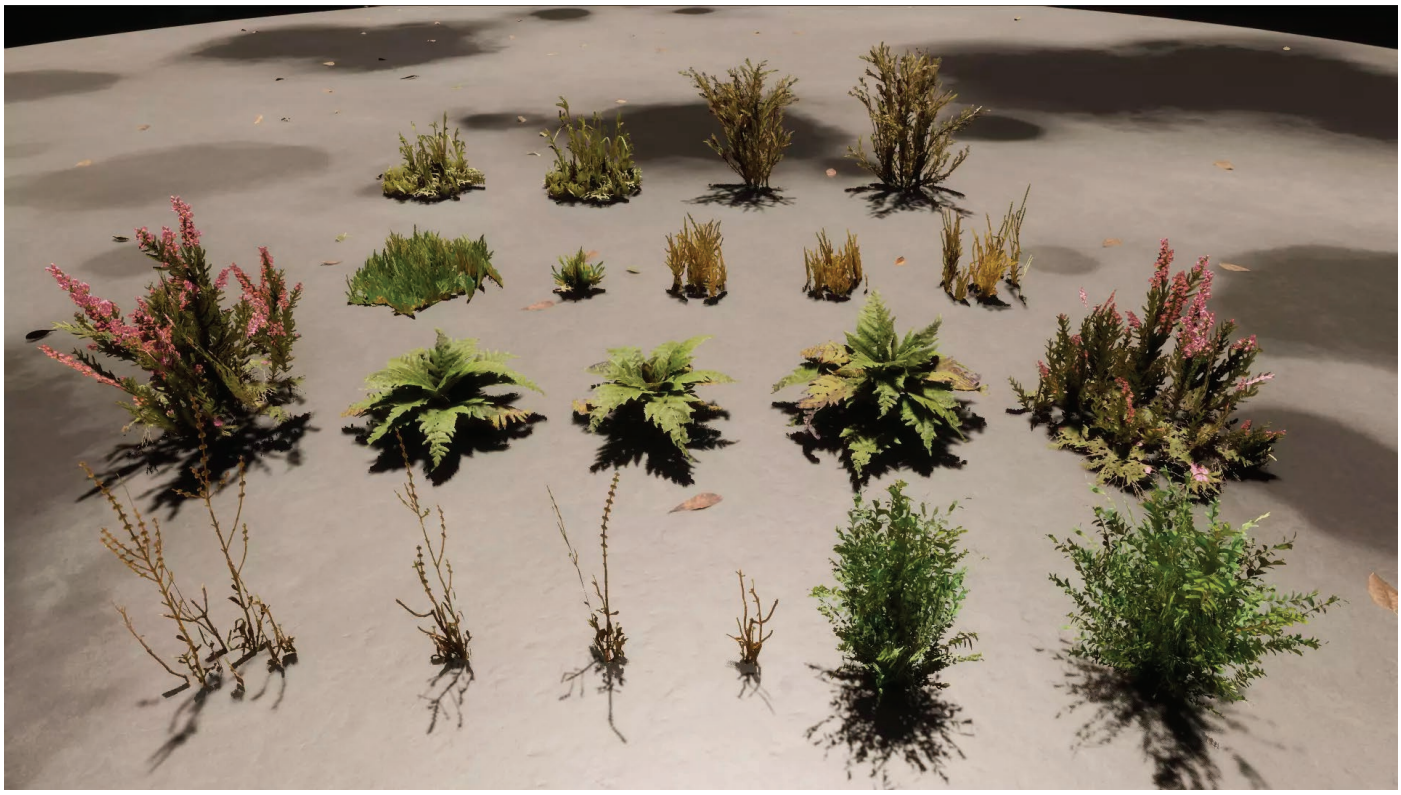
そして、草や花、小さな岩などの細かいディテールには **ディテールマップ** を使用します。これらは地形に直接ペイントしましょう。



スプラットマップを使って地形のテクスチャを描きます

樹木および植生

地形を完成させるには植物が必要です。Unity は Terrain コンポーネントの一部として Tree Editor を提供しています。これは、さまざまな樹木の種類やバリエーションを含む、リッチな森やジャングルを作成したい場合に便利です。



地形システムは高度な植生機能を提供しています。

Terrain コンポーネントの樹木パレットは、成長段階が異なる樹木を混ぜることで、より自然な外観を作り出すことができます。樹木を個別に配置することも、大量配置ツールを使うこともできます。その後、草や花のテクスチャのビルボードを使って、ディテールのレイヤーを追加することができます。

Wind Zones コンポーネントを追加すれば、樹木や植生が風で揺れ、シーンに命が吹き込まれます。

SpeedTree インテグレーション

Unity は、SpeedTree モデル (.SPM または .ST ファイル) の直接インポートにも対応しており、標準のゲームオブジェクトと同様に扱うことができます。Terrain Editor を使用して、SpeedTree の植生を地形に直接ペイントしましょう。ペイントされた樹木には、ランタイムに他のオブジェクトと相互作用するためのコライダーが自動的に与えられます。

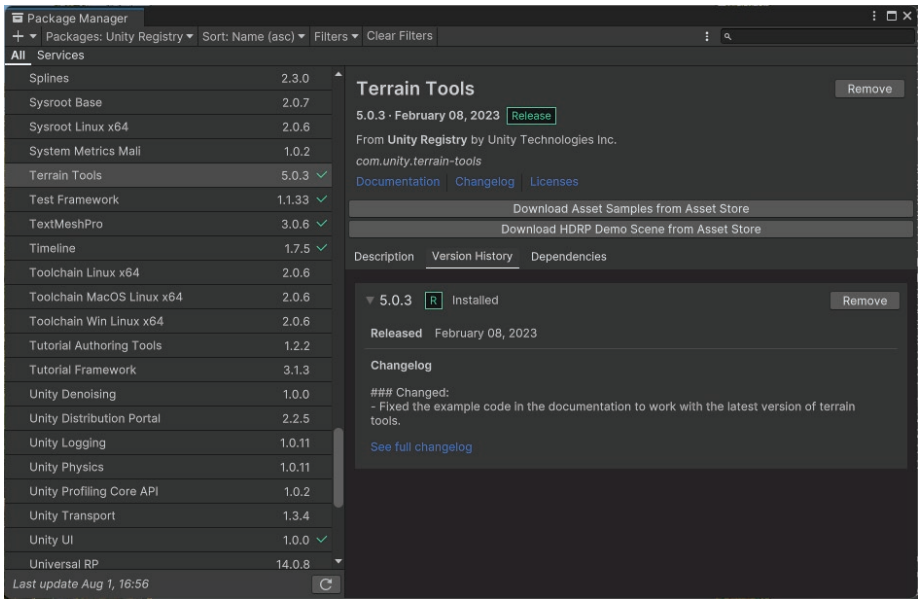
SpeedTree モデルには、パフォーマンスを最適化するための LOD が組み込まれています。レンダリングシステムは、効率化のためにモデルをバッチ処理します。SpeedTree の植生には影を与えることができ、グローバルイルミネーションに貢献できます。

SpeedTree の植生

Terrain Tools パッケージ

Terrain Tools パッケージは、地形スカルプト用のブラシやツールを追加します。このアドオンツールセットは、地形の外観をより細かく制御する必要があり、地形のワークフローを合理化したい場合に最適です。

Terrain Tools パッケージを追加すると高度な機能をご利用いただけます。

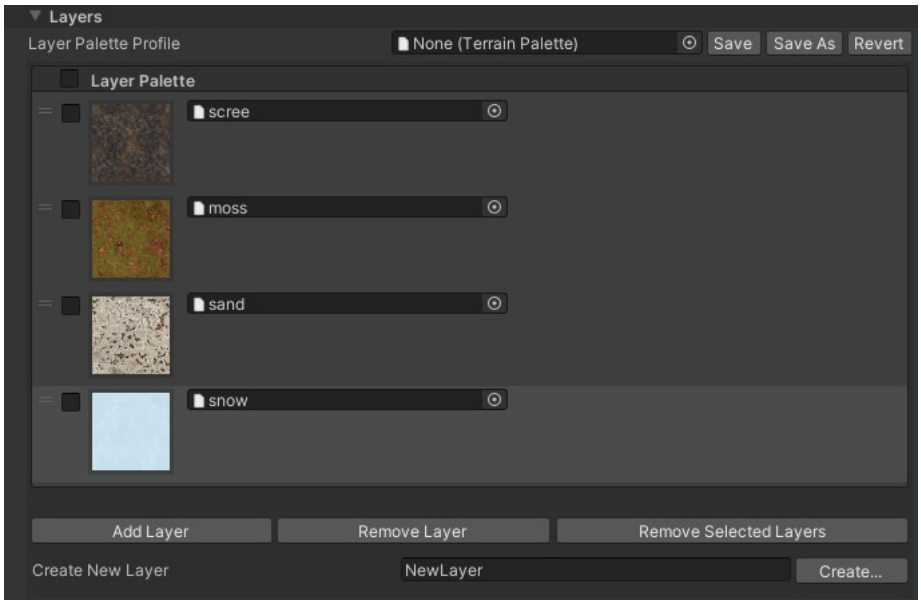


このパッケージは、より複雑に見える地形を作成したり、外部のデジタルコンテンツ作成ツールで地形テクスチャデータをオーサリングしたりするのに役立ちます。

詳しくは Getting started with Terrain Tools (Terrain Tools の使用を開始する) を参照してください。

地形のペイント

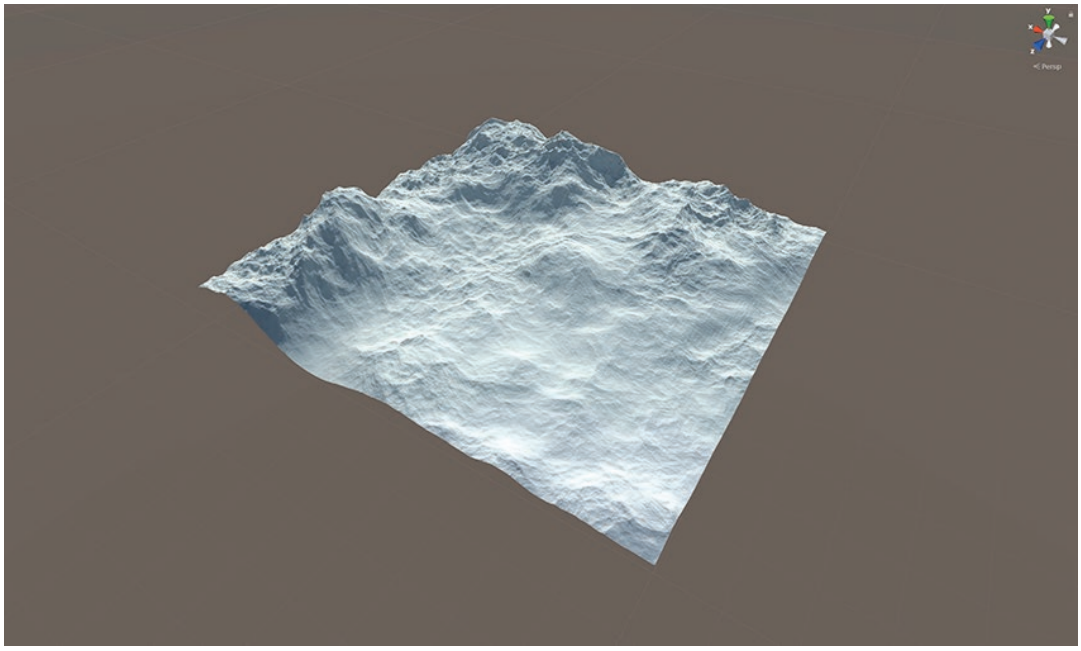
Terrain Tools パッケージは、ツールを追加するとともに、Paint Texture、Smooth Height、Stamp Terrain といったビルトインツールの機能を向上させます。Paint Terrain ツールの完全なリストについては、こちらの ドキュメントページ を参照してください。



Terrain Tools を使ったレイヤーでのペイント

Sculpt ツールは、加算法と減算法で地形の形状を変更します。

- Bridge は、選択した 2 点間にブラシストロークを作成し、陸橋を作成します。
- Clone は、ある地域から別の地域に地形を複製します。
- Noise は、異なるノイズタイプとフラクタルタイプを使用して地形の高さを変更します。
- Terrace は、地形を階段のような一連の平らなエリアへと変化させます。



Noise は地形の高さを変更します。

Effects ツールは、地形の既存の高さに基づいて修正します。

— Contrast は、地形の全体的な高低差を大きくしたり小さくしたりします。

— Sharpen Peaks は、地形のピークを鋭くしたり、すでに平坦な部分をさらに平らにしたりできます。

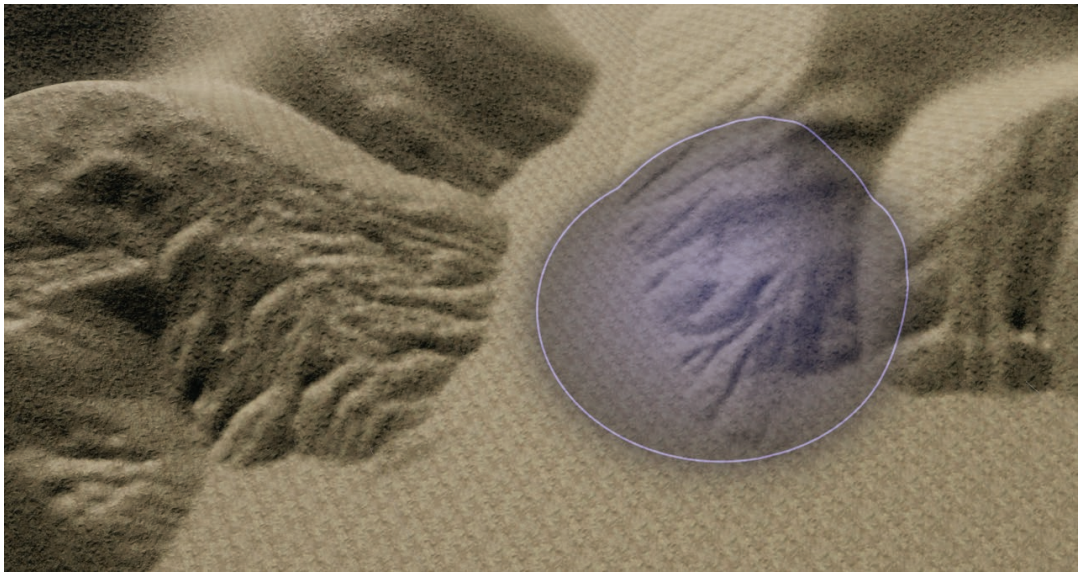
— Slope Flatten は、勾配の平均を維持したまま地形を平らにします。

Erosion ツールは、流水や風の影響、土砂の動きの効果のシミュレーションを行います。

— Hydraulic は、流れ場に沿った土砂による水の浸食のシミュレーションを行います。このツールを使って、谷や河川の地形を作成しましょう。

— Thermal は、自然な勾配を維持しながら、土砂が地形に沈殿する効果のシミュレーションを行います。

— Wind は、風による浸食と土砂の再分配の効果のシミュレーションを行います。



Erosion ツールは、風や流水の影響、土砂の動きの効果のシミュレーションを行います。

Transform ツールは、地形をプッシュしたり、引いたり、回転させたりします。

— Pinch は高さを引っ張ってブラシの中心に寄せたり、膨らませるようにして中心から遠ざけたりします。

— Smudge は、地形の特徴をブラシストロークの経路に沿って移動させます。

— Twist は、ブラシの中心を軸に、地形の特徴をブラシストロークの経路に沿って回転させます。

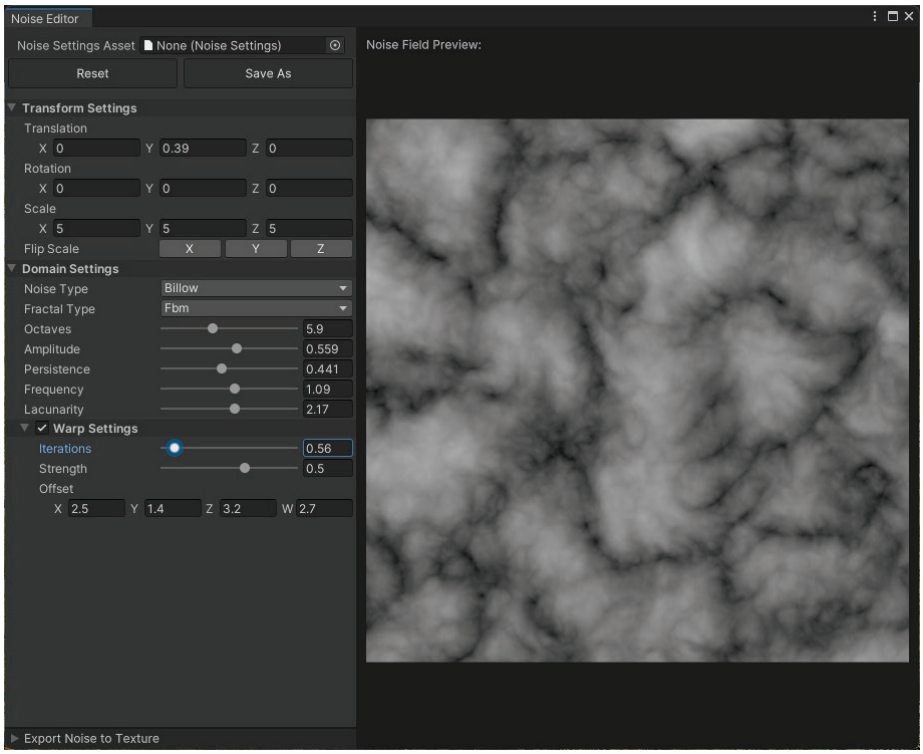
Twist は、ブラシの中心を中心に、地形の特徴をブラシの中心に沿って回転させます。

また、独自のカスタム地形ペイントツールを作成することもできます。詳しくは TerrainAPI.TerrainPaintTool および Create a custom terrain tool (カスタムの Terrain ツールの作成) を参照してください。

ノイズエディター

Noise Editor (ノイズエディター) では、Noise Height Tool や Noise Brush Mask Filter で使用できる Noise Setting (ノイズ設定) アセットの作成と管理を行うことができます。また、ノイズエディターを使って外部で使用するプロシージャルテクスチャを生成することもできます。

ノイズエディターを開くには、メニューから **Window > Terrain > Edit Noise** を選択します。

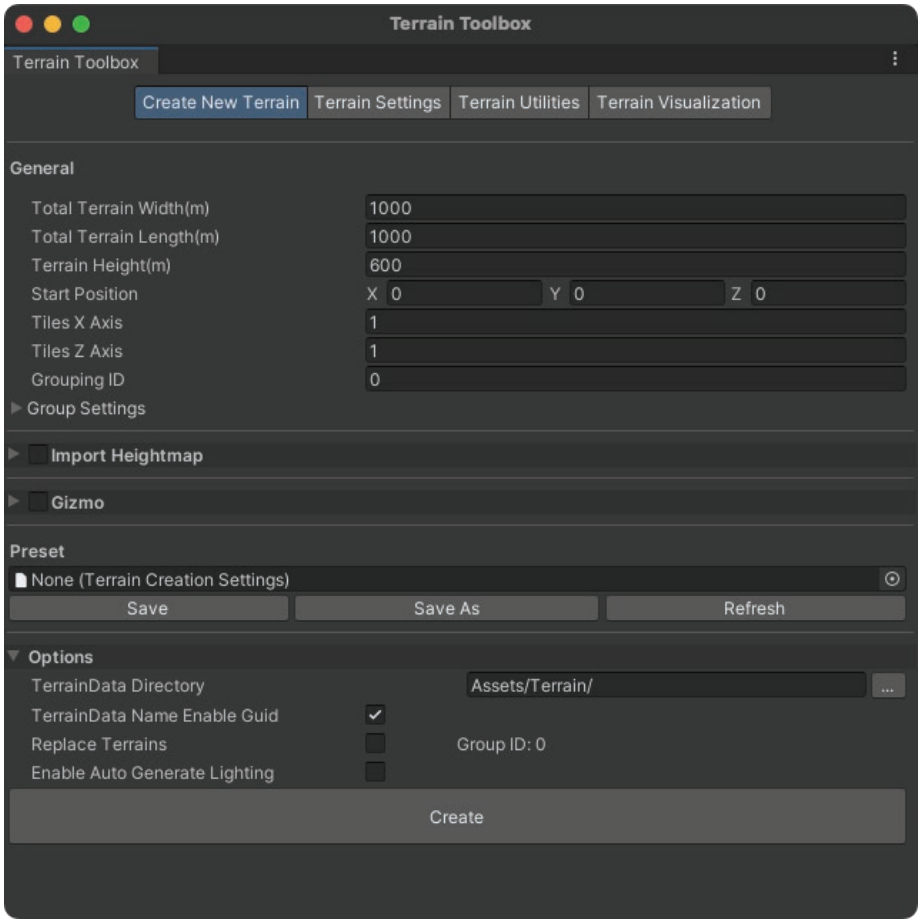


ノイズエディター

Terrain Toolbox

Terrain Toolbox は、地形ワークフローの効率化に役立つツールを提供するエディターウィンドウです。プリセット設定やインポートしたマップから新しい地形を作成したり、地形設定を一括で変更したり、スプラットマップやハイトマップをインポートまたはエクスポートしたりできます。

Terrain Toolbox を起動するには **Window > Terrain > Terrain Toolbox** を選択します。



Terrain Toolbox はワークフローを改善します。

地形におけるレイトレーシングサポート

Unity は、いくつかの制限付きで地形のレイトレーシングをサポートしています。地形はレイトレーシング効果の影響を受ける一方、積極的にそれに貢献することはありません。

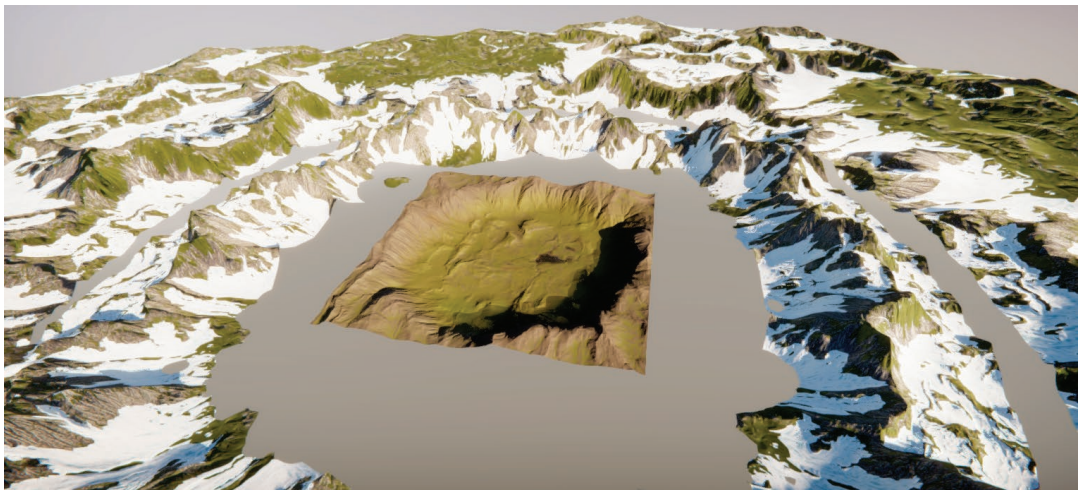
具体的には、レイトレーシングによるリフレクションでは地形が省略されます。代わりに、必要に応じて平面リフレクションを使う必要があります (例: 湖面など)。また、レイトレーシングによるアンビエントオクルージョンやグローバルイルミネーションも地形に影響しません。

HDRP 地形デモ

このパッケージは Unity Asset Store で入手可能 で、Unity Terrain システムを最大限に活用するのに役立つサンプルコンテンツを含んでいます。このサンプルパックに含まれるテクスチャ、ブラシ、モデルは、ご自身のプロジェクトにも使用できます。

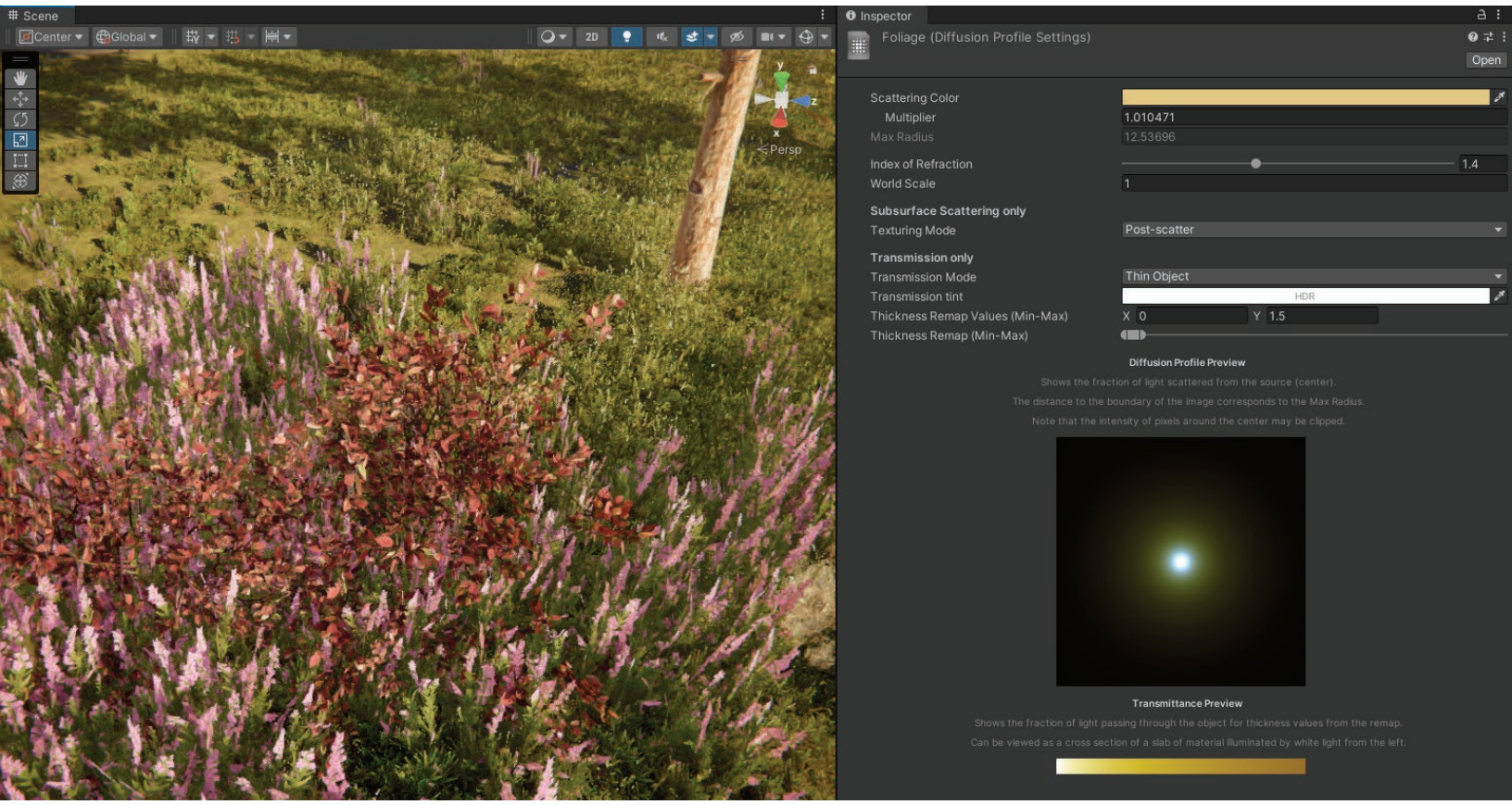
HDRP 地形デモは、Package Manager を介して入手できます。

このシーンには、メモやヒントが書かれた一連のブックマークロケーションが含まれています。Inspector のボタンを使って、興味のあるポイントにカメラの焦点を合わせ、ブックマークを見ていきましょう。



HDRP 地形デモの地形タイル

このシーンで使用されている植生シェーダーは、サブサーフェススキャタリングに HDRP Diffusion Profile アセットを利用しています。これにより、半透明の有機マテリアルを、ざらざらしたプラスチックのようではなく、滑らかで自然に見せることができます。



サブサーフェススキャタリングの設定は Diffusion Profiles に保存されます。

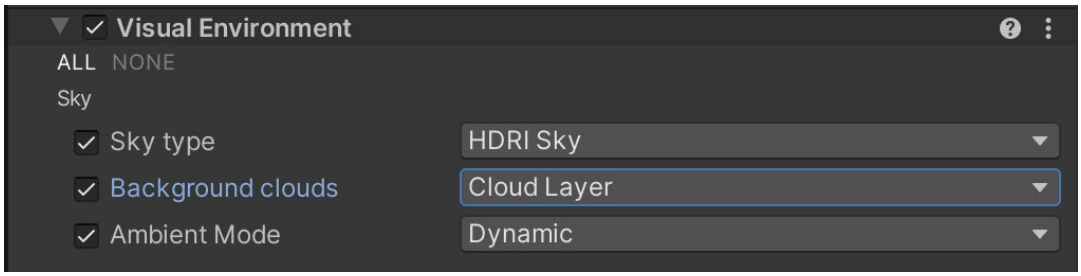
Clouds

空は雲がないと完成しません。**Cloud Layer** の Volume コンポーネントオーバーライドは、自然な雲を生成し、**Sky** や **Visual Environment** のオーバーライドを補完します。**ボリュメトリッククラウド**はライティングや風に反応し、実際に厚みを持つリアルな雲を作り出します。

Cloud Layer は HDRI の空の前に表示されます。

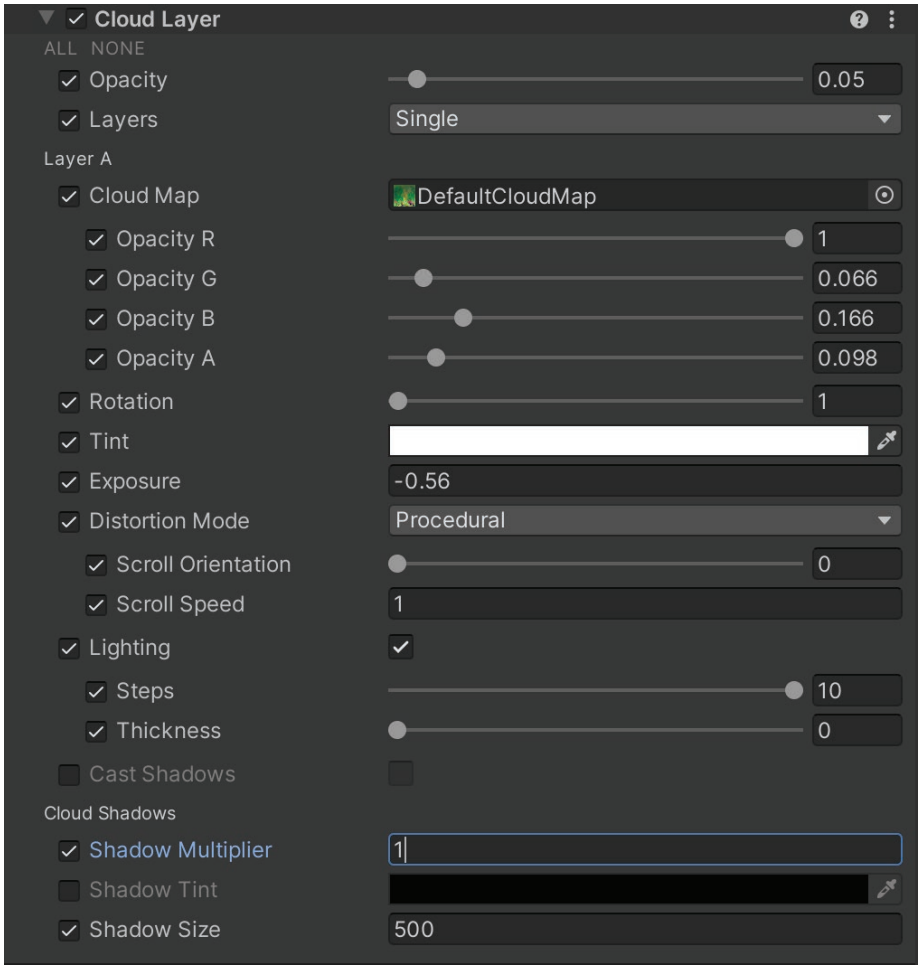
Cloud Layer

Cloud Layer のオーバーライドは、フローマップ でアニメーション化できる 2D テクスチャで、赤と緑のチャンネルを使ってベクターのディスプレイスメントを制御します。再生モードで、雲は空にわずかな動きを加えることができ、背景をより動的なものにします。雲のレイヤーは空の前に位置し、地面に影を落とすオプションもあります。



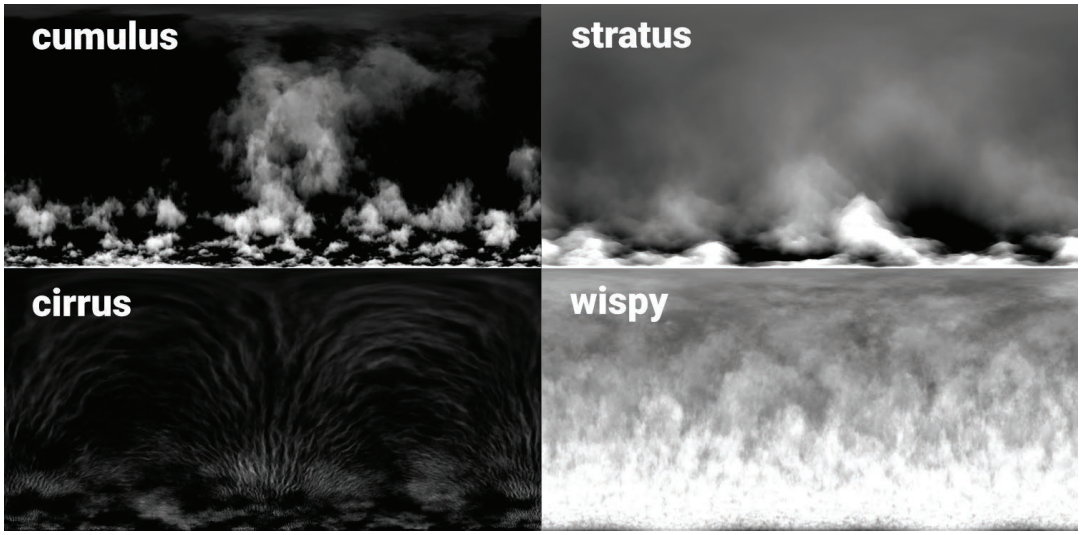
Visual Environment オーバーライドで Cloud Layer のオーバーライドを有効にします。

ローカルまたはグローバルのボリュームで、Visual Environment オーバーライドの **Background clouds** を有効にし、**Cloud Layer** のオーバーライドを追加します。



Cloud Layer のオーバーライド

雲マップ自体は 円筒投影 を使ったテクスチャで、RGBA チャンネルはすべて異なる雲のテクスチャ (積雲、層雲、巻雲、さざ波雲)　を含んでいます。その後、雲レイヤーの制御を使って各チャンネルをブレンドし、お好みの雲を形成することができます。.4 つのチャンネルを持つ 2 つのレイヤーで、最大 8 つの雲のシミュレーションを行い、ブレンドすることができます。



DefaultCloudMap の 4 つのチャンネル。

そして、好みのスカイスケープを作り出せるまで、雲のアニメーション、ライティング、カラー、シャドウを修正します。

大気および太陽に基づいたライティング

Unity 6 では、雲のレイヤーの物理ベースのレンダリングが改善されました。

雲レイヤーを Physically Based Sky のオーバーライドと組み合わせて使用する場合、太陽光の色が大気の減衰を正しく反映するようになりました。

さらに、レイマーチングが無効になっている場合でも、太陽光の色は常に雲の色に影響するようになりました。また、散乱を改善するためにレイマーチングのアルゴリズムにも改良が加えられ、ステップ数を変更しても、HDRP でより一貫した結果を得られるようになりました。

Physically Based Sky をオーバーライドした Cloud レイヤー。

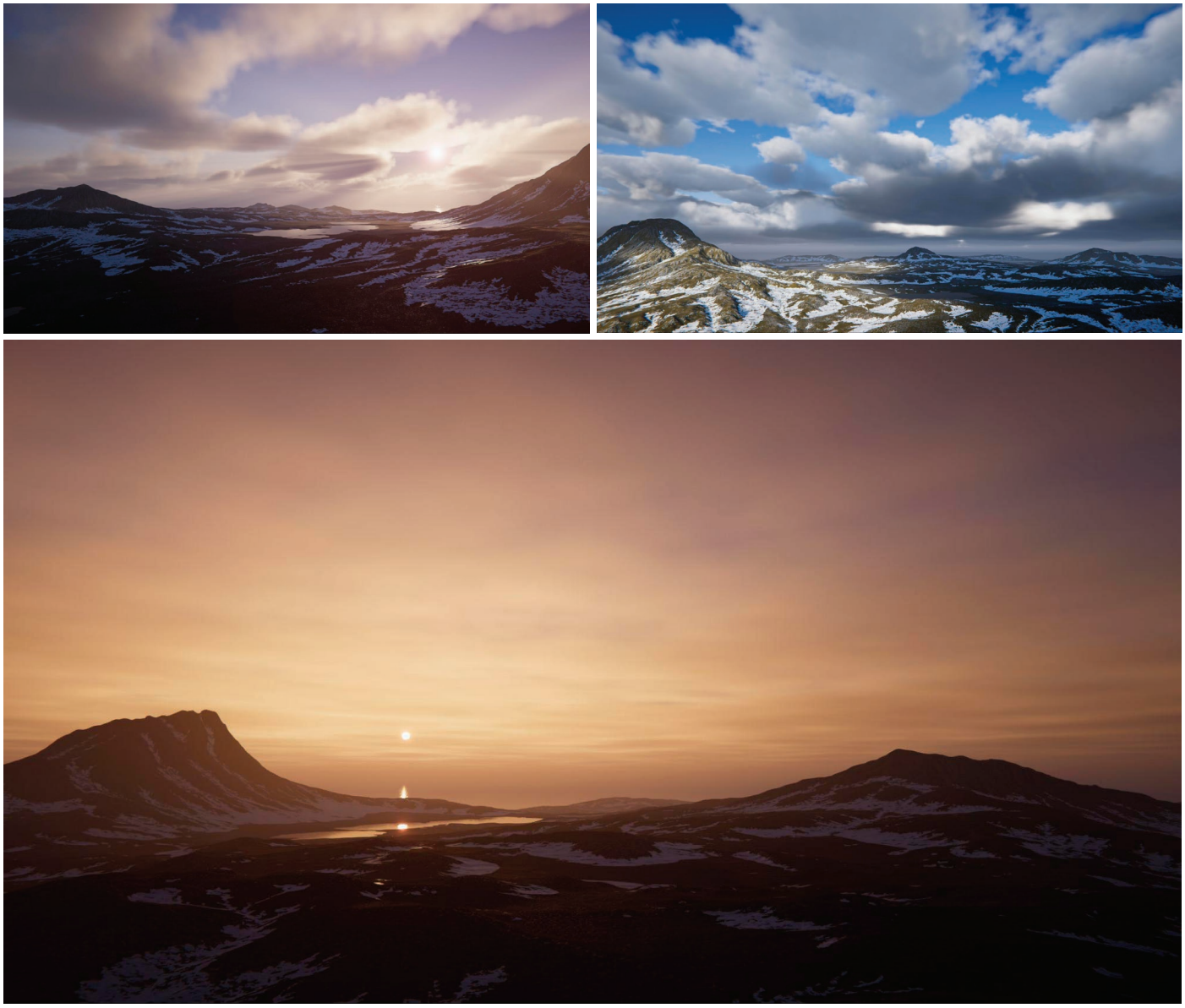
ボリュメトリッククラウド

雲と光を相互作用させる必要がある場合、**Volumetric Clouds** のオーバーライドを使用します。これらは、シャドウをレンダリングし、フォグを受け取り、ボリュメトリックな光軸を作り出すことができます。これらを Cloud レイヤーの雲と組み合わせたり、別々に追加したりしましょう。

ボリュメトリッククラウドのオーバーライドを有効にする方法

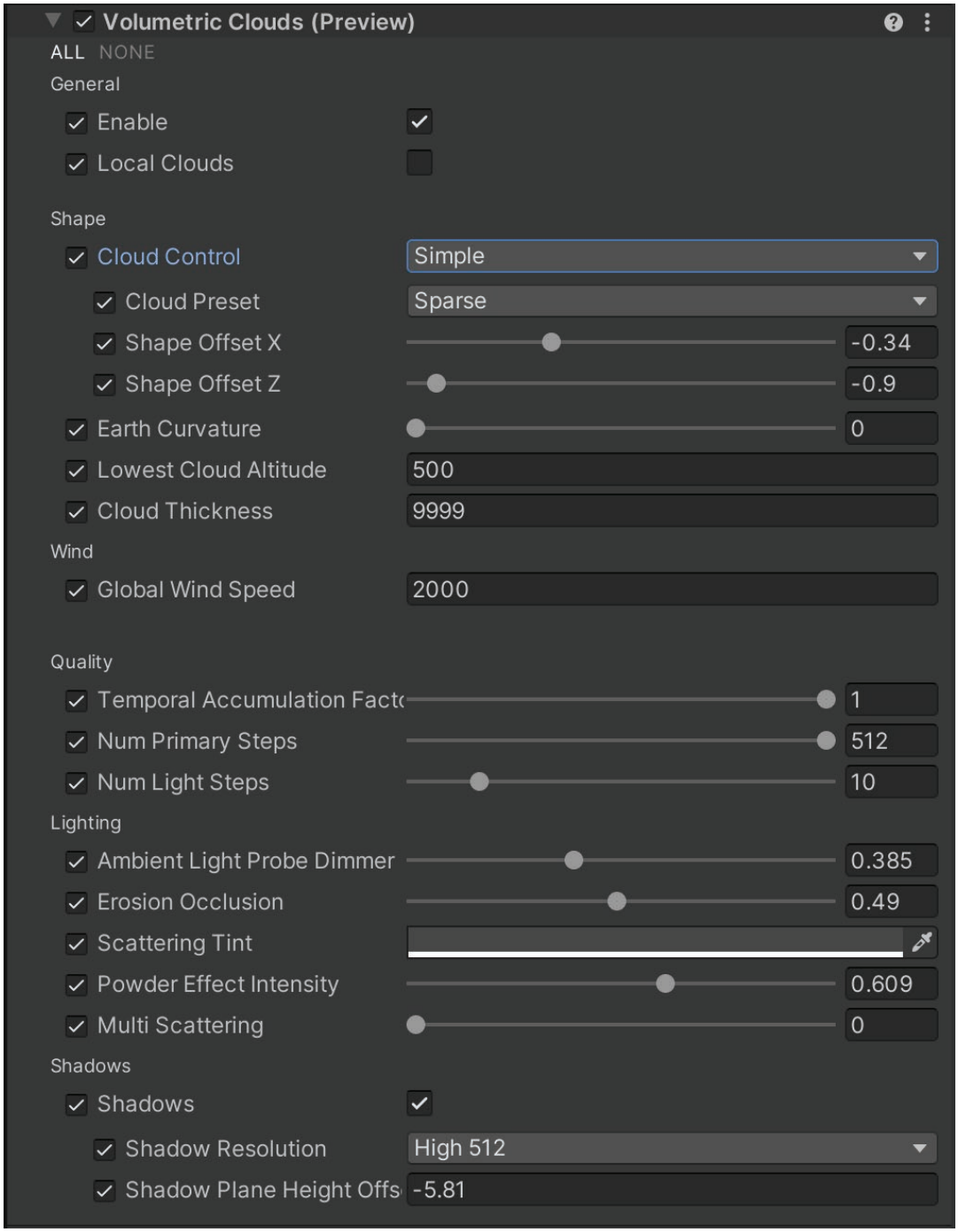
— HDRP アセット内：**Lighting > Volumetric Clouds > Volumetric Clouds** を選択します。

— ローカルまたはグローバルボリューム内：**Volumetric Clouds** のオーバーライドを追加します。



ボリュメトリッククラウドはメインのディレクショナルライトに反応できます。

Advanced および Manual の **Cloud Control** オプションでは、雲の種類ごとにマップを定義できます。



ボリュメトリッククラウドのオーバーライド

Unity 6 のアップデート

Unity 6 での Volumetric Clouds のアップデートは次のとおりです。

- Volumetric Clouds はファークリップ面でクリップされなくなり、カメラの距離に関係なく完全にレンダリングされます。
- 惑星の設定が **Visual Environment** オーバーライドの一部として管理されて個別に地球を設定する必要がなくなり、従来の Local Clouds オプションも不要になりました。
- 大気散乱の改善によって、遠くにある雲のライティングが強化され、以前はパフォーマンスに影響を与えていた、フォグ距離を高く設定する必要性が緩和されました。
- シャドウマップの統合の強化により、セルフシャドウイングが改善され、ボリュメトリッククラウドと組み合わせてレンダリングすることで、より詳細でリアルな雲が形成されるようになりました。
- アップデートされた雲のレイヤーシステムは、明らかに異なる 2 種類の雲のブレンドをサポートし、滑らかな遷移（ボリュームブレンディングによる晴天から曇天へ遷移など）が可能になりました。ボリュームフレームワークのカーブブレンディングにより、ボリュメトリッククラウド間を滑らかに遷移させることができます。
- このアップデートにより、(ルックアップテーブルに依存していた以前のプリセットで見られた) ちらつきや揺らぎなどの視覚的なアーティファクトが軽減されました。
- Volumetric Clouds は、**Visual Environment** の **Rendering Space** 設定に従います。**カメラ空間**を使用して、カメラが雲の中に入るのを防ぎます。

Cloud Layer と Volumetric Clouds のオーバーライドについては、HDRP のドキュメントを参照してください。

Environment Samples

Package Manager の Samples タブにある Environment Samples には、雲のサンプルがいくつか含まれています。これらをたたき台として、さまざまな雲を作ることができます。

このサンプルには、さまざまな **Volumetric Cloud** の設定が含まれています。各プレハブは、雲の形成を定義するためのさまざまな設定を示しています。

Simple モード:事前定義された密度と浸食の設定を使用して、一般的な種類の雲を生成します。パラメーターを調整して、カスタムテクスチャを使用せずに雲の占有率と形状を変更できます。

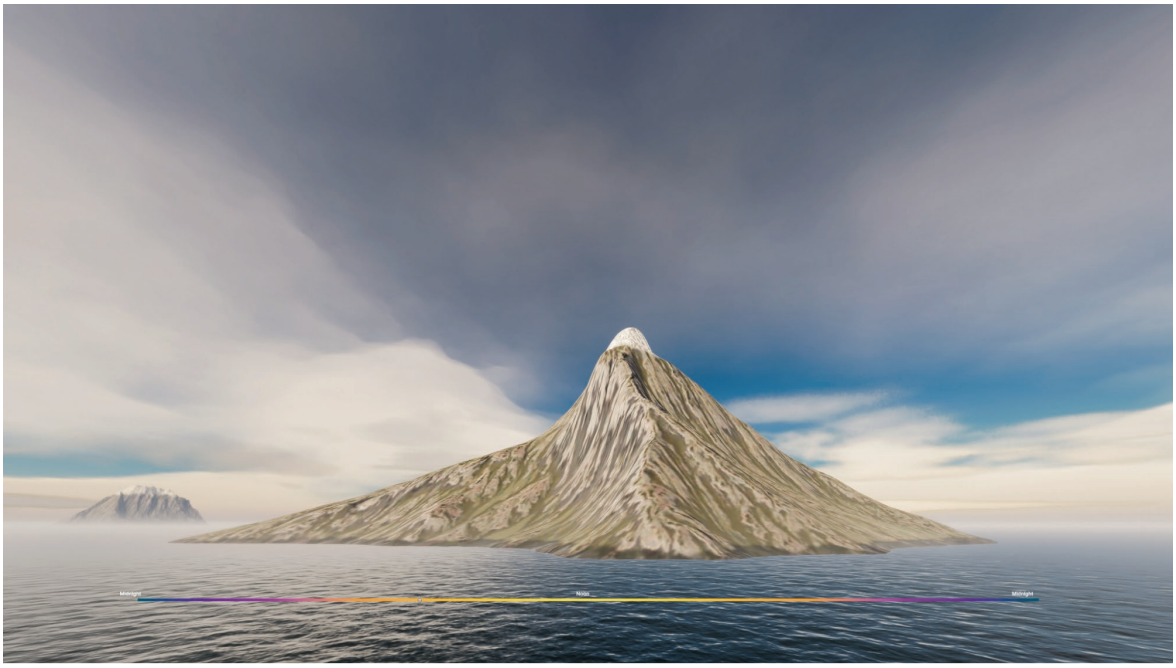
Advanced モード:事前定義された雲の種類 (積雲、高層雲、積乱雲) がワールド空間の特定の位置に配置されます。プロシージャル生成された **Cloud LUT** によって、雲の形状、高度、密度を定義します。

Manual モード:特定の種類の雲で **Cloud LUT** を定義することにより、完全なカスタマイズが可能になります。**Cloud Map** の青チャンネルに含まれるさまざまな色の情報によって、雲の配置が決まります。Inspector UI で、利用可能なプレハブをスクロールして選択できます。

Simple - Clear Skies:雲がまばらなこのプリセットには、ほんのわずかな層雲しかありません。

Simple - Clear Skies の空

Simple - Cloudy:曇りのプリセットは、雲のベールが増え、占有率が高くなっています。

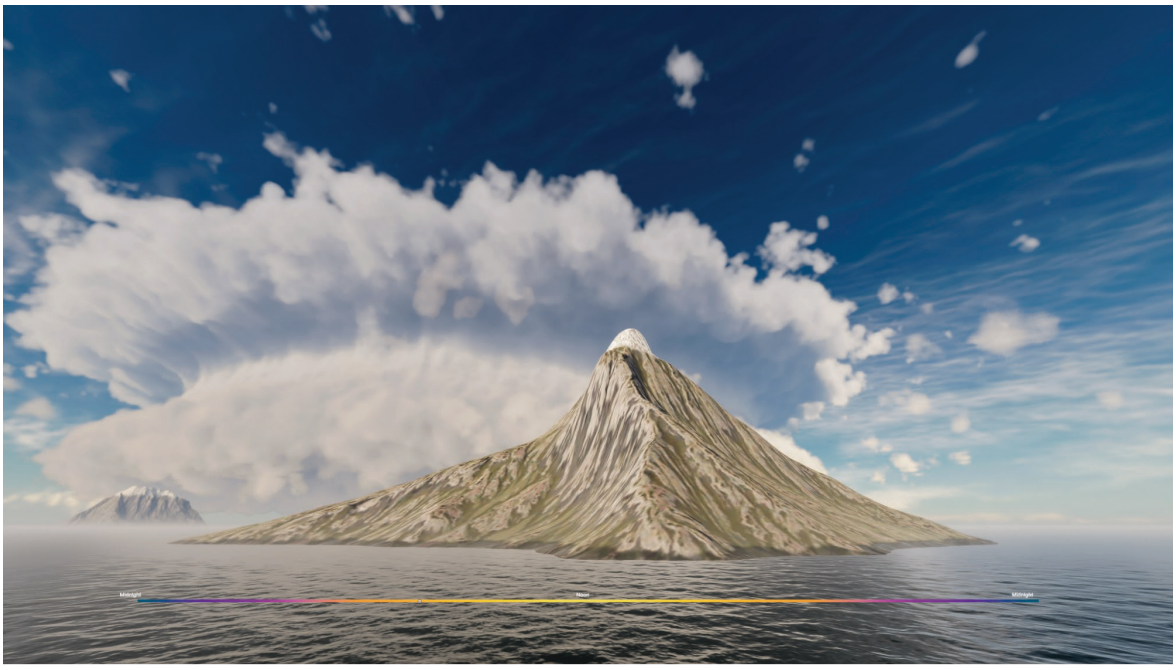


Simple - Cloudy の空

Advanced Clouds:島の上空の異なる高度に、積雲、層雲、積乱雲が配置されます。

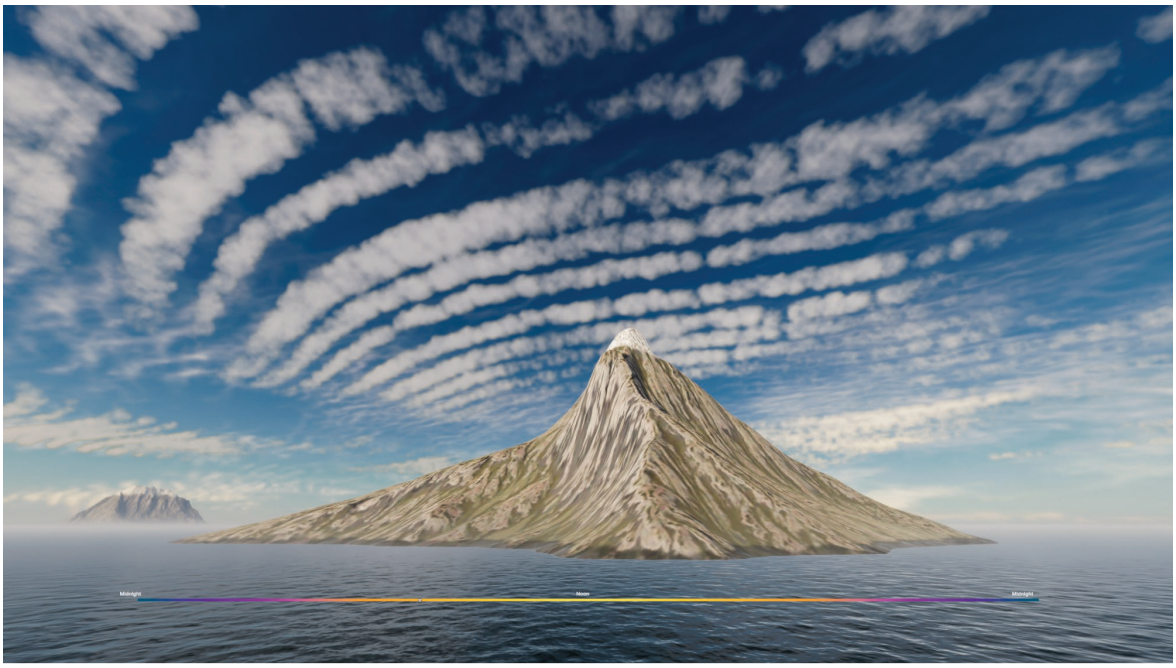
Advanced Clouds の空

Manual - Cumulonimbus:カスタムの **Cloud LUT** を使用して積乱雲を 1 つ配置し、印象的な夕焼け効果を生み出します。



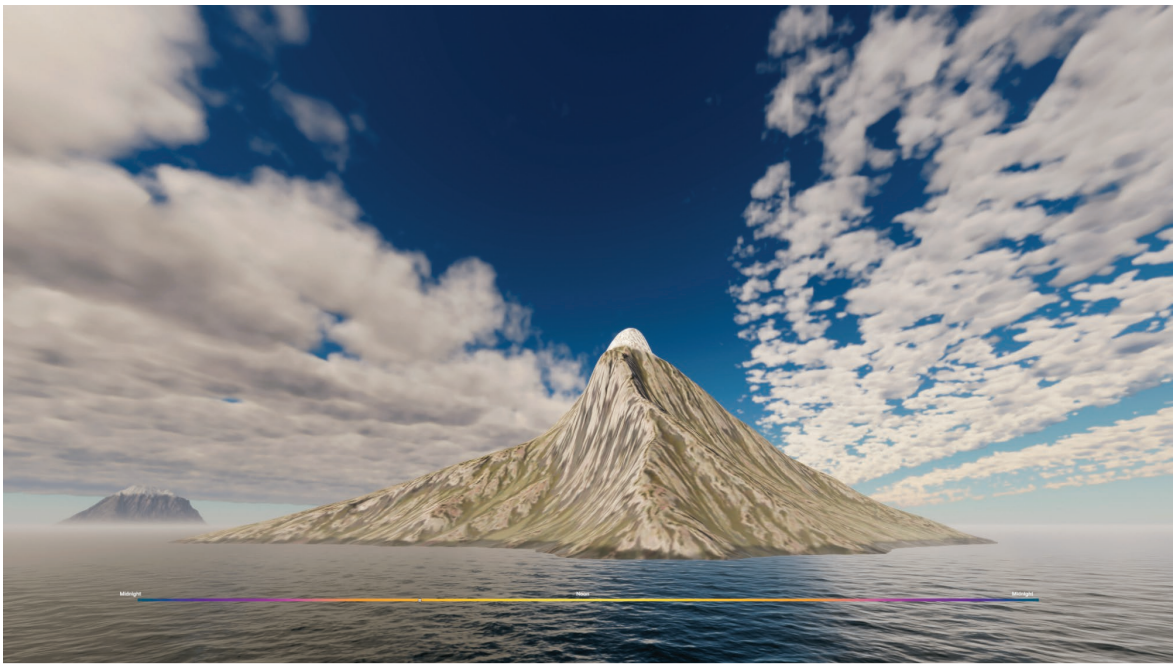
Manual - Cumulonimbus の空

Manual - Cirro Stratus:デフォームされたグラデーションとノイズテクスチャによる巻層雲が作成され、自然な変化が得られます。



Manual - Cirro Stratus の空

Manual - Complete LUT:多様性のためにグラデーションとノイズテクスチャを使用し、**Cloud Map** を介してサンプリングされた複数種の雲を持つ完全な **Cloud LUT** です。



Manual - Complete LUT の空

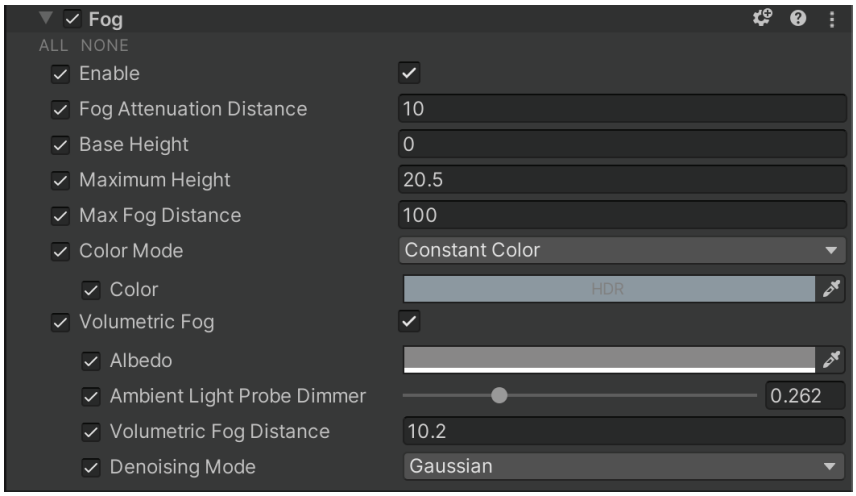
フォグと大気散乱

スモーク、フォグ、もやは、映画的な画面作りの定番のツールです。奥行きや立体感を加えてライティングを演出したり、趣のある雰囲気を創出したりするのに役立ちます。HDRP　では、フォグを使用して同様の効果を実現できます。

不透明度は、オブジェクトとカメラの距離に応じます。フォグを利用すると、カメラのファークリップ面を非表示にして、遠くにあるジオメトリをシーンにブレンドすることもできます。

グローバルフォグ

HDRP では、**Fog**　のオーバーライドとしてグローバルフォグを実装しています。この例では、カメラとの距離とワールド空間の高さに合わせて、フォグが急激にフェードしていきます。



Fog のオーバーライド

シーン内の Volume で Fog のオーバーライドを設定します。**Base Height** では、一定の濃いフォグが上方向に向かうにつれ、薄くなり始める境目を指定します。この値を上回ると、**Maximum Height** に到達するまで、フォグの密度が急激にフェードしていきます。

Base Height と Maximum Height の設定を使用して、低い位置に漂うフォグを表現します。

同様に、**Fog Attenuation Distance** と **Max Fog Distance** では、カメラから距離が離れるにつれ、フォグをどのようにフェードさせるのかを設定します。**Color Mode** は、**Constant Color** と既存の **Sky Color** のどちらかに設定します。

Fog Attenuation Distance の設定は、フォグがどのように背景にフェードインするかを変更します (表示されているのはボリュメトリックフォグ)。

Volumetric Fog を有効にすると、大気散乱のシミュレーションが行われます。Lighting の **Frame Settings** (カメラの下または HDRP デフォルト設定) で、**Fog** と **Volumetrics** をオンにします。また、HDRP パイプラインアセットで**ボリュメトリックフォグ**も有効にします。

Volumetric Fog Distance には、カメラのニアクリップ面からボリュメトリックライティングバッファの背部までの距離 (メートル単位) を設定します。これにより、大気中に浮遊物質が満たされ、範囲内のゲームオブジェクトが部分的に隠れます。

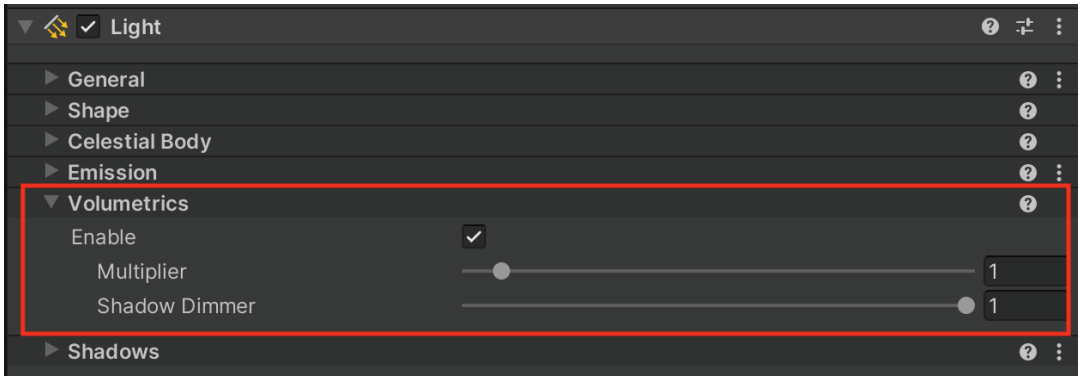
ボリュメトリックフォグは、正確に前景のジオメトリを保持します。

ボリュメトリックライティング

ボリュメトリックライティング によって、日没時の雲の背後や木々の間から射す 光芒 など、印象的な太陽光線をシミュレートしてレンダリングできます。

それぞれの Light コンポーネント (エリアライト以外) に、**Volumetrics** グループがあります。**Enable** をオンにしてから、**Multiplier** と **Shadow Dimmer** を設定します。**Real-time** または **Mixed Mode** のライトを使用すると、Volumetric Fog 内に "薄明光線" が生成されます。

Multiplier で強度を調整し、Shadow Dimmer を使用して、シャドウを投影するサーフェスがライトを遮る具合を設定します。



Light コンポーネントの Volumetrics

光の軸はボリュメトリックフォグ内にしか現れないので、フォグの Base Height と Maximum Height を調整することで減衰を制御できます。

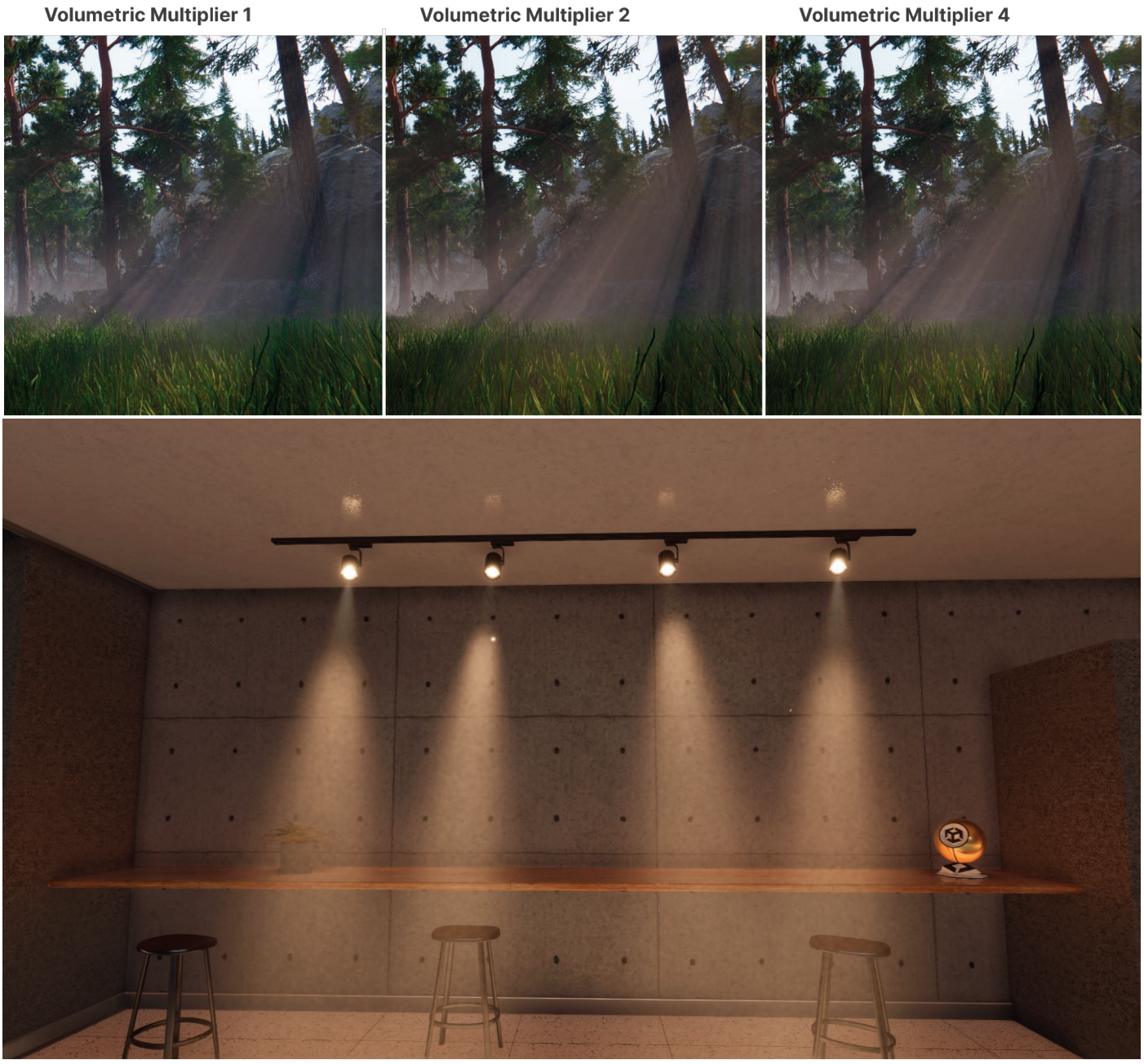
ボリュメトリックライティングやシャドウについてのヒント

ボリュメトリックライティングとシャドウを使ったドラマチックな演出が必要ですか？これらのヒントを考慮してください。

- 建物や木のような光を遮るオブジェクトは影を作り、これらのオブジェクトの縁では光軸がより見やすくなります。
- 光軸の見え方は、光源に対するカメラの位置と角度に依存します。カメラが光の向きとほぼ一直線になると、光線がより目立つようになります。
- 光の明度や色は、ゴッドレイの見た目を左右します。強度を上げて特定の色を選択することで、光線をより際立たせることが可能です。
- シーン内のボリュメトリックフォグ、埃、その他のパーティクルは光を散乱させ、光線をより強調します。VFX Graph はそれぞれの光軸にディテールを追加し、より活き活きとさせることができます。

ディレクショナルライトのボリュメトリックオプションを有効にすると、ボリュメトリックフォグ内に光軸が表示されます。

これらの例では、ディレクショナルライトとスポットライトに対する Volumetric Multiplier の効果を見て取れます。

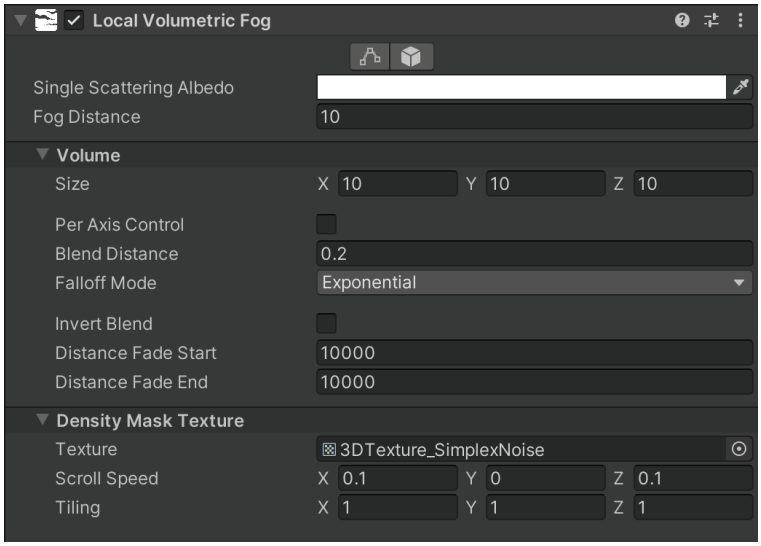


HDRP サンプルの天井のスポットライトに適用されたボリュメトリックライティング

ローカルボリュメトリックフォグ

HDRP では、Fog のオーバーライドよりも細かいフォグエフェクトが必要な場合に、**Local Volumetric Fog** (ローカルボリュメトリックフォグ) コンポーネント (古いバージョンの HDRP での呼称は Density Volume) を利用できます。

これは Volume システムの外にある別個のコンポーネントです。メニューから **Local Volumetric Fog** のゲームオブジェクトを作成 (**GameObject > Rendering > Local Volumetric Fog**) するか、Hierarchy で右クリックします (**Rendering > Local Volumetric Fog**)。



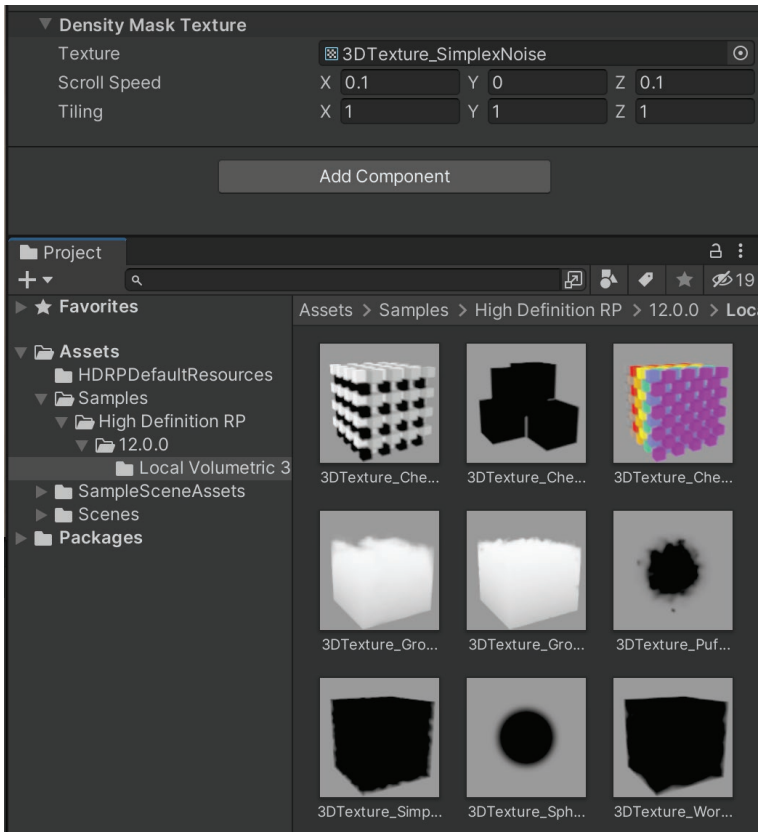
Local Volumetric Fog コンポーネント

これにより、フォグが満ちたバウンディングボックスが生成されます。サイズ、軸のコントロール、ブレンディング/フェーディングのオプションを調整します。

ローカルボリュメトリックフォグはバウンディングボックスに表示されます。

デフォルトではフォグは一様ですが、3D テクスチャを **Density Mask Texture** サブセクションの **Texture** フィールドに適用できます。これにより、ユーザーはより柔軟にフォグの外観をコントロールできます。Package Manager の **Local Volumetric 3D Texture Samples** からサンプルをダウンロードするか、ドキュメントの手順 に従って独自の Density Mask を作成します。

アニメーションの **Scroll Speed** を設定し、**Tiling** を調整します。これで、ボリュメトリックフォグが徐々にシーン内に描画されます。



Local Volumetric 3D Texture Samples の Density Mask Texture

注意:HDRP では、ローカルボリュメトリックフォグをボクセル化してパフォーマンスを高めます。ただし、ボクセル化によって外観が非常に粗くなる場合があります。エイリアシングを低減するためには、Density Mask Texture を使用し、Blend Distance を上げてフォグのエッジを滑らかにします。HDRP パイプラインアセットで、最大 256×256×256 のローカルボリュメトリック解像度を有効化でき、より精密で大規模な効果を生み出すことが可能になります。

HDRP 水システム

水は現実世界ではどこにでもあるものですが、Unity　の水システムによって、ゲームの世界にもこれまで以上に簡単に組み込むことができるようになりました。プロジェクトで、人里離れた熱帯のラグーンや、危険な氷に覆われたフィヨルドのデザインを行っていますか？HDRP 環境への水要素の追加は、ほんの数クリックで完了します。

水システムを使用し、海、川、湖を追加しましょう。

水システムの主な特徴の一部を以下に示します。

- **物理ベースの水レンダリング:**水システムには、滑らかさ、屈折、光散乱のプロパティを調整できる物理ベースの水シェーダーが備わっています。
- **水の動き**:水システムは、うねり、攪拌、波紋、流れを含む水面の変形のシミュレーションを行います。
- **水中効果**:カメラが水面下に潜ると、水中レンダリングが水の物理的特性を考慮し、光の屈折と吸収を再現します。
- **Foam:**水面シミュレーションと風速に基づいて、泡を自動的に生成できます。ローカルの泡ジェネレーターによって、ボートの航跡などの小さなエリアにおける白波効果のシミュレーションを行うこともできます。
- **インテグレーション:**マスク、デカール、デフォーマーなど、アーティストにとって使いやすいコンポーネントを活用し、水面を周囲の小道具や地形につなげることができます。
- **パフォーマンス:**このシステムはさまざまなプラットフォームに最適化されており、可視化のための複数のデバッグモードが用意されています。

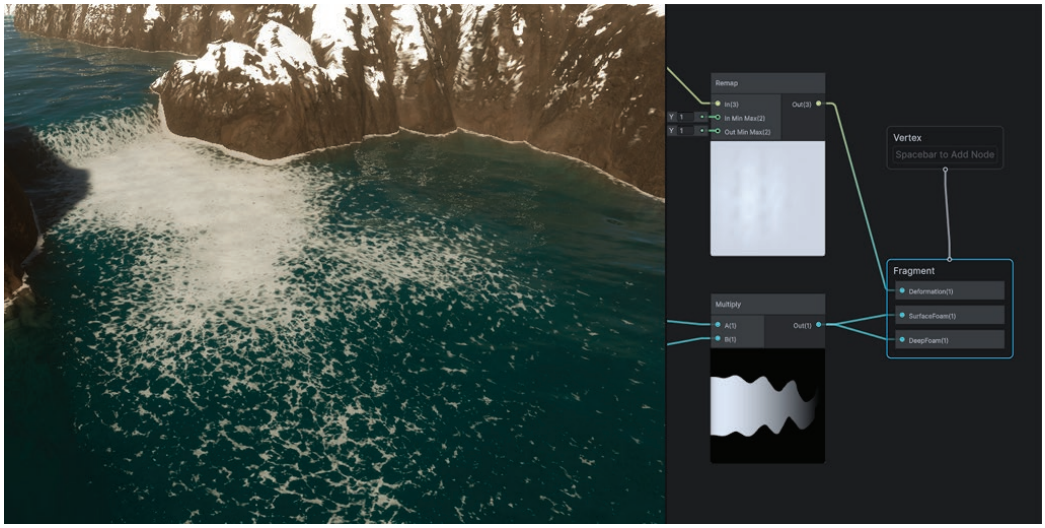
水システムは柔軟かつカスタマイズ可能です。

Unity 6.1 の新機能

Unity 6.1 (HDRP 17) には、水システムと併用するための新機能がいくつかあり、初めて使用する場合に推奨されています。

- **水面の Transform サポートの改善:**Infinite Oceans と Instanced Quad サーフェスで平行移動、回転、負のスケーリングがサポートされ、水面の反転が可能になりました。
- **水のデカール ShaderGraph ターゲット:**新しい ShaderGraph ターゲットにより、水のデフォーメーションや泡をアトラスに直接出力できるようになりました。これにより、カスタムレンダーテクスチャが不要になります。泡のアニメーションが水の方向に追従するようになり、より動的なエフェクトが得られます。
- **水中レンダリングの強化:**ボリュメトリックフォグが薄明光線とライトシャフトに対応するようになり、水中環境の没入感が高まりました。

— **ShaderGraph の新しい Camera Height ノード:**新しい ShaderGraph ノードでは、水面に対するカメラの高さをサンプリングできます。これにより、カメラレンズの水滴などのエフェクトが可能になります (下の **水中** サンプルシーンを参照)。



新しい ShaderGraph ターゲットにより、泡の出力が可能になります。

Unity 6.1 以降があれば、**Cave** サンプルシーンや **RollingWave** サンプルシーンを確認することもできます。

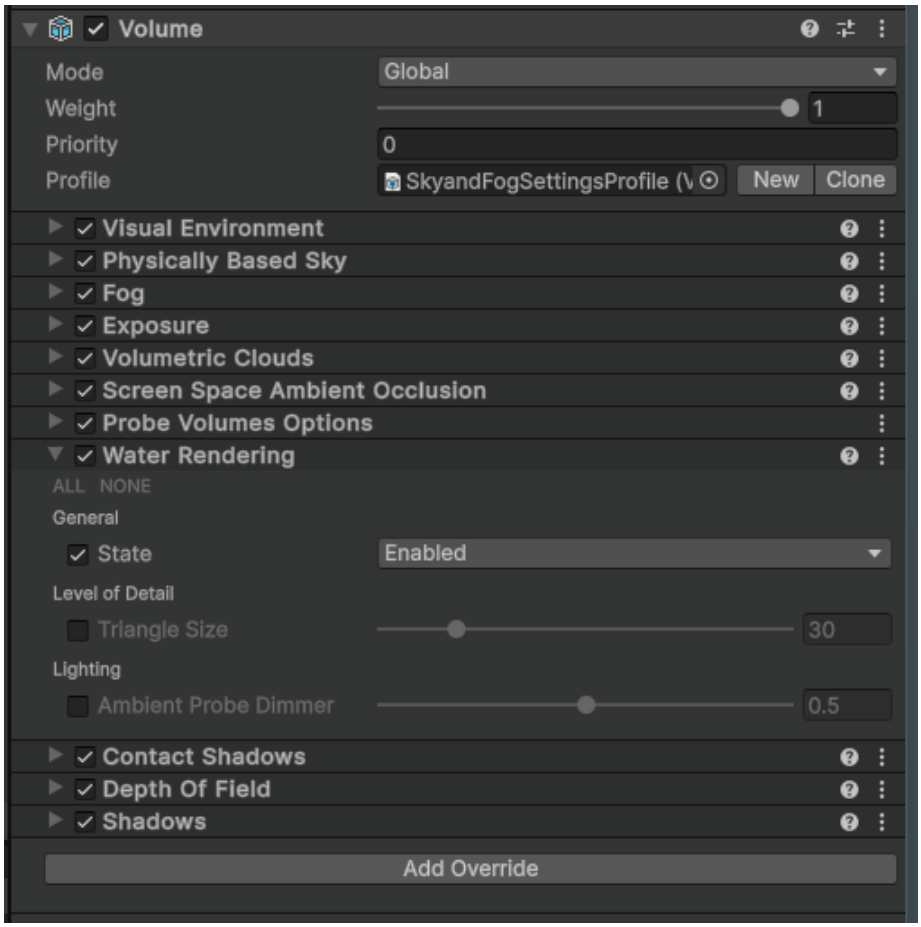
水システムの使用を開始する

プロジェクトを水システムに対応させる方法:

— 各レンダーパイプラインアセット (水を必要とする品質レベルごと) に対して水をアクティベートし、設定します。

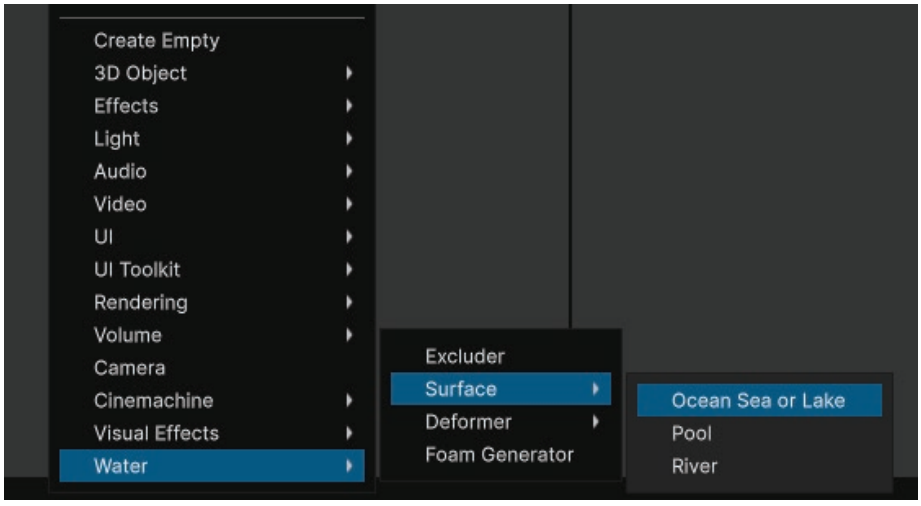
— カメラの Frame Settings で水を有効にします (**Edit > Project Settings > Graphics > HDRP Global Settings**)。

— シーン内の **Water Rendering** ボリュームオーバーライドを使用し、カメラの位置に基づいて水のレンダリングがアクティブになる場所を制御します。



Volume オーバーライドで Water Rendering を有効化します。

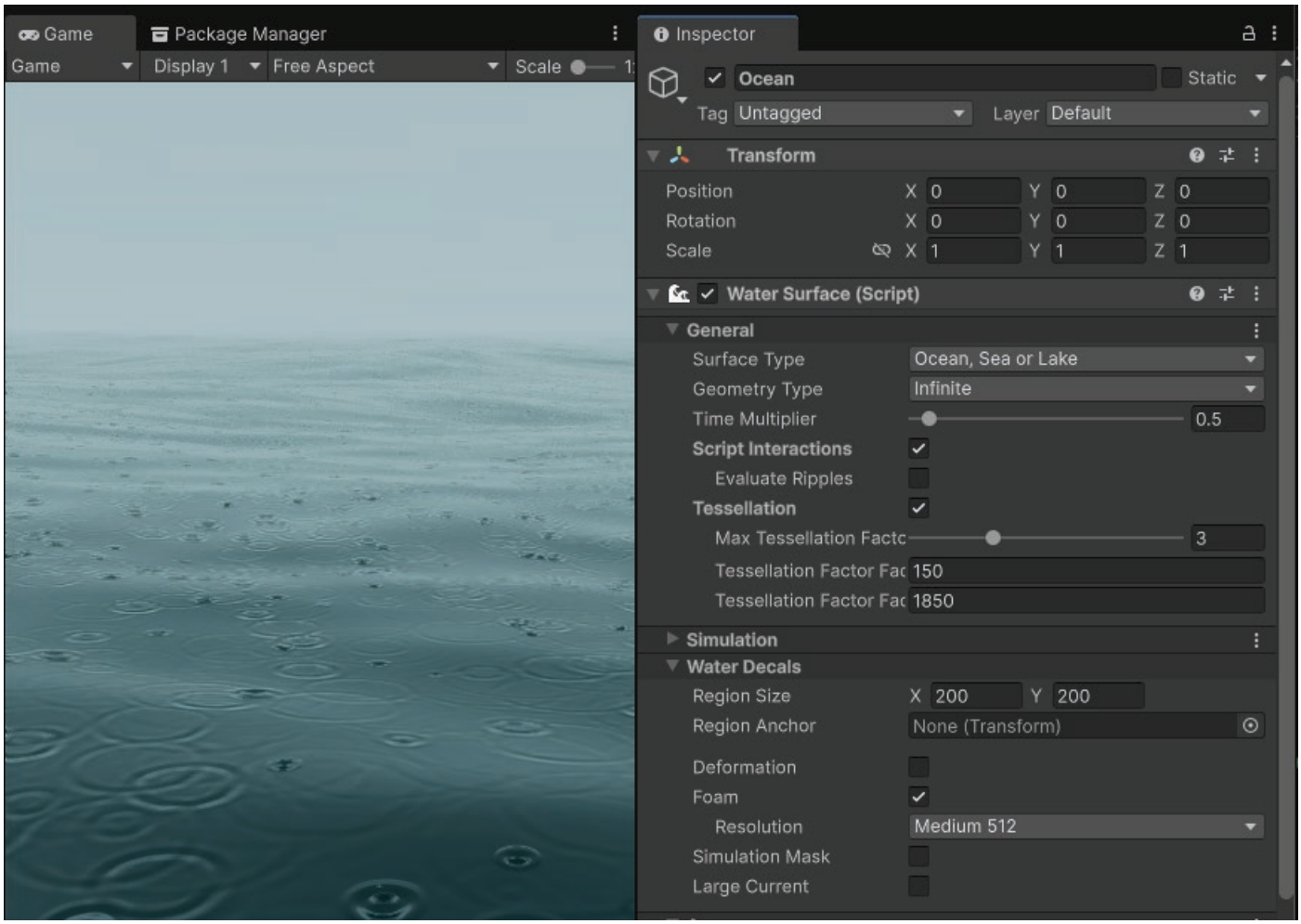
水が有効になったら、ゲームオブジェクトメニューから設定済みの水域を追加できます。HDRP には 3 種類の水面があります。プール、川、海です。



ゲームオブジェクトメニューから水オブジェクトを追加します。

水面コンポーネント

水面オブジェクトには、水の全般的なパラメーターを制御する **Water Surface** (水面) コンポーネントスクリプトがあります。



水面コンポーネント

これらのパラメーターは、水面のシミュレーションとレンダリングを設定します。

— **General:**水域の全体的な種類 (例: 海洋/海/湖、川、プール) と、水面のレンダリングに使用するジオメトリ (例: クアッド、カスタム、InstancedQuads、Infinite) を定義します。

— **Simulation:**風と月が水面に与える影響のエミュレーションを行い、波のパターンと波紋の形成を制御します

— **Water Decals:**水デカールの中心とサイズに影響します (下記参照)。この設定を使用して、デフォーメーション、泡、シミュレーションマスク、水流を切り替えたり、テクスチャの解像度を設定したりできます

- **Appearance:**水の色、滑らかさ、屈折、光の散乱を決定します。コースティクスと特殊な水中設定で、物理ベースのシェーディングを強化します
- **泡:**波の頂上、水中のオブジェクトの周辺、海岸線における泡の外観と挙動を制御します
- **Miscellaneous:**Rendering Layer Mask とデバッグモードを制御します

水デカール (Unity 6.1)

Unity 6.1 では、水デカールを使用して水の効果を細かく制御できます。水デカールは、ワールド空間に適用されるShader Graph マスタースタック です。これにはいくつかのアプリケーションがあります。

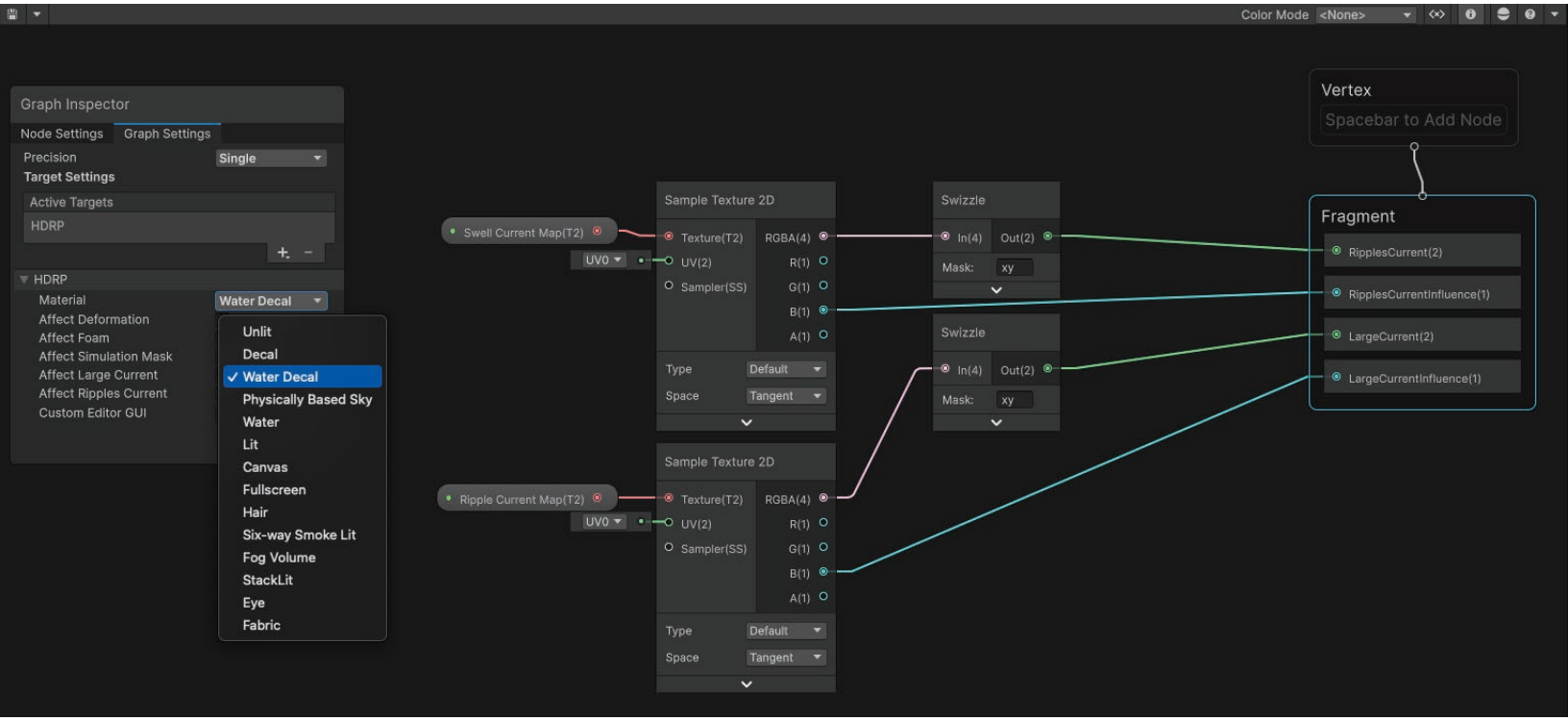
- **Currents:**局所的な水の流れを定義できます。
- **Deformation:**水面の高さとディスプレイスメントを変更します。
- **Foam:**水がオブジェクトや地形と相互作用する場所に、ホワイトウォーター（白く泡立った水）のエフェクトを加えます。

水デカールを作成するには、**Create** (作成) または **GameObject** メニューに移動し、**Water > Water Decal** を選択します。HDRP　によりゲームオブジェクトがシーンに追加されます。エフェクトを適用したい領域にデカールを移動させます。

Water Surface コンポーネントの **Water Decal** プロパティの設定を使用して、テクスチャの解像度を変更したり、特定の

水デカールの Shader Graph を編集し、デカールが水にどのように影響するかを決定します。Enable Mask And Current Water Decals を有効にすると、Graph Inspector で以下の水機能を利用できます。

- HorizontalDeformation
- SimulationMask
- SimulationFoamMask
- LargeCurrent
- LargeCurrentInfluence
- RipplesCurrent
- RipplesCurrentInfluence



水デカールのエフェクトは Shader Graph マスタースタックを使用します。

デフォルトでは、水デカールの領域はカメラに固定されています。ゲームオブジェクトに固定することもできます。水デカールの使用方法については、Water パッケージサンプル の **RollingWave** シーンを参照してください。

Unity 6 の水サンプル

HDRP の水システムを詳しく見る最善の方法は、さまざまなユースケースを示すサンプルシーンを使用することです。Package Manager で、**High Definition Render Pipeline (HDRP)** パッケージを選択し、**Samples** タブから **Water Samples** をインストールします。

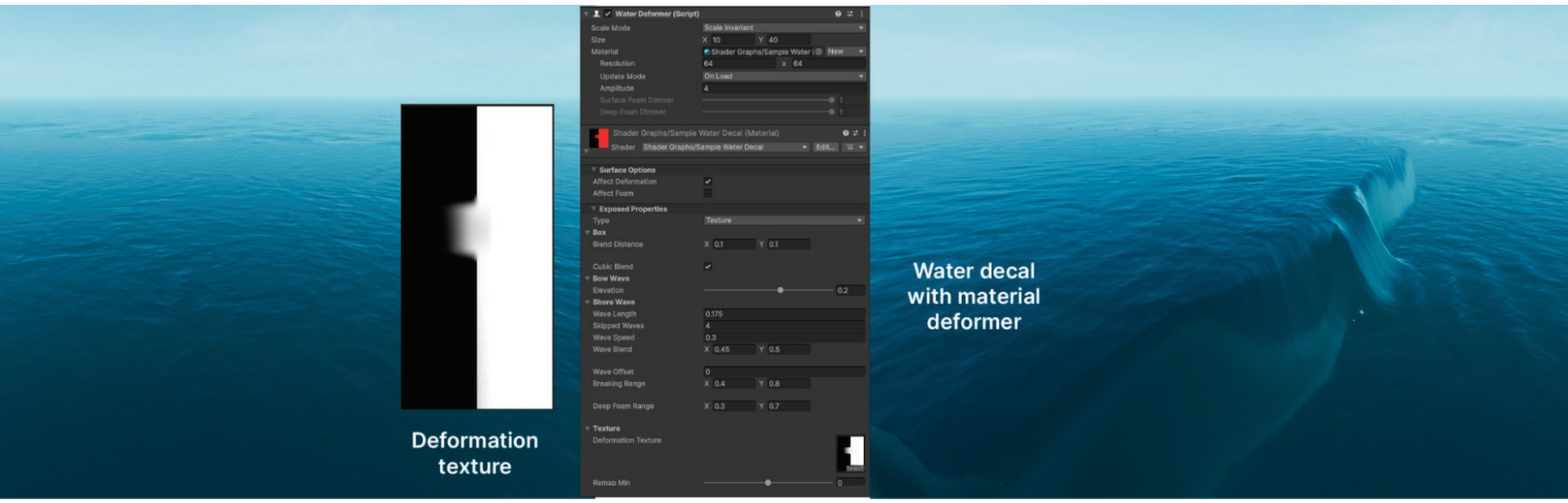
Unity 6.1 でアップデートされたこれらのサンプルは、リアルな水をプロジェクトに統合するための出発点となります。

氷河

氷河（Glacier）のサンプルは、川から海のような大きな水域に遷移する方法を示しています。Unity 6.1 でアップデートされ、**Mask and Current Water Decal** ワークフローを使用するようになりました (**Project Settings > Graphics** で有効化)。

水デカールは、従来の水デフォーマーシステムに代わって、水のすべてのデフォーメーションを処理するようになりました。

マテリアルデフォーマーを備えた水デカールが中央の滝を形作っています。デフォーメーションテクスチャに基づいて水の高さを上げています。一方、カスタムレンダーテクスチャを持つ Decal Projector コンポーネント は、落ちる水をシミュレートするアニメーションテクスチャを作成します。



水面を水デカールでデフォームさせます。

Water Simulation Decal は、川沿いに海のうねりが侵入するのを防ぎ、川の始点で泡を発生させます。一方、**Water Current Decal** は、川から海への水の流れを導きます。

氷河のサンプルは川と海をつないでいます。

ピクセル深度の値を比較して、川岸に沿って泡を追加しています。川床と水面の差が小さい場合に泡を出します。

川の端に泡が立ちます。

コースティクスの効果は、光の屈折をシミュレートすることで水が干渉し合う表現のリアリティを高めます。氷壁のコースティクスは、GetCausticsBuffer 関数を通じてリアルタイムで更新される Decal Projector コンポーネントを使用して投影されます。

川床では、コースティクスは Water Surface コンポーネント上で直接有効にされますが、水流マップの影響を受けないため、水の動きに関係なく一貫したライティング効果が得られます。

コースティクスと氷の塊

浮かんでいる氷の塊は、水面に対する浮力をシミュレートする FloatingIceberg スクリプトを使用して水流とともに移動します。それぞれの塊には Foam Generator も含まれており、漂うにつれて後ろに泡のエフェクトが生成されます。

島

島 (Island) のシーンでは、開放水域と岸や浮遊物との相互作用を実際に表現しています。1 つの水デカールで複数の目的に対応しています。

— 水デカールは、Affect Simulation Mask オプションを使用して、水のうねりや渦巻きの一部をマスクします。

— 同じシミュレーションマスクテクスチャが Water Decal Shader Graph で使用され、泡をマスクしています。

— Current Map テクスチャは、**Affect Large Current** オプションを使用して水面が流れる方向を変えます。

島のサンプルは、岸辺の波頭を示しています。

岸辺での臨場感を高めるために、寄せる波は海岸線で砕ける様子をシミュレートします。三人称視点のカプセルコントローラーが波打ち際を動き回り、水面をかき乱す動的な航跡を作り出します。一方、**Water Excluder** ゲームオブジェクトは、シーン内のボートなど水に浮かんでいるオブジェクトの内部に水が表示されないようにします。

三人称視点のカプセルコントローラーが波打ち際を動きます。

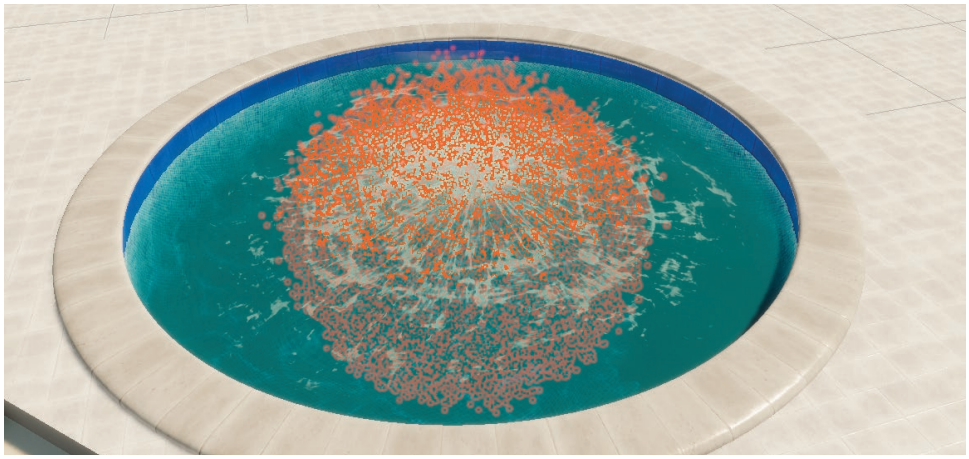
プール

プール (Pool) のシーンでは、浮力の物理演算と緻密なリフレクションが表現されています。ビーチボールをプールに落として、その衝撃が水面に沿って局所的なデフォーメーションと水しぶきのエフェクトを引き起こす様子を確認してください。

ビーチボールは最終的には、質量と体積を考慮したカスタム浮力物理演算を使用して表面に浮かびます。

プールのサンプルでは浮力体オブジェクトが浮いています。

水面のさまざまな反射を表現するために、HDRP アセットで **Screen Space Reflection on Transparent** を有効にしています。また、Caustics 設定の Directional Shadow が、プールの水中の影になった部分でコースティクスの効果をどのように暗くしているかにも注目してください。



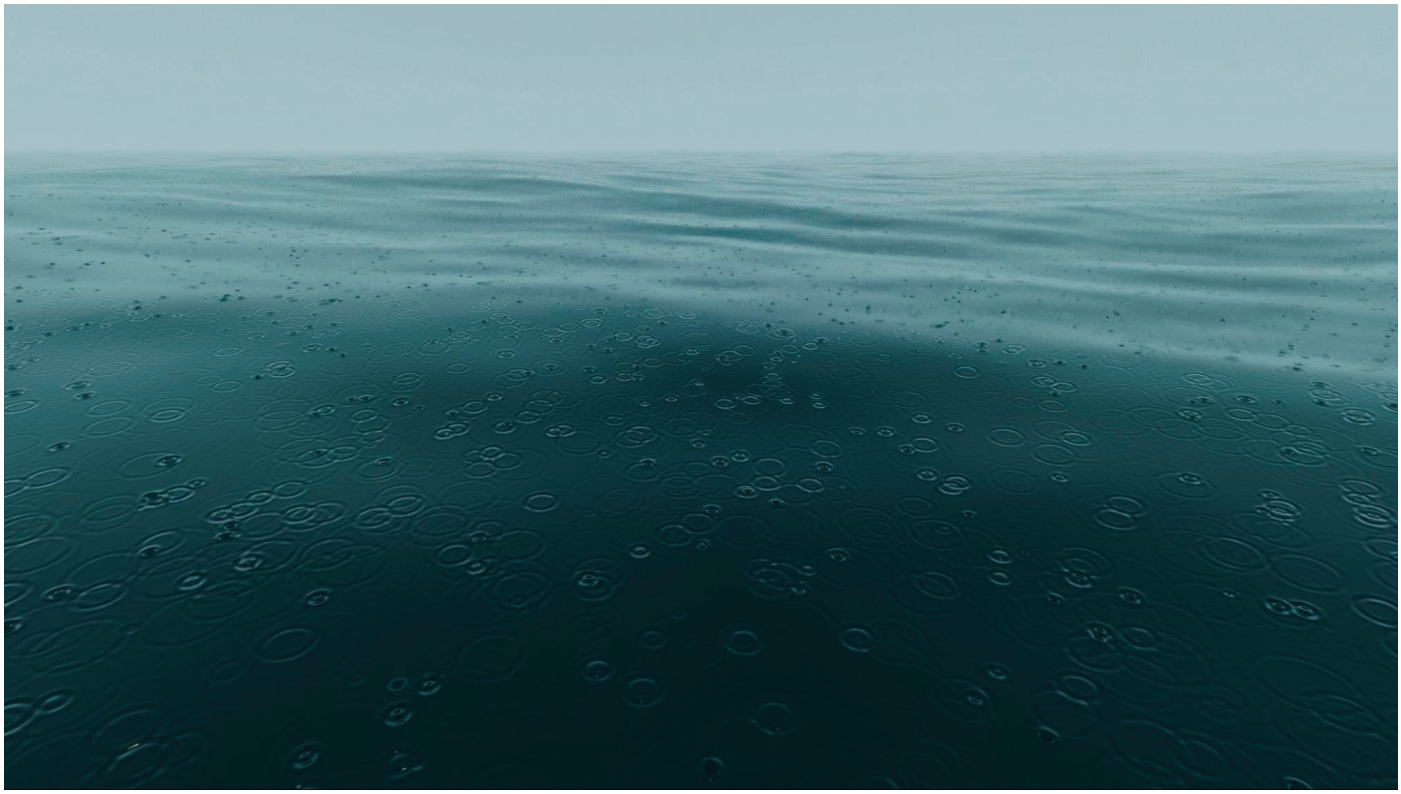
VFX Graph とデカールが泡風呂の泡を作り出します。

一方、泡風呂のシミュレーションでは Bubbles ゲームオブジェクトを使用しています。これにより、Decal Projector とアタッチされた VFX Graph を使用して、動く水のエフェクトが作成されます。

雨

雨 (Rain) のシーンは、雨が海面と相互作用する様子をシミュレートしています。カスタムレンダーテクスチャは、雨による波紋をリアルタイムで生成します。Shader Graph のマテリアルは、水面に波紋を動的に適用します。

調整可能な設定を使用して、雨滴の速度、方向、強度を制御します。



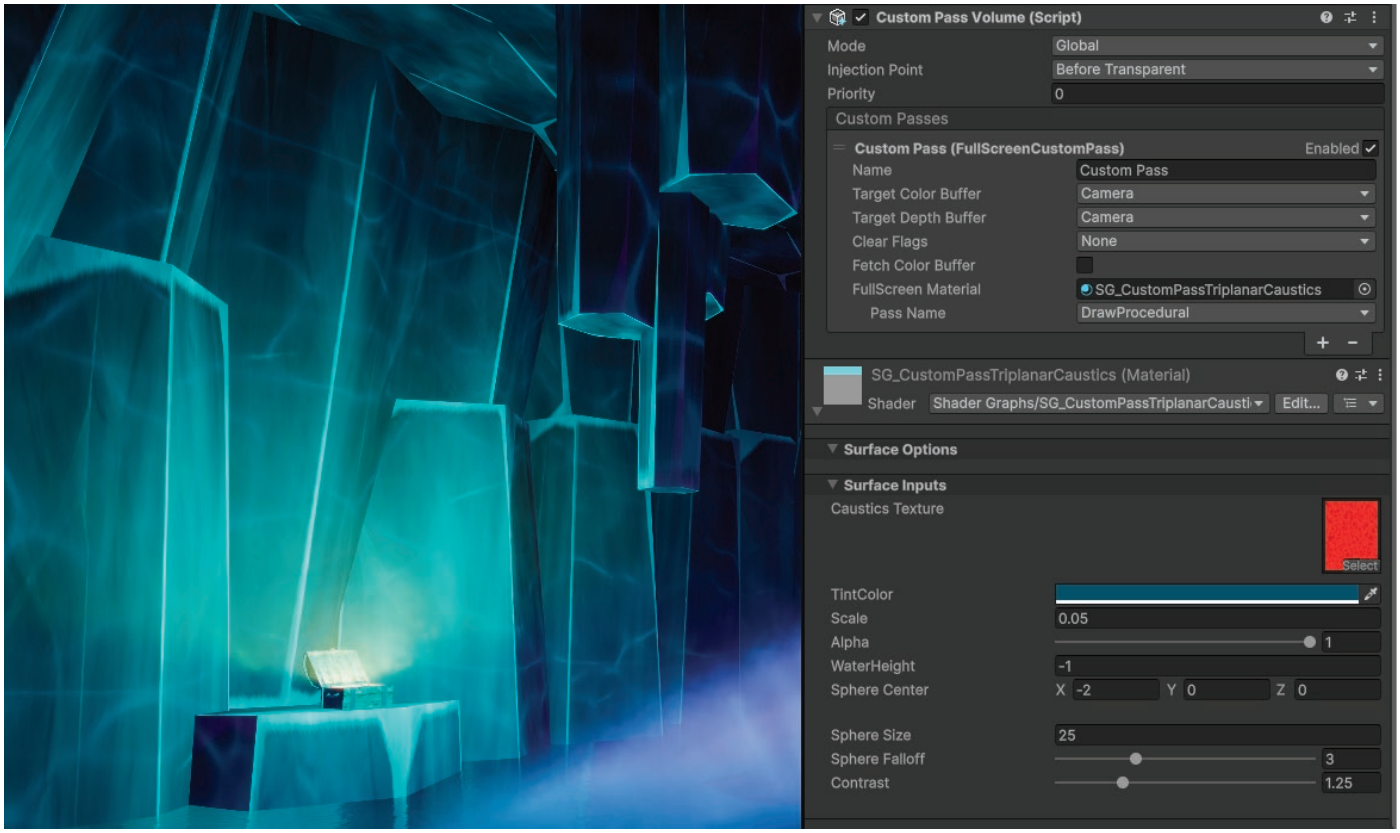
カスタムレンダーテクスチャは雨による波紋を生成します。

以降のサンプルは、Unity 6.1 以降に固有の新しいワークフローを紹介するため、このバージョンの Unity が必要です。

洞窟

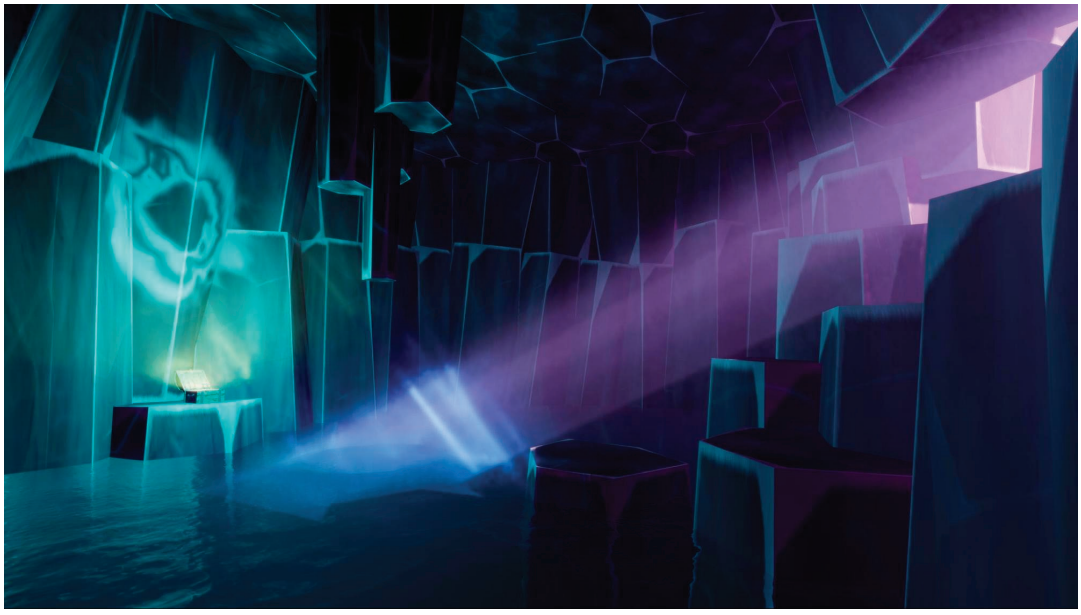
洞窟 (Cave) のシーンでは、Unity 6.1 以降で使用可能なコースティクステクスチャのさまざまな活用方法を示しています。ここでは、洞窟環境で見られる可能性のあるライティングエフェクトを適用しています。光が水中でどのように曲がり収束するかをシミュレートして、周囲の表面に揺らめくパターンを作成できます。

CustomPassは、フルスクリーンマテリアルとShaderGraphを使用して、洞窟の壁と天井にコースティクスライティングの効果を投影します。シェーダーは水面からコースティクステクスチャをサンプリングします。**SG_CustomPassTriplanarCaustics** マテリアル設定を調整して、外観を微調整できます。



Custom Pass はコースティクスライティングの効果を投影します。

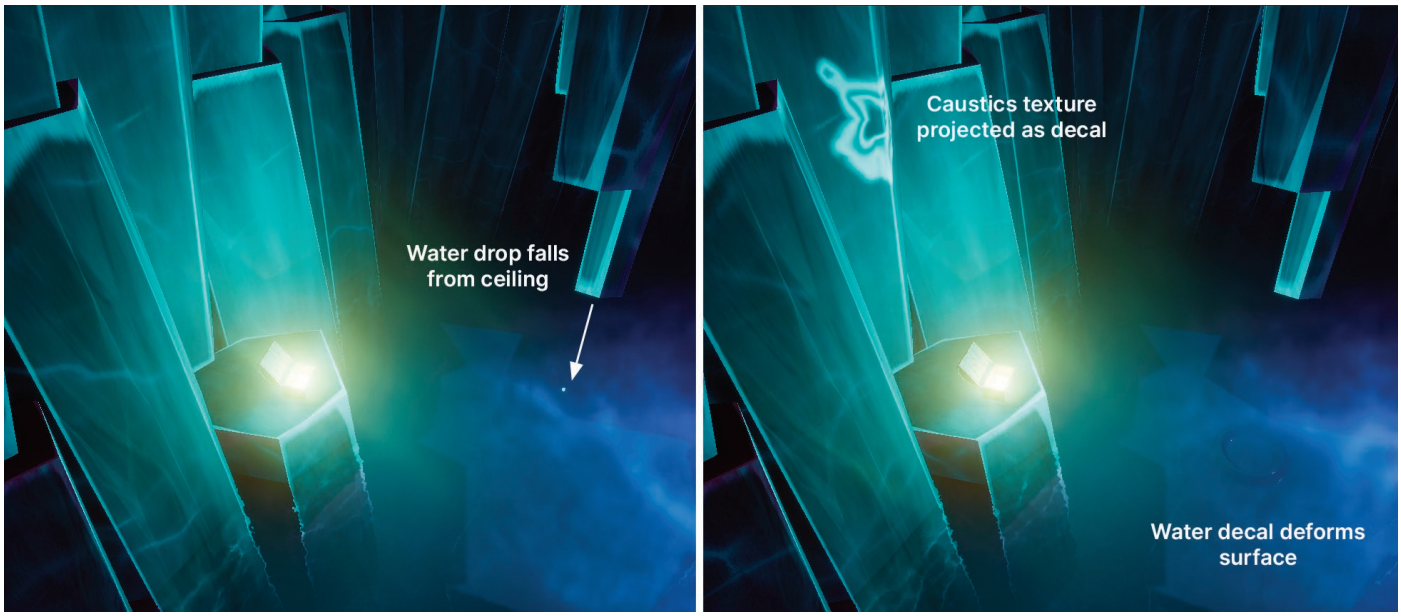
Shader Graph を編集し、各軸に沿ってコースティクステクスチャがどのように投影されるかを確認します。シェーダーは、距離に応じて効果を減衰させ、また平らで上向きの面にコースティクスが現れるのを避けるために -Y 方向でも効果を減衰させます。また、すでに水システムが処理している水面下にコースティクスが表示されることも防ぎます。



洞窟のシーンでは、コースティクスの投影方法とボリュメトリックフォグを紹介します。

光の軸については、カスタムマテリアルを使用した **Local Volumetric Fog** (ローカルボリュメトリックフォグ) が水面で反射する光のパターンをシミュレートします。Shader Graph は、Water Surface コンポーネントからコースティクステクスチャをサンプリングし、柔らかい空気感のあるエフェクトを作成します。

最後に、アニメーション化された水滴が天井から落ち、水面に波紋を作り出します。Decal Projector は、周囲の壁に現われるコースティクステクスチャをシミュートします。



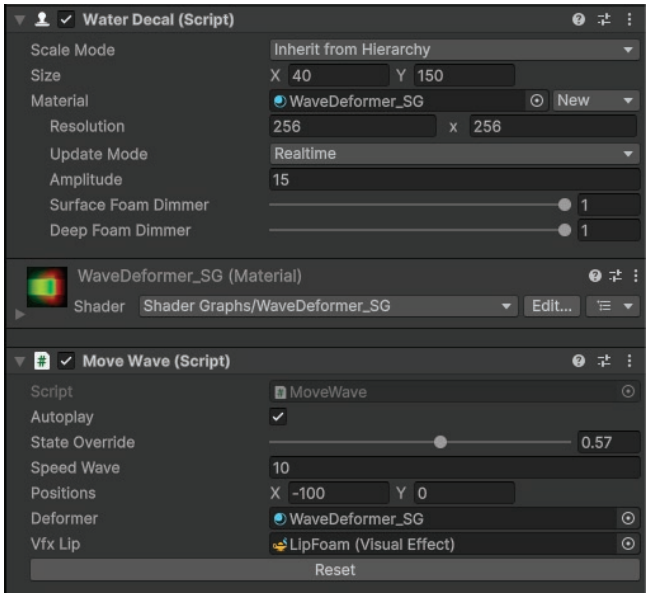
水滴は表面をデフォームさせ、壁にコースティクスを投影します。

うねる波

うねる波 (Rollong Wave) のシーンは、Unity 6.1 の水デカールデフォーメーションとデフォーメーションテクスチャを使用して、大きくうねる波を表現しています。Shader Graph とカスタム移動スクリプトを使用して波を作成します。

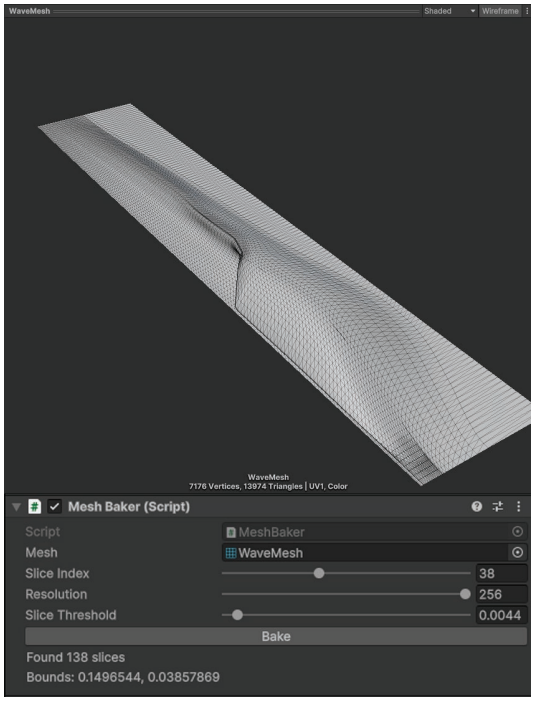
デフォーメーションテクスチャは波を作り出します。

波を作成するために、Shader Graph でデフォーメーションテクスチャをサンプリングし、波の形状を定義します。シェーダーは UV を調整して対称性を保たせ、ブレンド領域を使用して波を周囲の水に滑らかに戻します。MoveWave スクリプトは、波を +X 軸に沿って移動させることでアニメーション化します。



Water Decal スクリプトは水をデフォームします。

デフォーメーションテクスチャ自体は、それぞれ異なる波の状態を表す複数のスライス で構成される既製のメッシュから生成されます。MeshBaker スクリプトはこれらのスライスを処理し、頂点位置を記録してテクスチャに格納します。各テクスチャ行はスライスに対応し、テクスチャチャンネルは X、Y、Z デフォーメーションをエンコードします。



カスタム Baker を使用してデフォーメーションテクスチャを生成します。

スクリプトをテストするには、**Mesh Baker** ゲームオブジェクトを有効にして設定を調整し、Inspector で **Bake** を押します。

水中

水中 (Underwater) サンプルは、カメラが水面下に潜ったときに何が起こるかを示しています。水と空気を水面ラインが分けています。デフォルトでは、これは水上と水中のビューを分けるはっきりとした線です。より自然に見えるよう、**Custom Pass** を使用すると、この遷移を緩やかにすることができます。

このエフェクトは、水面ラインの周囲にぼかしを適用する **Full Screen Shader Graph** を使用します。シェーダーは、特殊な水中バッファをサンプリングすることで、どのピクセルが水中にあるかを検出します。その後、**HD Scene Color** ノードを使用して水面ラインの端だけをぼかし、遷移をより滑らかにします。

シェーダーが水面ラインをぼかします。

このエフェクトをよりリアルに見せるために、シェーダーは画像を水面ラインの上にわずかに伸ばしてメニスカス効果を加えています。

Custom Pass が画面上に水滴を加えます。

Custom Pass は、**HD Scene Color** ノードのサンプリング前に、画面の UV を歪める水滴も追加します。その後、**Custom Render Texture** (カスタムレンダーテクスチャ) が各フレームをずらして、滴るエフェクトを作成します。

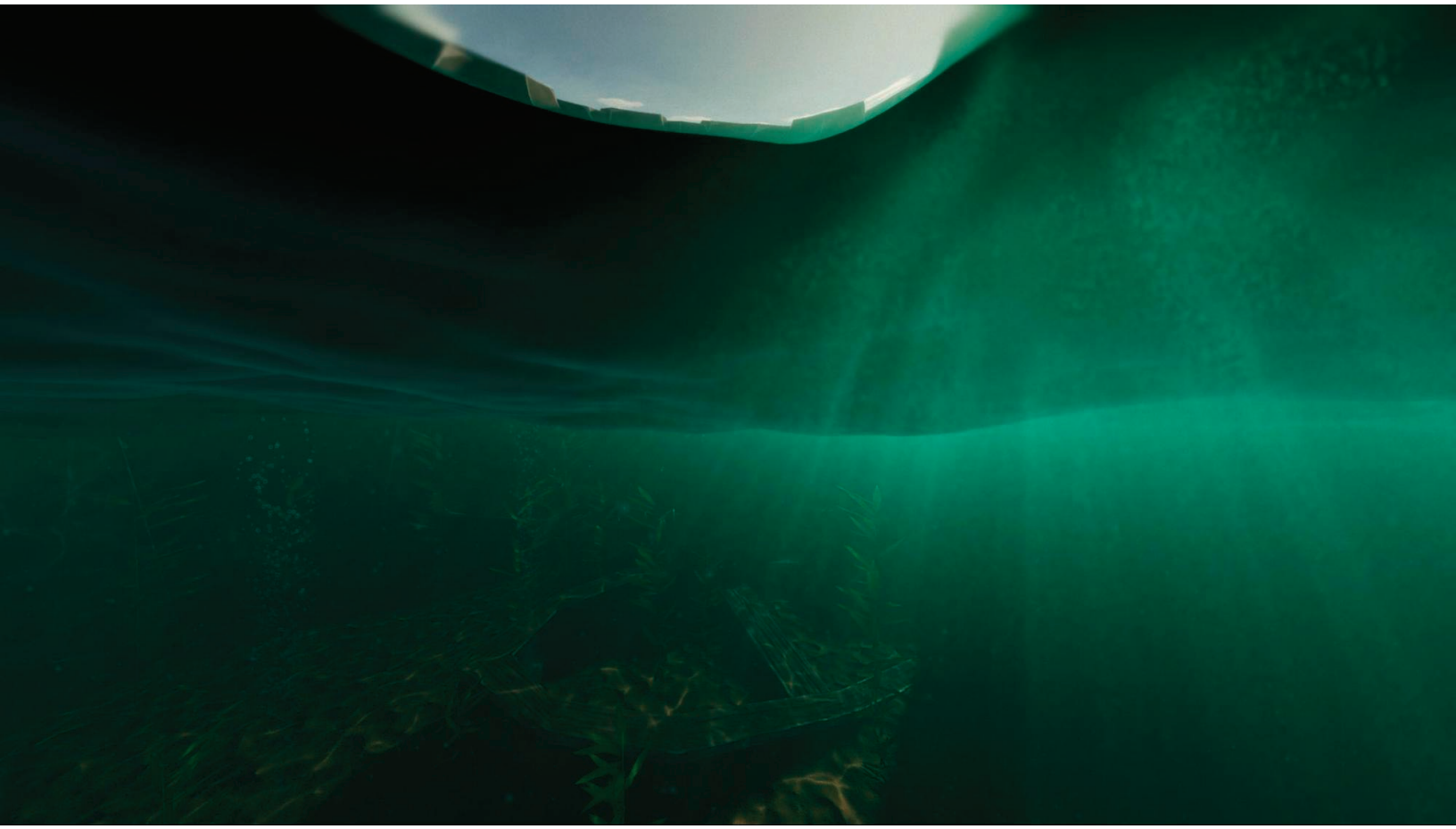
このシーンでは、背景に命を吹き込むために VFX Graph エフェクトも使用されています。

— 植物は頂点色と正弦波を使ってアニメーションします。

— 魚の群れが水の中を躍動的に泳ぎます。

— 泡は、**Sample Water Surface** ノードを使用して、水面上に表示されないようにします。

— 浮遊粒子は、水中の濁った雰囲気をシミュレートします。



ボリュメトリックシャドウが水中のリアルさを加えます。

ボリュメトリックフォグは、光が水中でどのように散乱するかをシミュレートします。Water Surface コンポーネントからコースティクステクスチャをサンプリングして、水を透過する光線のエフェクトを作成し、印象的なボリュメトリックなシャドウを加えます。

シェーダーとマテリアル

マテリアルは、オブジェクトの光の反射や発光の仕方を決定するため、レンダリングにおいて重要な役割を果たします。マテリアルを使うと、オブジェクトを金属、ガラス、木、あるいは抽象的だったり不思議なものに見せることができます。

マテリアル にはシェーダーオブジェクトへのリファレンスが含まれています。シェーダーオブジェクトが マテリアルのプロパティ を定義している場合、マテリアルには色やテクスチャリファレンスなどのデータも含まれます。

シェーダー自体は GPU 上で動作するプログラムで、各ピクセルの色を決定するものです。テクスチャ、ライト情報、マテリアルプロパティなどの入力データに基づいて計算を行います。

マテリアルのサンプル

HDRP の Package Manager には Materials サンプルが含まれています。

HDRP Materials サンプル には、HDRP に特化したマテリアルやシェーダーのさまざまなサンプルが含まれています。サンプルからは、サブサーフェススキャタリング、ディスプレイスメント、異方性などの効果が確認できます。同梱されている Shader Graph は、Lit シェーダー スタック、Fabric マスタースタック、Decal マスタースタック などの マスタースタック (後述) を利用します。

Fabric、Hair、Eye マスターノードでは通常、アーティストによる Shader Graph 内での細かい作業が必要です。これらのサンプルは、シェーダーを自作する出発点として使うことができます。

Fabric サンプル

Eye (目) サンプルは、特殊な UV 設定と特定のインポートスケールファクターを持つ慎重に設計されたメッシュを使用しています。同様の品質の目を作成するには、3D モデリングソフトウェアでこれらの目のメッシュを調べ、その構造と UV レイアウトを理解しましょう。

マテリアルによっては、適切に可視化するために レイトレーシング をサポートする GPU が必要であることにご注意ください。

マテリアルバリアント

どうすれば Unity で複雑なマテリアルライブラリを効率的に管理し、変更を適用できるのか疑問に思ったことはありませんか？マテリアルバリアントを使用すると、通常のマテリアルをベースにしたテンプレートやマテリアルプレハブを作成できます。

マテリアルバリアントの例

バリアントは、テンプレートマテリアルと共通のプロパティを共有し、必要なものだけをオーバーライドすることができます。ゲームオブジェクトにおけるプレハブと同じように、マテリアルバリアントでは、特定のプロパティをオーバーライドされたくない場合にロックすることができます。

例えば、プロジェクトに木のマテリアルを複数設定したい場合、Wood Base マテリアルから始めて、異なるテクスチャを持つ Oak や Cherry などのバリエーションを作ることができます。これらのいずれかを基に、通常色の色合いなどの他のプロパティをオーバーライドする独自のマテリアルバリアント (例: ホワイトオーク、レッドオーク) を作成することもできます。

マテリアルバリアントを使用すると、複数のプロジェクトで使用できるベースマテリアルの汎用ライブラリを作成し、必要に応じてオーバーライドすることが可能になります。そのため、アーティストは微調整のためにマテリアルを複製する必要がなくなり、特に大規模なプロジェクトでは視覚的なバグやパフォーマンスの問題を減らすことができます。

マテリアルバリアントの詳細については、こちらのドキュメントページ を参照してください。

マテリアルのプロパティー

シーンのマテリアルの多くは HDRP Lit シェーダー を使用します。以下はその最も重要なプロパティの一部で、他の HDRP シェーダーにも見られる可能性があります。

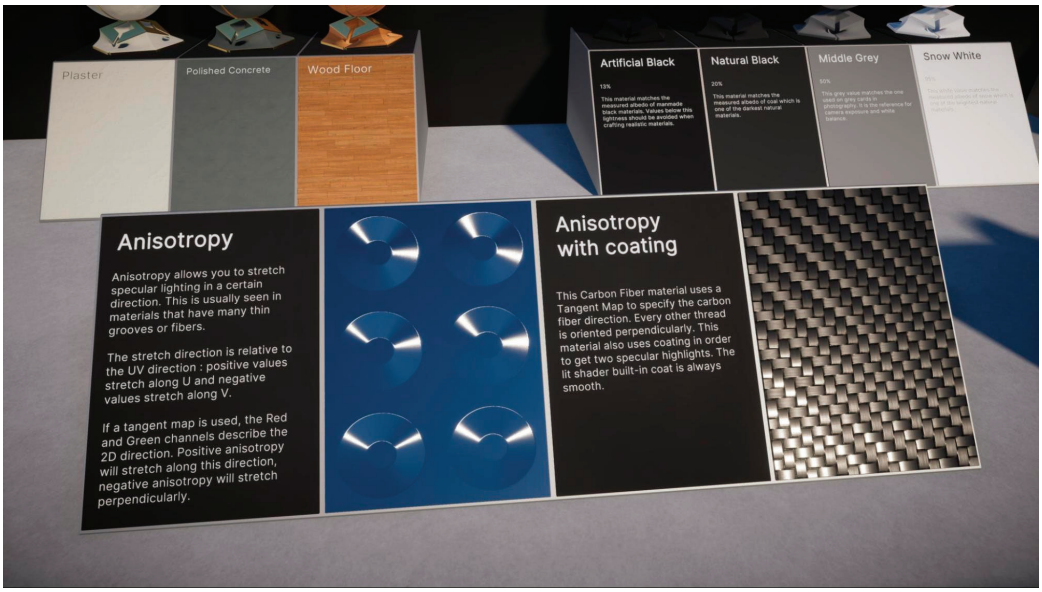
Base Color:マテリアルの色および不透明度を制御します。

Metallic:表面がどの程度 “金属的“ に見えるかを決定します。値が高いほど金属光沢が増し、値が低いほど非金属的 (または誘電的) な外観になります。

Specular:非金属マテリアルの反射率を決定します。金属的なマテリアルの場合、スペキュラーハイライトカラーは通常色から算出されます。

Smoothness:マテリアルのリフレクションの透明度を決定します。滑らかさの値を高くするとより光沢のある滑らかな表面になり、値を低くすると、よりマットで粗いマテリアルになります。

Anisotropy:マテリアルのハイライトが一方向に伸びる視覚効果を生み出します (例: カーボンファイバー、ブラッシュドメタルなど)。



異方性は、ハイライトを一方向に伸ばします。

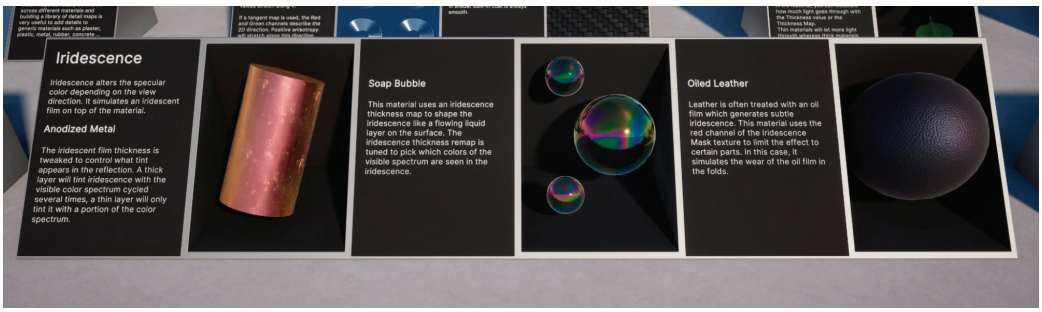
Thickness:通常、サブサーフェススキャタリング (SSS) とともに使用され、ライトが半透明のマテリアルの表面を通過し、内部で散乱するように見せます。

Emission:マテリアルを "光らせる" ことができ、ライトとして機能するオブジェクト (例: ネオンライト) を作る際に役立ちます。

法線マップ:凹凸や溝などの詳細な特性をサーフェスに加えられる、特殊なテクスチャタイプです。

Detail Map:ベースマップの上にさらにテクスチャレイヤーを重ねられるようになり、近くで見たときのテクスチャの外観をより正確に制御できるようになります。

Iridescence:見る角度が変わるにつれて色が変化する現象を表します（例: シャボン玉や特定の宝石に見られるもの)。

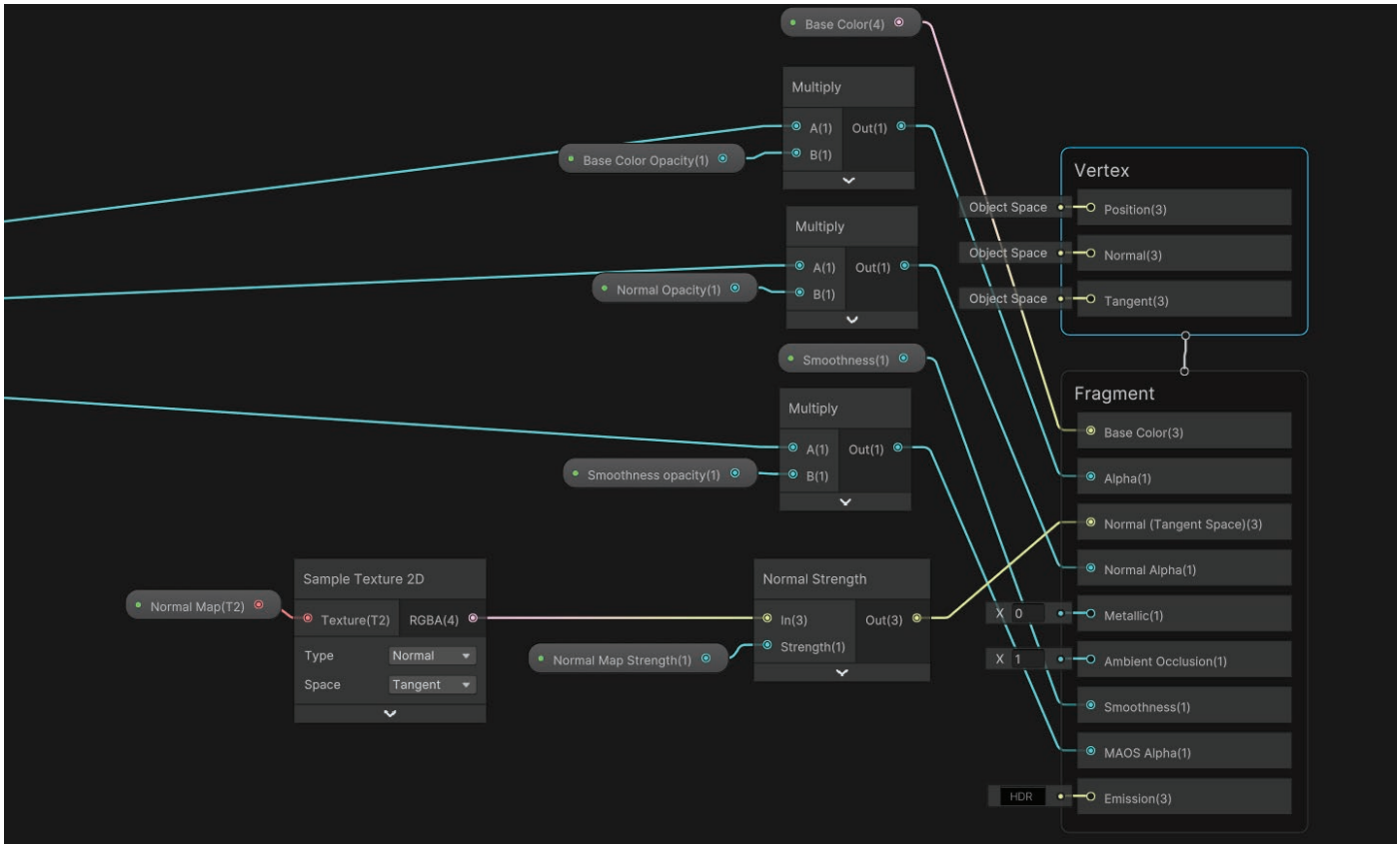


遊色効果のある (玉虫色の) オブジェクトは、見る角度によって色が変わります。

Shader Graph

Shader Graph は、コードを書くのではなく、シェーダーを視覚的にビルドできる強力なビジュアルツールです。HLSL のような言語でシェーダーコードを書いたり、デバッグしたり、保守したりする必要がなく、複雑なシェーダーを作成するためのノードベースのインターフェースを提供します。

Shader Graph のノードをつなげることで、開発者は複雑なシェーダーを視覚的に構築することができ、シェーダープログラミングの熟練者でない人にとっても、プロセスがより扱いやすいものとなります。Shader Graph の各ノードは、数学演算、関数、またはシェーダーのプロパティを表します。



ノードベースの Shader Graph でグラフィカルにシェーダーをビルドしましょう。

Shader Graph は HDRP と完全に統合されており、HDRP 向けに特別に設計された一連のノードや機能を提供します。これには、HDRP の物理ベースのライティングモデル、高度なマテリアルタイプ、ポストプロセスエフェクトのサポートが含まれます。

HDRP は高解像度のビジュアルを優先するため、よりリアルな物理ベースのマテリアルを作成するのに役立つ様々なサポートシェーダーが含まれています。各 HDRP マテリアルタイプは、Shader Graph のマスタースタックに対応しています。

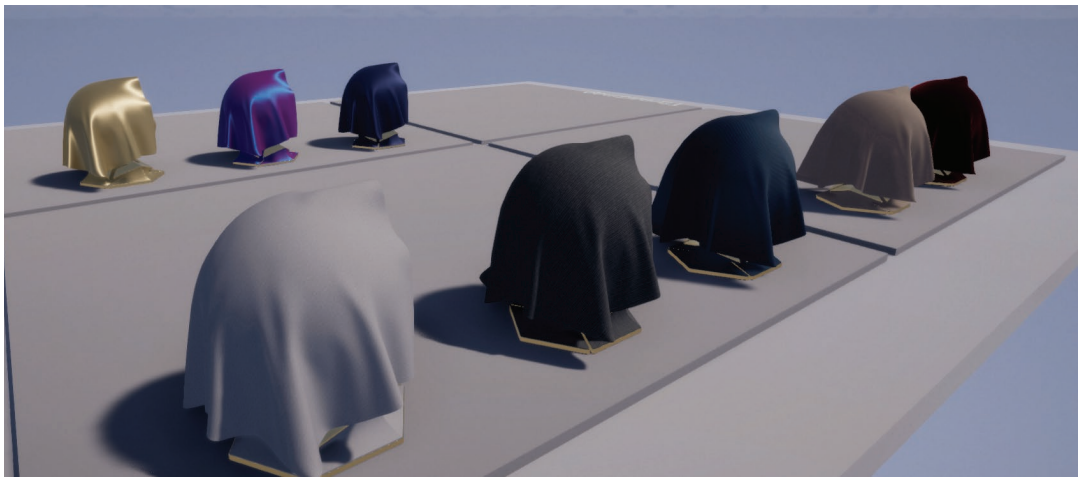
HDRP マスタースタック

マスタースタックは Shader Graph の終点で、シェーダーのサーフェスの外観を最終定義するものです。Shader Graph には、マスタースタックが 1 つのみ含まれます。

HDRP のマスタースタックには以下が含まれます。

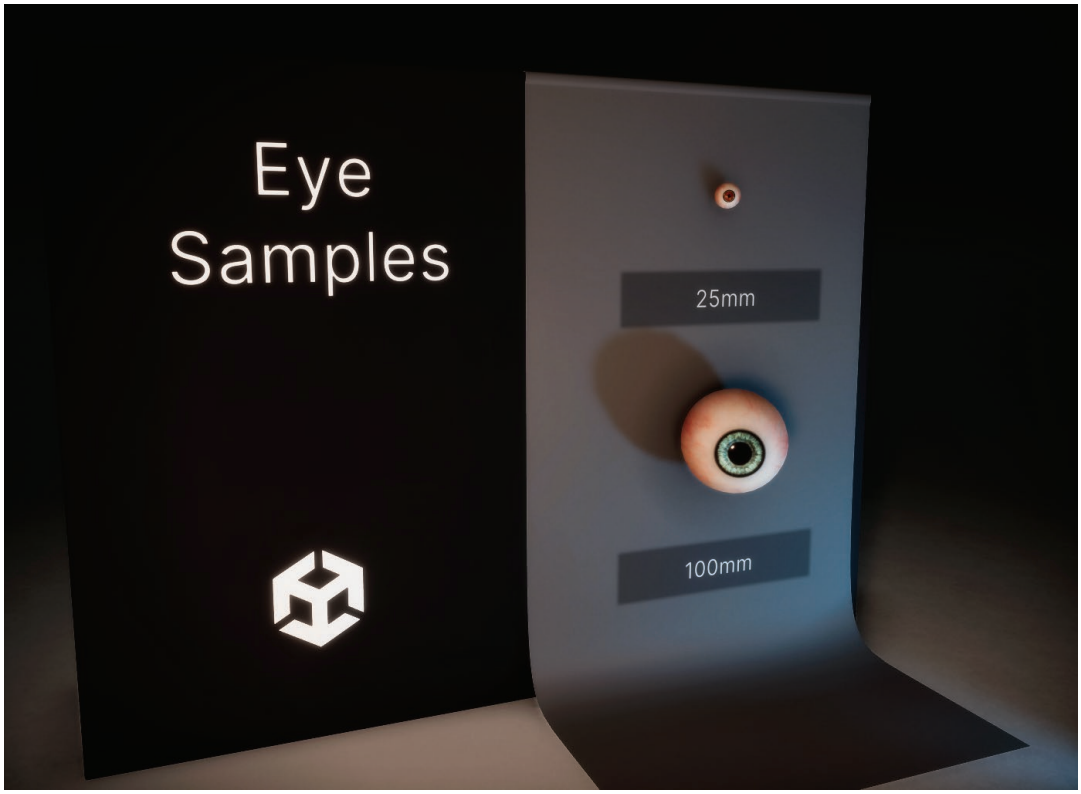
- **Lit マスタースタック:**物理ベースのライティングを持つ汎用マテリアルを作成します。現実味のあるマテリアルの最も一般的なマスタースタックで、サブサーフェススキャタリング、透過度、異方性など、幅広い機能をサポートしています。
- **StackLit マスタースタック:**複数のレイヤーを持つ複雑なマテリアル向けに設計されており、ライトが各レイヤーとどのように相互作用するかを詳細に制御できます。例えば、ベースレイヤー、メタリックフレークレイヤー、クリアコートを必要とする自動車塗装材料には、StackLit マスタースタックを使用します。
- **Unlit マスタースタック:**ライトと相互作用しないマテリアルに最適で、光と影を考慮することなく、色とテクスチャを完全に制御できます。
- **Decal マスタースタック:**サーフェスにグラフィカルなオーバーレイを追加します。完成したシェーダーを使用して、サーフェスに汚れ、傷、落書きなどのディテールを加えることができます。
- **Water マスタースタック:**このシェーダーは、リフレクション、屈折、その他の水に関連する視覚効果 (VFX) など、水面のシミュレーションに特化しています。詳しくは水システムのセクションを参照してください。

— **Fabric マスタースタック:**布マテリアルのシミュレーションを行います。対応するシェーダーは布マテリアルをレンダリングすることができ、光沢の色や強度に関するオプションが含まれています。



布マテリアルには、光沢、色、強さを変更するためのオプションがあります。

— **Eye マスタースタック:**角膜、強膜、虹彩を含むリアルな目のレンダリングに役立ち、反射や屈折を詳細に制御できます。



Eye マスタースタックはリアルな目のレンダリングに役立ちます。

— **Hair マスタースタック**:髪や毛皮をシミュレートするためのもので、色、スペキュラーハイライト、そして毛束における光の散乱具合を制御できます。

Hair マスタースタックは、毛束において光が散乱する様子をシミュレートします。

— **Canvas マスタースタック**:ユーザーインターフェース要素の UGUI シェーダーに使用します。

— **Fog Volume マスタースタック**:フォグボリュームの特定の設定とコンテキストを提供します。局所的な煙や霧などのリアルな環境エフェクトを作成できます。

— **Fullscreen マスタースタック**:ポストプロセスシェーダーなどの全画面エフェクトに使用します。

Enemies のデモには、デジタルな人型のキャラクターが登場します。

『*Enemies*』デモプロジェクトや Time Ghost:Character アセットで Eye マスタースタックと Hair マスタースタックの動作を確認できます。

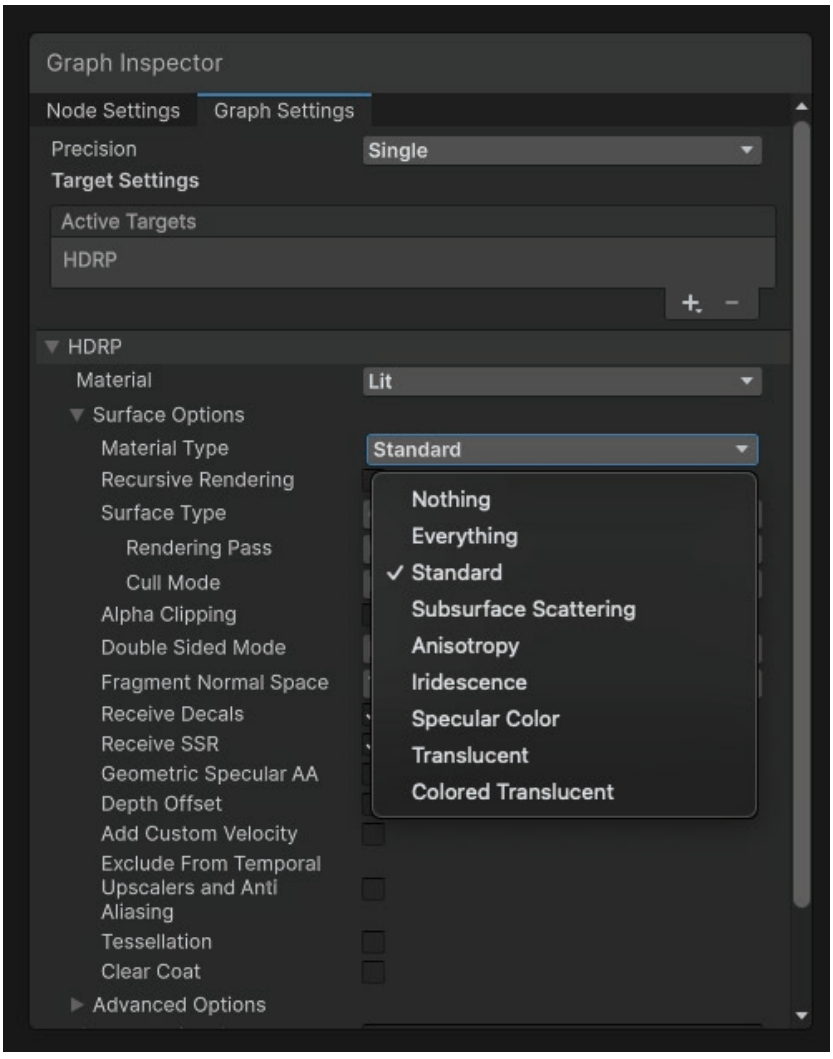
ShaderGraphのノードの中には、HDRP専用の機能を提供するものもあります。これにはDiffusionProfile、Emission、Exposure、HD Scene Color、HD Scene Depth などが含まれます。特に Eye マスタースタックや Water マスタースタックには、そのスタック専用のノードが数多くあります。HDRP 専用ノードの完全なリストについては、ドキュメントをご覧ください。

Material Type プロパティ

Unity 6.1 では、Shader Graph を使用して、**Material Type** (マテリアルタイプ) プロパティ (Standard、Subsurface Scattering、Translucent など) をシェーダー内で直接公開できます。

Shader Graph 内で、どのマテリアルタイプを公開するかを指定でき、マテリアルの Inspector で動的にマテリアルタイプを選択できます。

これにより、1 つの Shader Graph を複数のマテリアルタイプやマテリアルバリアントにわたって使用することが可能になります。



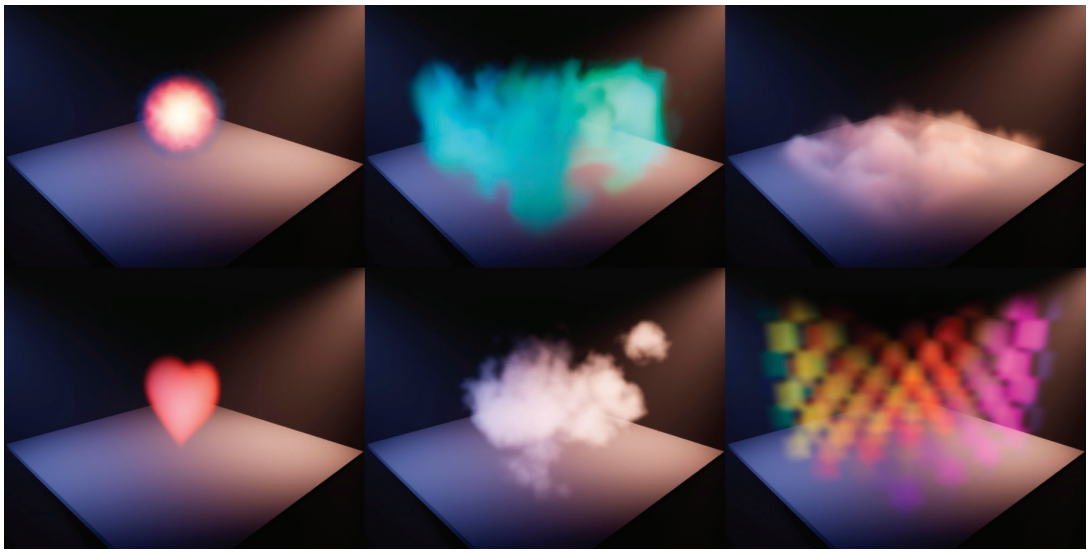
ボリュメトリックシェーダーグラフフォグ

ボリュメトリックマテリアルを使って、高度なプロシージャルフォグとボリュメトリック効果を作成しましょう。これらは、任意のローカルボリュメトリックフォグコンポーネントに適用される Shader Graph の組み合わせを使用します。

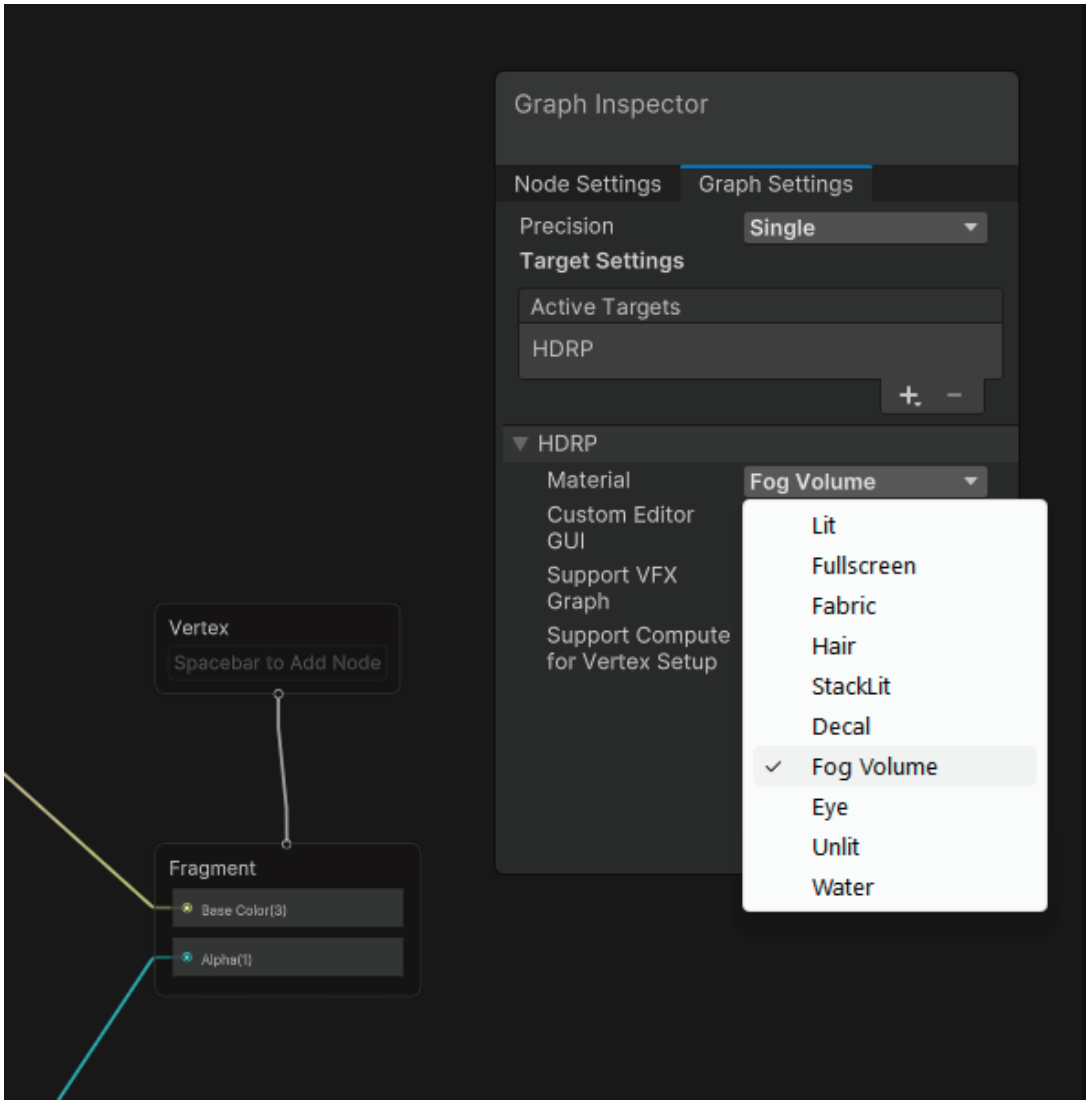
ボリュメトリックプロシージャルフォグの一例。

これを使用することで、カスタムグラウンドフォグや雲の効果を作成できます。また、砂嵐やオーロラなど、他の自然現象をシーン内で作り出すこともできます。

より多くの例を確認するには、**HDRP Volumetrics サンプル** を Package Manager からインポートしてください。ここでは、Shader Graph は HDRP マテリアルとして**フォグボリューム**を使用し、ローカルボリュメトリックフォグのコンポーネントにディテールとアニメーションを追加しています。



HDRP ボリュメトリックサンプルコレクションは Package Manager で利用できます。



Fog Volume を Shader Graph マテリアルとして設定します。

全画面 Shader Graph

Unity の全画面シェーダーは、開発者やアーティストがスクリーンビュー全体にわたるカスタムエフェクトを作成することを可能にします。これを使用することで、キャラクターがダメージを受けると画面が赤くなったり、水滴がついたりといった効果をかけられるようになります。

全画面シェーダーの用途は 3 つあります。

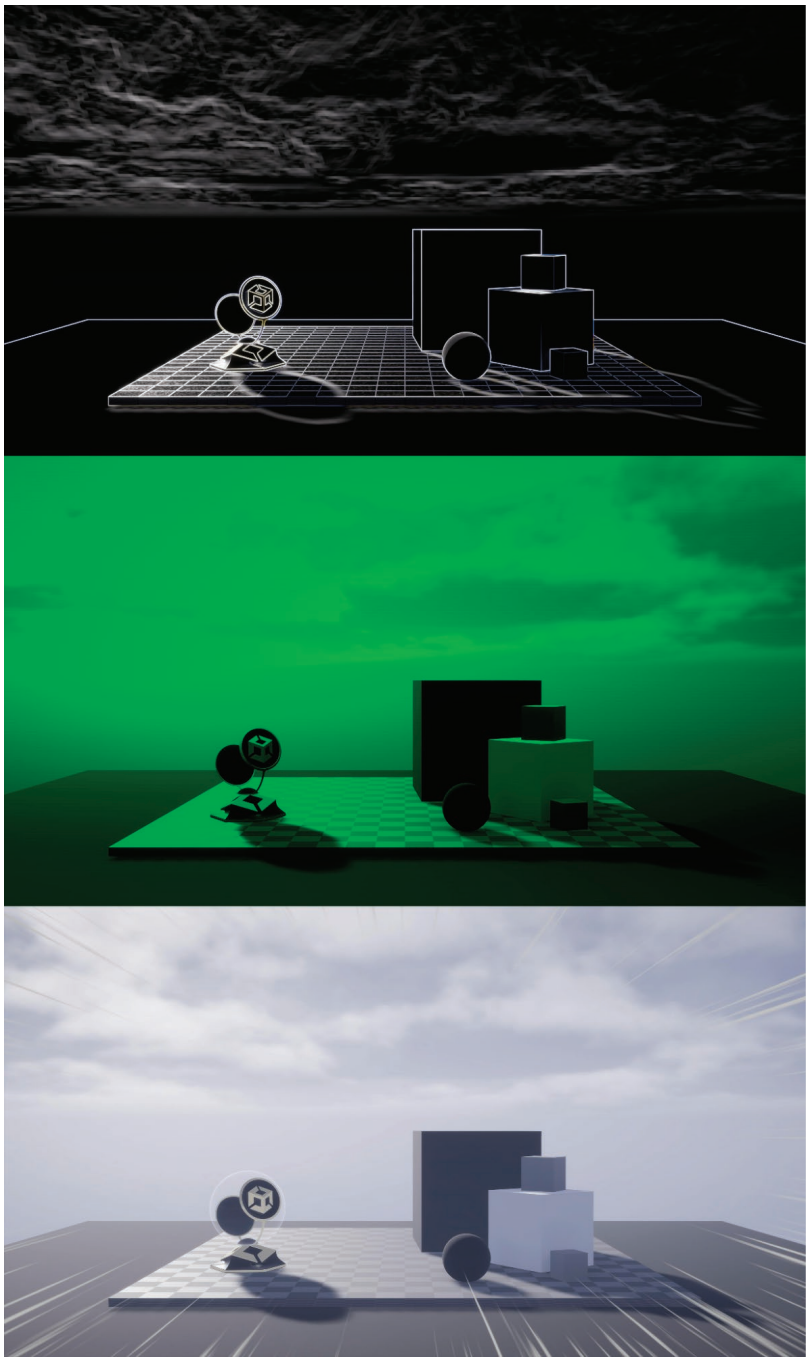
— カスタムパス効果の作成

— カスタムポストプロセスエフェクトの作成

— **HDUtils.DrawFullscreen** または **Graphics.Blit** 関数を使用した C# スクリプト内での使用

全画面シェーダーを作成するには、新しい全画面 Shader Graph を作成するか、既存のものを修正します。全画面マスタースタックを必ず含めてください。

HDRP には全画面シェーダーのサンプルも含まれており、Package Manager を介してプロジェクトにインポートすることができます。これらは全画面 Shader Graph でできることの代表例です。

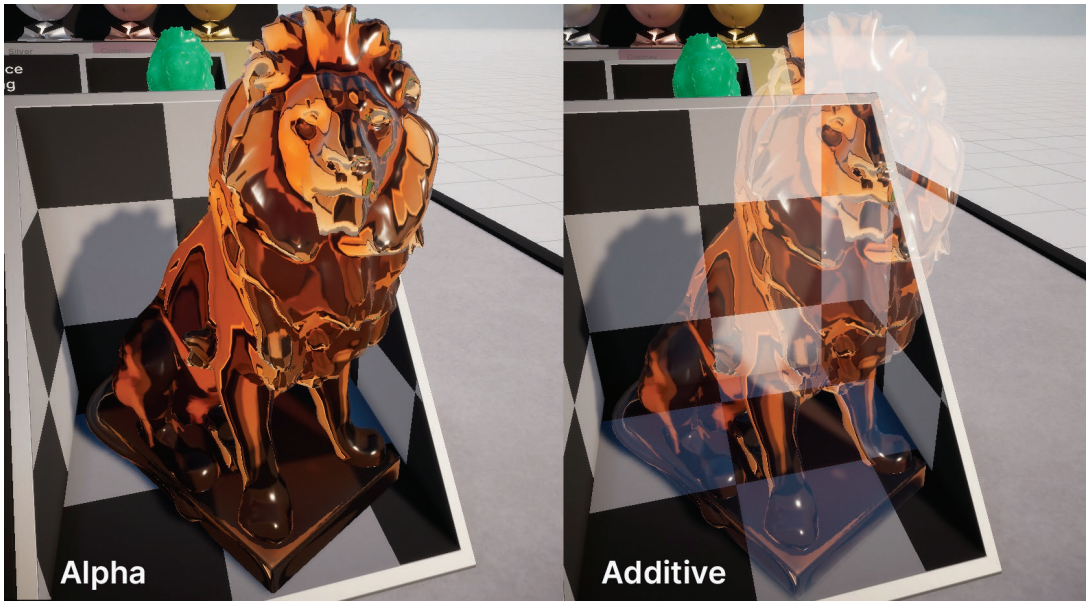


全画面 Shader Graph サンプル

Transparency

Transparency(透明度)とは、ガラスや透明なプラスチックのようにマテリアルが光を通す度合いのことです。HDRP では、透明度はマテリアルの通常色のアルファ値を調整することで処理されます。アルファ値が低いとマテリアルの透明度が上がり、アルファ値が 1 だとマテリアルは完全に不透明になります。

HDRP には透明度のモードが複数あります。**Alpha**、**Premultiply**、**Additive** です。Alpha はテクスチャのアルファ値を直接使用し、Premultiply は適用する前にアルファ値と色を乗算します。Additive はマテリアルの色を背景に加えるもので、光り輝くエフェクトを作るときに便利です。



Alpha と Additive のブレンドモードを比較します。

Lit Material サンプル

Lit Material サンプル (Package Manager から入手可能) は、透明なマテリアルの設定方法の例を示しています。Surface Type を Transparent とする場合、Refraction Model (屈折モデル) でモデルを選択し、マテリアルに適した **Index of Refraction** を設定します。

Thin Glass の屈折モデルを使用して、窓のような薄いガラスの表面をシミュレートします。**Transmittance Color** は屈折を着色します。

厚みのあるガラスオブジェクトの場合は、**Sphere** または **Planar** 屈折モデルを使用します。Sphere 屈折モデルは宝石やビー玉を近似できます。Planar 屈折モデルは、角氷のような平らなものに対して機能します。

レンダラーはオブジェクトの形状とサイズを認識しないため、これらのモデルで **Thickness Map** (厚みマップ) を指定して屈折をよりリアルにすることができます。マップ内のマテリアルの厚みを増すほど、色も濃くなります。

Lit Material サンプルには、さまざまな屈折モデルが示されています。

Transparent Material サンプル

Unity 6 には、透明度の影響をより詳細に確認するための **Transparent** サンプル (Package Manager で個別に入手可能) もあります。これらのマテリアルは、透明な表面を扱う際にシャドウ、スタッキング、屈折を設定する方法を示しています。

Transparent サンプル

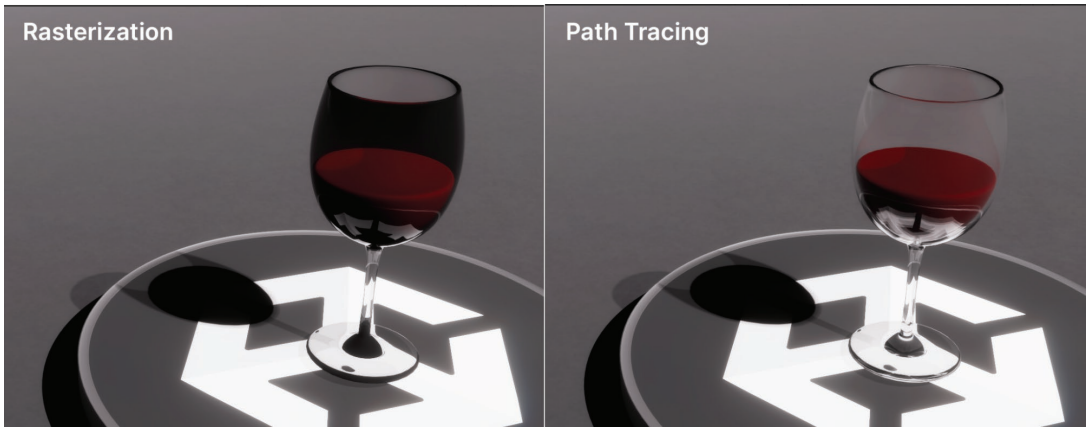
シーン内での透明なマテリアルの見え方に影響する複数のレンダリングモードを切り替えることができます。

— **Rasterization:**デフォルトの Rasterization (ラスタライゼーション) モードでは、HDRP はパフォーマンスを優先するために、屈折効果を近似して物理的な精度を犠牲にします。

— **Recursive rendering:**Recursive rendering (再帰レンダリング) モードは、透明なマテリアル内での複数の光の反射を処理し、よりリアルな屈折と内部反射を生成します。基本的なラスタライゼーションだけでは不十分な複雑な透明オブジェクトに使用します。

— **Ray traced shadows:**Ray traced shadows (レイトレーシングによるシャドウ) モードでは、透明なマテリアルに標準のラスタライゼーションを使用しますが、レイトレーシングでシャドウを計算します。こうすることで、パストレーシングを完全に行わなくても、透明な物体の影をより正確に見せることができます。

— **Path tracing:**Path tracing (パストレーシング) モードは、光が透明な媒体をどのように通過するかをシミュレートします。物理的に最も正確な結果が得られますが、計算コストも高くなります。パフォーマンスよりも品質を優先する場合に使用します。



レンダリングモードの比較

Shader Graph ノード

Shader Graph には、ラスタライゼーションの際にカスタム透明度に使用できるノードがデフォルトでいくつか含まれています　(レイトレーシングには非対応)。これらのノードは、屈折と深度ベースの透明度のシミュレーションに使用できます。

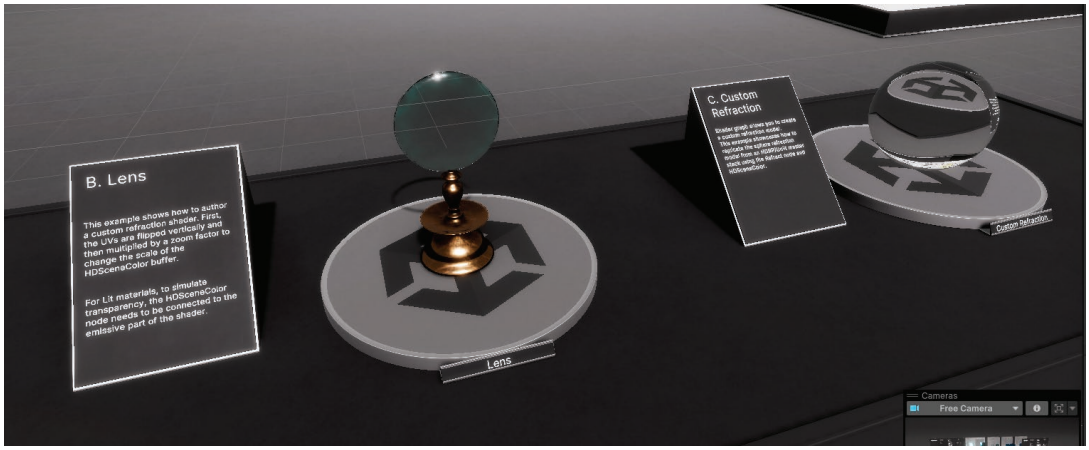
— **Refract**:透明なマテリアル (ガラス、水など) を通過する光の折れ曲がり具合をシミュレートします。

— **Scene Depth**:深度ベースの透明度とブレンドのために、透明な表面の背後にある深度情報をキャプチャします。

— **Scene Depth Difference**:透明なオブジェクトと背景との間の深度の変動を測定します。柔らかいエッジや歪みに使用します。

— **HD Scene Color**:背景色データを取得し、他の透明度、ブレンド、屈折効果に利用します。

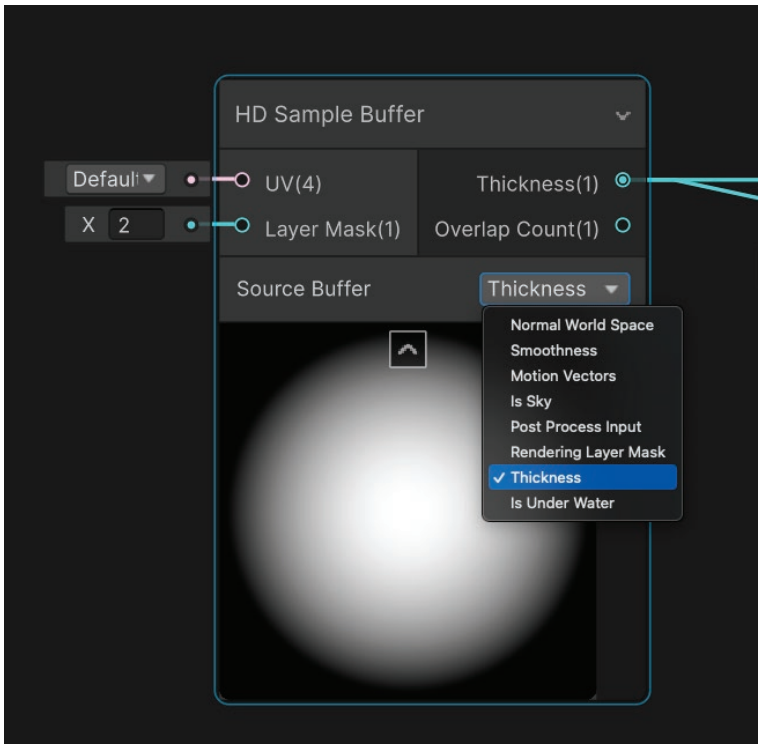
これらのノードを組み合わせて、シーンのカスタム透明度エフェクトを作成できます。これらの例では、UV と HDSceneColor バッファの操作によって屈折がどのように再現されるかを示しています。



Lens (レンズ) と Custom Refraction (カスタム屈折) のサンプル

Compute Thickness

HDRP アセットで **Compute Thickness** (厚みの計算) を有効にすると、指定した LayerMask 内のオブジェクトの厚み情報を含むバッファが割り当てられます。Shader Graph で HD Sample Buffer ノードを使用してこのバッファをサンプリングし、色や屈折結果に影響を与えます。



HD Sample Buffer で Compute Thickness をサンプリングします。

例えば、グミのクマに適用された X 線シェーダーは、Compute Thickness バッファを使用してグミのクマの骨格を表示します。Shader Graph は、エフェクトの外観を変更するために、Inspector でいくつかの追加コントロールを公開します。

X 線シェーダーは Compute Thickness バッファを使用します。

サブサーフェススキャタリングと透過度

サブサーフェススキャタリング（SSS）は、半透明のマテリアルに関連する現象です。透過とは、光が物質を通過することはできるものの、その過程で散乱される特殊な透明性のことです。

この効果は、皮膚や植生のような特定の有機マテリアルを通過する光が、ざらざらしたプラスチックではなく、滑らかに見えるようにするソフトな輝きを生み出します。

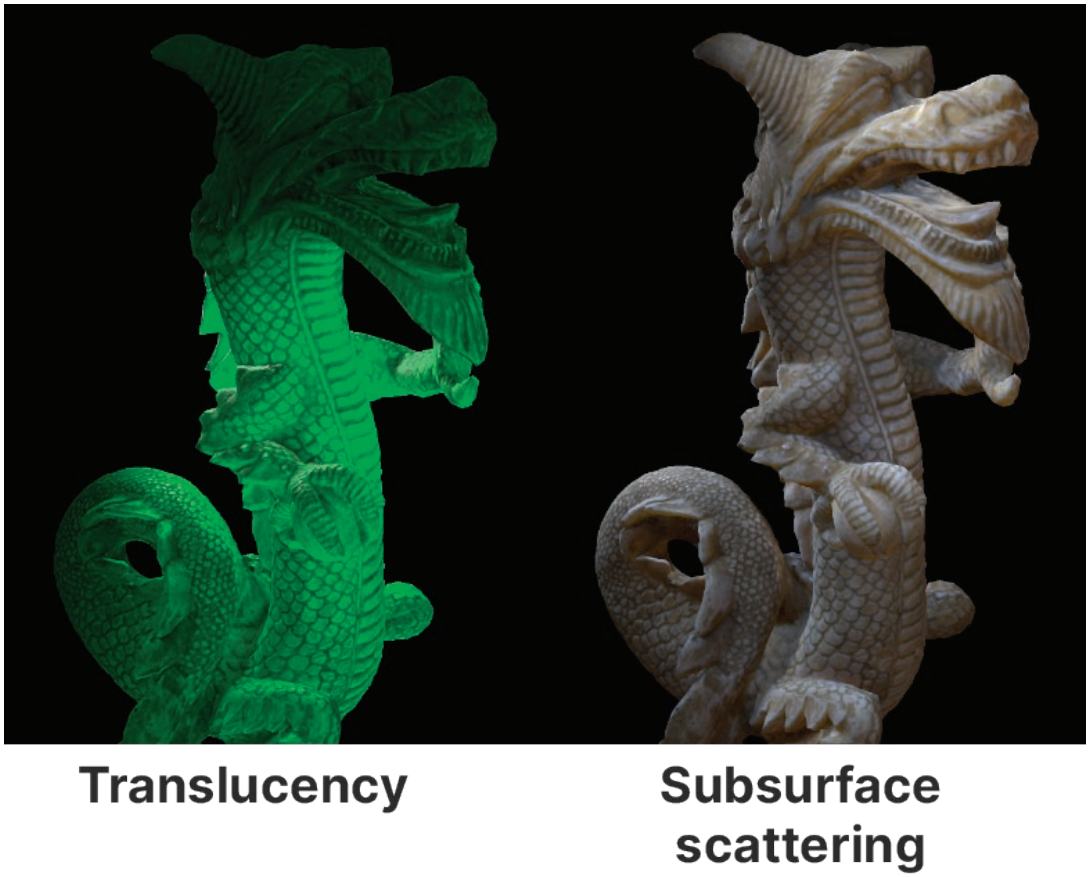
半透明のマテリアルに光が当たった時、そのすべてが反射するわけではありません。一部が内部を貫通し、マテリアル内部で跳ね返りながら散乱するのです。散乱した光の一部は、最終的に入った場所とは異なる地点でマテリアルから放出されます。

HDRP は、スクリーンスペースのぼかし技術を用いてサブサーフェススキャタリングを実装しています。また、ライトがゲームオブジェクトの背後から通過して透明に見えるようにするトランスミッションも扱います。トランスミッションの場合は、光が通過する際に色をつけることもできます。

HDRP では、SSS のマテリアルタイプが 2 つあります。

— **Subsurface Scattering** (サブサーフェススキャタリング) は、スクリーンスペースのぼかし効果とトランスミッションの両方を実現します。

— **Translucent** (半透明) はトランスミッションのみを扱います。



Translucent はトランスミッションのみを扱います。サブサーフェススキャタリングは、スクリーンスペースのぼかし効果を加えます。

HDRP アセット、および **Project Settings > HDRP Default Settings** でサブサーフェススキャタリングを有効にします。次に、SSS 設定を保存する Diffusion Profile アセットを作成します。HDRP は、オーバーライドオプションとともに、同時に最大 15 個のカスタムプロファイルを表示できます。

必要に応じて各マテリアルタイプを **Subsurface Scattering** または **Translucent** に設定し、Diffusion Profile を割り当てます。

このシーンのドレープのような SSS マテリアルは Diffusion Profile を使用します。

Diffusion Profile は、散乱光の色、トランスミッションの色合い、厚みなどのプロパティを制御します。

また、各マテリアルには、SSS に影響する他のオプションが 2 つあります。

— **Thickness Map** は、オブジェクトのさまざまな部分にわたる散乱とトランスミッションの効果を制御します。

— **Transmission Mask** は、トランスミッションの全体的な強度を制御します。例えば、これを使用することで、木の幹と葉の両方に 1 つのシェーダーを使用できるでしょう。

サブサーフェススキャタリングを使用すると植生をよりリアルにレンダリングできます。

SpeedTree と Transmission Mask

Transmission Mask は、SpeedTree の植生を扱う際に便利で、樹皮や小枝を不透明にしたまま、葉だけを半透明にできます。Transmission Mask を使用して、本来存在しない場所にサブサーフェススキャタリングが適用されないようにします。

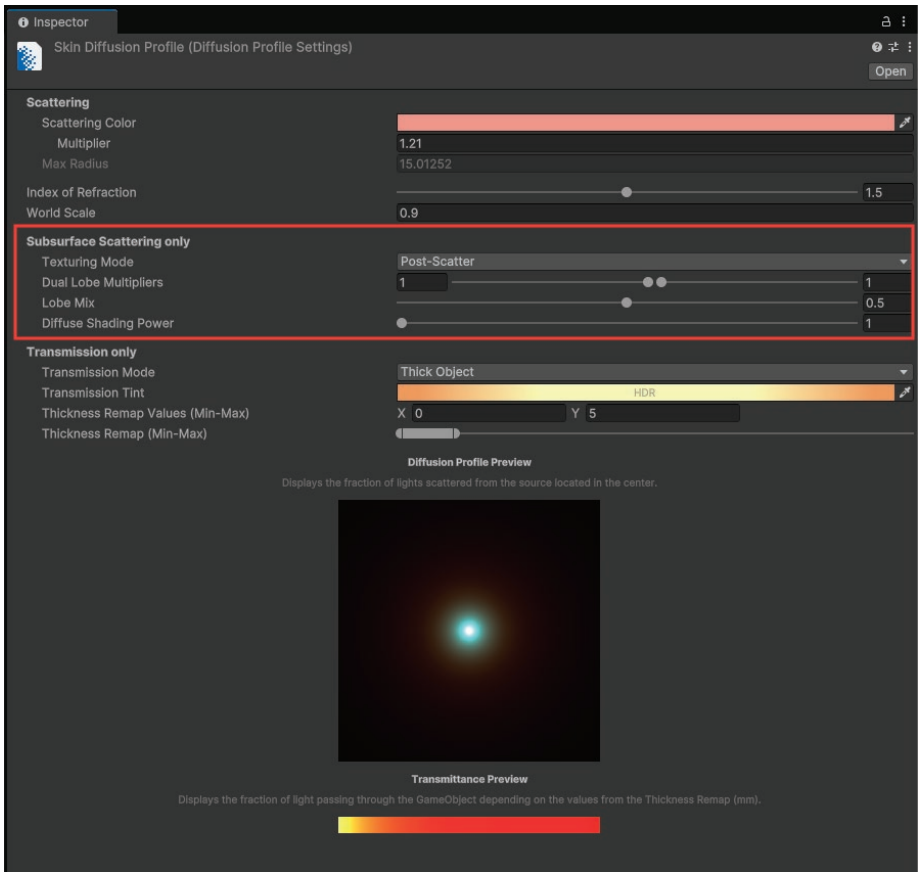
SpeedTree 8 Sub Graph Assets (HDRP/Nature/SpeedTree8.shadergraph) は、Transmission Mask ノードのサブサーフェスマップを使用して、樹皮や小枝からの意図しない光のトランスミッションを除去します。これにより、3D ジオメトリのライティングと一致していなかった、明るすぎるビルボードライティングの問題も修正されます。

Unity 6 の新機能:皮膚のレンダリングの改善

半透明マテリアルの Diffusion Profile は、光がサーフェス内でどのように散乱するかを制御します。Unity 6 では、Diffusion Profile に新しいパラメーターが導入され、皮膚のレンダリングが改善されています。

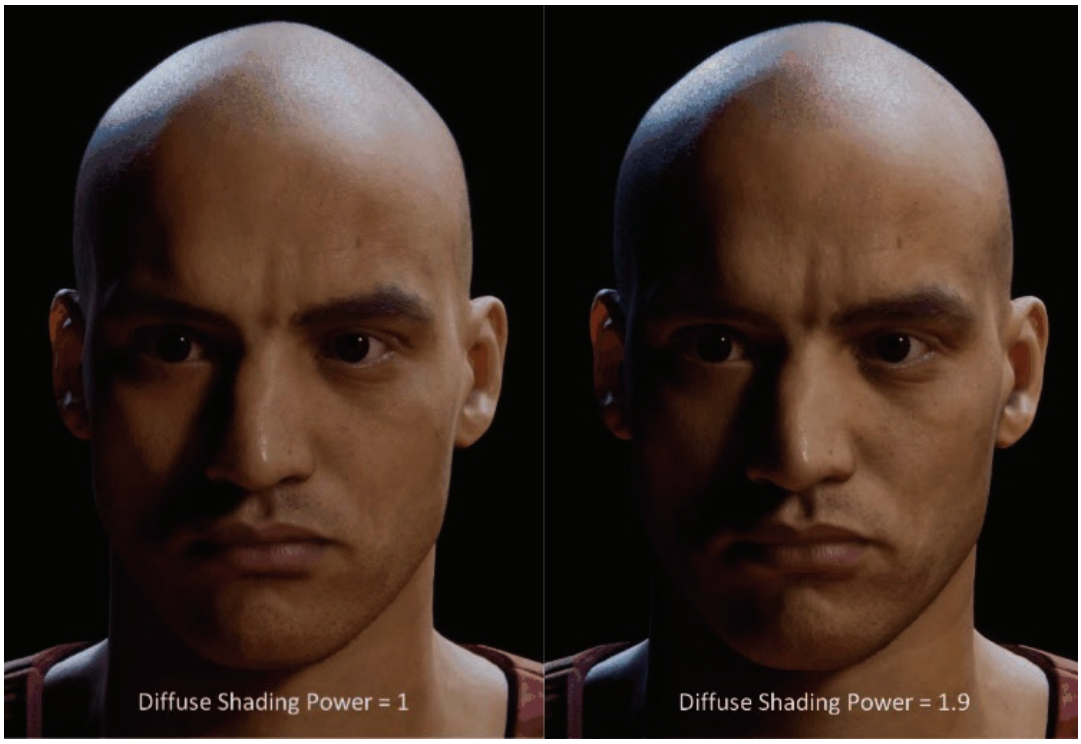
- **Dual lobe** 鏡面反射は、皮膚表面の薄い脂質層をシミュレートし、脂質による鋭いハイライトと、より深い層からの柔らかいリフレクションの両方を作成します。
- **Diffuse power** は、皮膚のようなサブサーフェススキャタリングが強いマテリアルにおける光の拡散を改善します。

これらのアップデートにより、さまざまなライティング条件下での皮膚のマテリアルの見た目がより自然になりました。



Unity 6 では、Diffusion Profile にパラメーターが追加されています。

Dual lobe は人間の皮膚の鏡面反射を捉えます。



Diffuse Shading Power は、光が皮膚のマテリアルにどのように広がるかを制御します。

サブサーフェススキャタリングにより、皮膚がよりリアルにレンダリングされます。

デカール

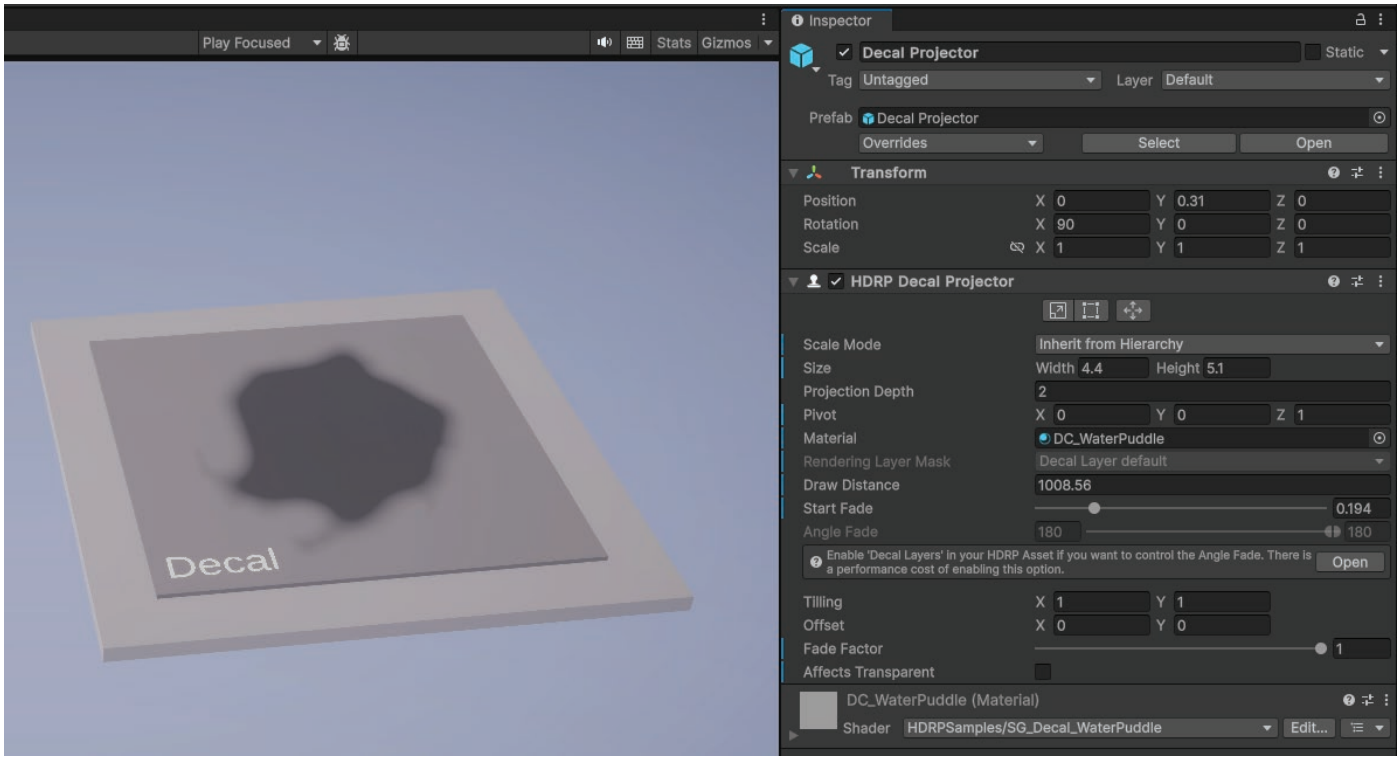
デカールは Decal Shader または Decal マスタースタックを使用するマテリアルです。ジオメトリを追加することなく、グラフィカルなオーバーレイとしてマテリアルやサーフェスにディテールを追加します。

デカールは基本的に、シーン内の他のオブジェクトに投影されるテクスチャであり、平面画像 (投影デカール) としても、3D オブジェクトの形状に合わせたもの (メッシュデカール) としても使用できます。

デカールを作成するための主要ツールは Decal Projector コンポーネントです。これを使用して、デカールのサイズ、深度、その他のプロパティを制御できます。

Start Fade と **Draw Distance** を使用して、距離によるデカールの可視性を決定します。**Angle Fade** を有効にすると、頂点法線に対する向きに応じてデカールはフェードします。

デカールを使用すると、街の壁に落書きを重ねたり、床や機械に磨耗や傷を追加したり、乗り物やオブジェクトにパターンやエンブレムを追加したりできます。



デカールはディテールを追加する手法です。

Unity 6.1 では、Shader Graph で作成したデカールが透明なオブジェクトに影響を与えるようになりました。これを使用して、雨滴、波紋、独自の刻印、ガラスの汚れのエフェクトなどのプロシージャルエフェクトを構築できます。

シェーダー:毛皮と髪

ヘアシェーダーまたはヘア Shader Graph を、HDRP で髪や毛皮をレンダリングするための出発点として使用できます。リアルな髪のエフェクトを作成するために、ヘアシェーダーとヘア Shader Graph は "ヘアカード" と呼ばれるレイヤーを使用します。

各ヘアカードは、髪の異なる部分を表します。半透明のヘアカードを使用する場合は、どの表示方向から見ても後ろから前の順になるように手動でソートする必要があります。

Hair マスタースタック は、髪と毛皮に使用できる 2 種類のマテリアルをサポートしています。

- **Approximate Material Type**:このヘアマテリアルタイプはヘアカードのジオメトリに適しており、Kajiya-Kay ヘアモデルに基づいています。このタイプを使用する場合は、Fragment コンテキスト内のブロックをシーン内のライティング環境に合わせて調整する必要があります。
- **Physical Material Type**:このヘアマテリアルタイプは、Marschner ヘアモデルに基づいた毛束ジオメトリ（細いチューブやリボン）に適しています。このタイプは、すべてのライティング環境に適しています。

  毛束ベースの髪は、より詳細でリアルですが、より多くの計算リソースを必要とします。そのため、メインキャラクターや高品質でのクローズアップに最適です。Physical Material Type は 多重散乱 もサポートしており、毛束内でのリアルな光の相互作用をシミュレートします。

HDRP はリアルな髪や毛皮を作成するのに役立ちます。

Unity 6.1 の新機能:

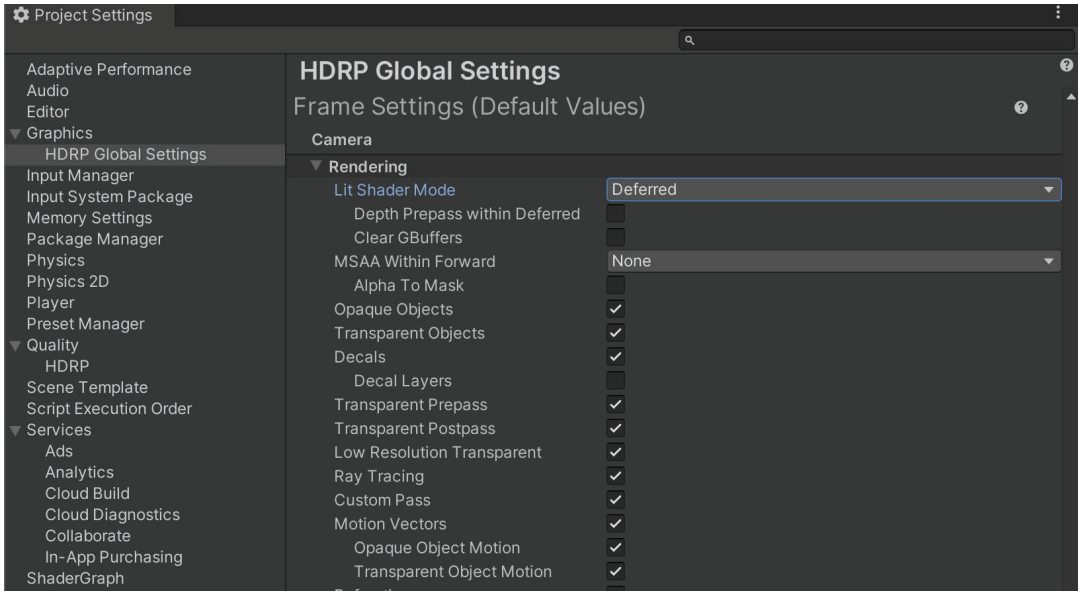
- **高品質なラインレンダリング:**従来のハードウェアラスタライゼーションを使用して細い線をレンダリングすると、エイリアシングやソートの問題が、特に髪や毛皮に生じる可能性があります。高品質のラインレンダリングにより、優れた透明性とアンチエイリアスの高品質な線が得られます。詳細については、Use high quality line rendering (高品質ラインレンダリングの使用) を参照してください。
- **スクリーンカバレッジ LOD モード:**このモードでは、スクリーンスペースの占有率に基づいてレンダリングされる毛束の量を動的に調整し、パフォーマンスを向上させます。
- **パフォーマンスの改善:**毛束ベースのヘアレンダリングでは、多数の毛束をレンダリングする際に、パフォーマンス、視覚的な品質、アンチエイリアスを最適化できるソフトウェアラスタライザーを使用します。

詳細については、Hair and fur (髪と毛皮) のドキュメントページを参照してください。Unity で髪や毛皮を設定するのが初めての場合は、ヘアシステムの概要について Get Started with Hair Simulation (髪のシミュレーションを開始する) を参照してください。

より優れた透明性とアンチエイリアスを備えた高品質なラインレンダリング

フォワードレンダリングとディファードレンダリング

パイプラインアセットで HDRP 設定を指定する場合、通常は **Rendering** の **Lit Shader Mode** を開きます。ここでは、**Deferred**、**Forward**、または **Both** を選択できます。これらは、パイプラインによるジオメトリのレンダリングおよびライティング方法に関連する一連の処理であるレンダリングパスを表します。



デフォルトの HDRP 設定の変更

レンダリングパスのカスタマイズ

Lit Shader Mode で **Forward** または **Deferred** を選択し、デフォルトのレンダリングパスを設定します。

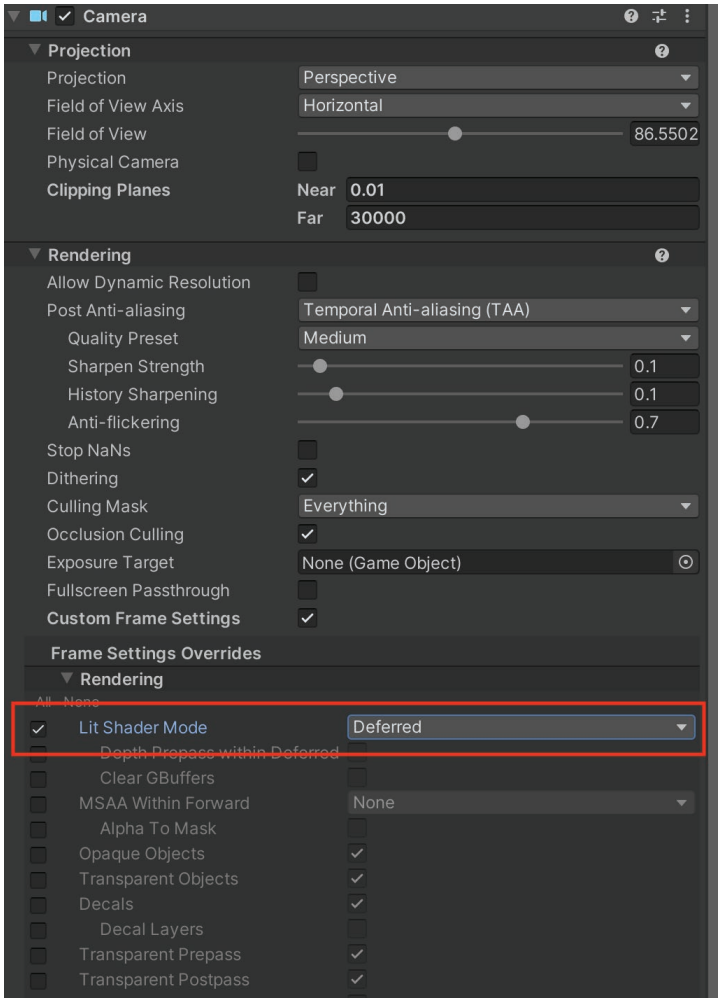
HDRP は柔軟性が高いので **Both** を選択することもできます。このオプションを利用した場合はほとんどのレンダリングにレンダリングパスを 1 つ使用することとなり、カメラごとにそのパスをオーバーライドできるようになります。ただし、この方法では GPU メモリの使用量が多くなってしまいます。ほとんどのケースでは、Forward か Deferred のいずれかを選択するのが有効です。

— デフォルトですべてのカメラを対象にするには、**HDRP Default Settings** に移動し、**Default Frame Settings** を確認します。これは **カメラ**、**ベイク**、**カスタム**、**リアルタイム**、いずれの **リフレクション** にも適用できます。

Rendering グループの **Lit Shader Mode** でレンダリングパスを設定します。

— 特定のカメラについては、**Custom Frame Settings** を確認してオーバーライドします。

その後 **Rendering** グループにある **Lit Shader Mode** のレンダリングパスをオーバーライドし変更します。



カメラのカスタムフレーム設定の変更

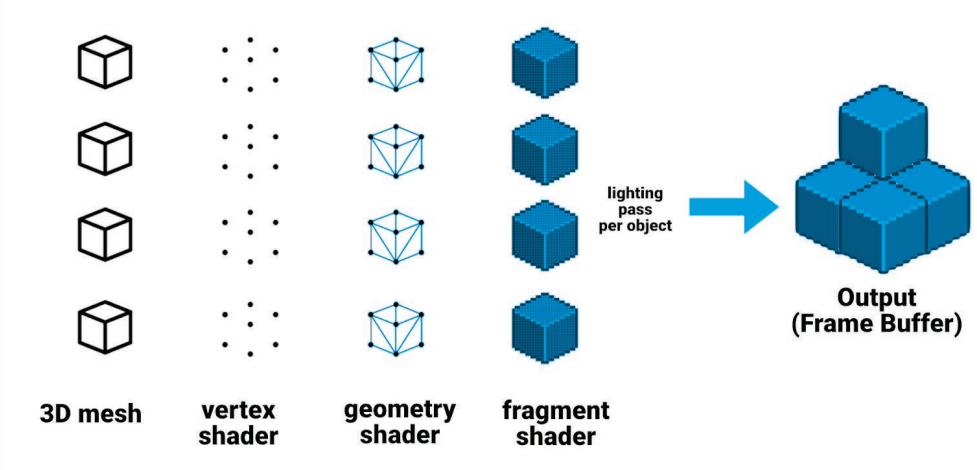
レンダリングパスに関する詳細

こうしたレンダリングパスの仕組みを理解し、Lit Shader Mode がパイプラインのその他の設定にどのように影響するのかを確認することをおすすめします。

フォワードレンダリング

フォワードレンダリングでは、グラフィックスカードによって画面上のジオメトリが頂点に分割されます。これらの頂点はさらにフラグメント（ピクセル）に分解され、最終的な画像を作成するために画面にレンダリングされます。

各オブジェクトは 1 度に 1 つずつグラフィックス API に渡されます。フォワードレンダリングでは、各ライトにコストがかかります。シーン内のライトが増えると、レンダリングの所要時間も長くなります。



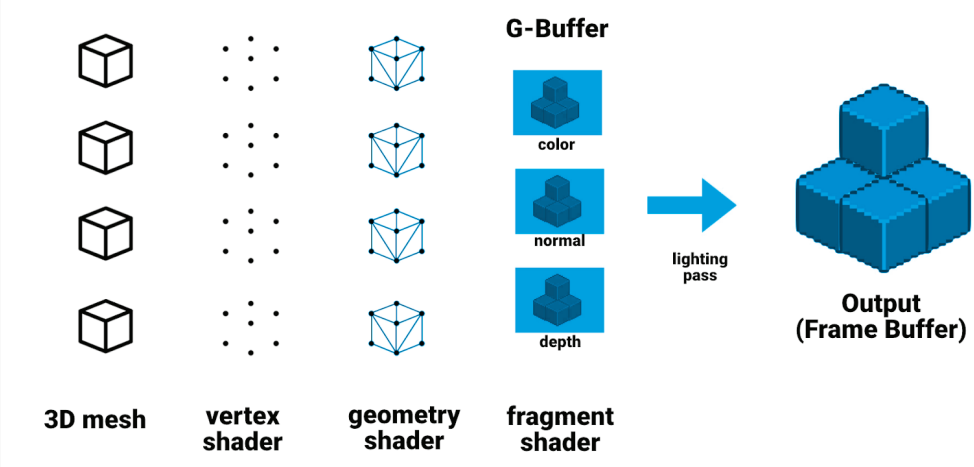
フォワードレンダリングパス

フォワードレンダリングでは、別々のパスでライトを描画します。同じゲームオブジェクトに複数のライトが当たる場合、ライトやオブジェクトの数が多いと、大幅な *オーバードロー* が生じ、動作が遅くなることがあります。

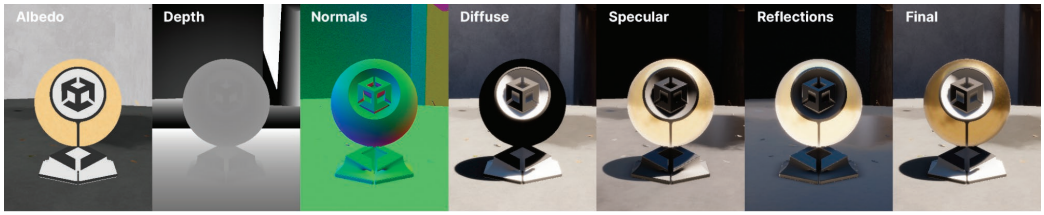
従来のフォワードレンダリングとは異なり、HDRP では、フォワードレンダラーに効率面の改良が加えられています。たとえば、オブジェクトマテリアルごとに 1 つのパスで、複数のライトをまとめてカリングし、レンダリングします。ただし、それでもプロセスのコストは比較的大きくなります。パフォーマンスに問題がある場合は、ディファードシェーディングを使用することをおすすめします。

ディファードシェーディング

HDRP では、オブジェクト単位でのライティング計算が発生しないディファードシェーディングも使用できます。ディファードシェーディングでは負荷の大きいレンダリングを後の段階に遅らせ、2 つのパスを使用します。



ディファードシェーディングパス



ディファードシェーディングは、各オブジェクトの代わりにバッファにライティングを適用する。それぞれのパスが、最終的にレンダリングされる画像に反映されます。

1 つ目のパスである G バッファ ジオメトリパスでは、Unity がゲームオブジェクトをレンダリングします。このパスは、複数種のジオメトリプロパティを取得し、テクスチャのセットに保管します (ディフューズ色、スペキュラー色、表面の滑らかさ、オクルージョン、法線など)。

2 つ目のパスの ライティングパス では、Unity が G バッファの完了後にシーンのライティングをレンダリングします。つまり、シェーディングを遅らせます。ディファードシェーディングパスはピクセルごとに反復され、個々のオブジェクトではなく、バッファーを基準にライティング情報を計算します。

各レンダリングパスの技術的な違いの詳細については、HDRP に関するドキュメントの Forward and Deferred rendering (フォワードレンダリングとディファードレンダリング) を参照してください。

アンチエイリアス

Lit Shader Mode のレンダリングパスは、アンチエイリアスを使用してレンダリングからギザギザのエッジを取り除く方法に影響します。HDRP では、制作のニーズに合わせて複数のアンチエイリアス手法を使用できます。

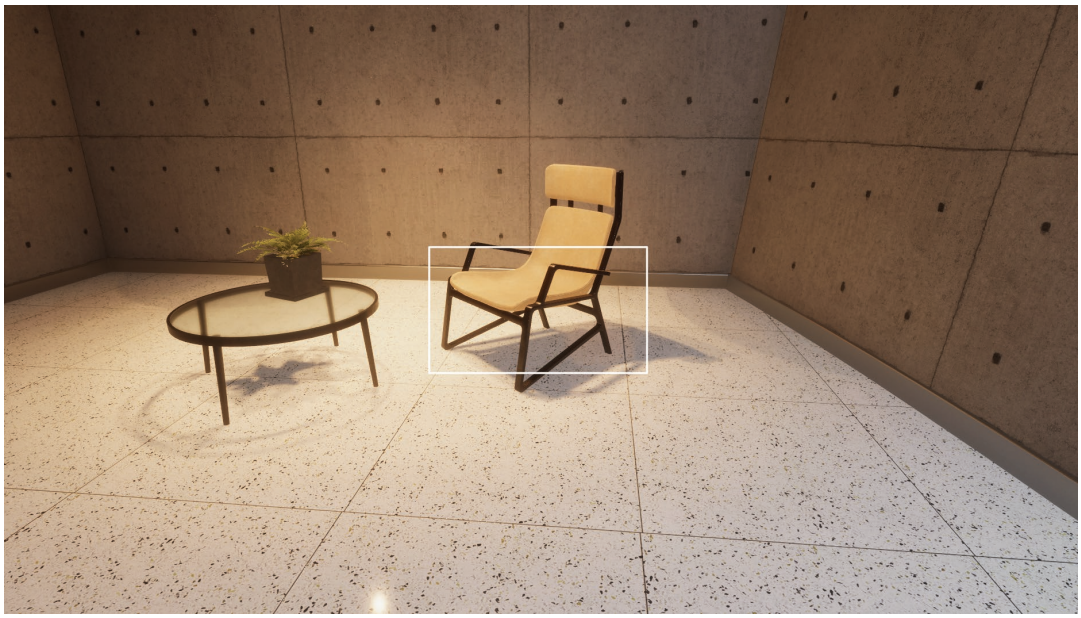
マルチサンプルアンチエイリアス (MSAA)

マルチサンプルアンチエイリアス (MSAA) は、PC ゲームで一般的なアンチエイリアス手法です。個々のポリゴンのエッジを滑らかにする高性能ハードウェア向けの手法で、Unity ではフォワードレンダリングを使用する場合のみ機能します。最新の GPU は、2x、4x、8x MSAA サンプルに対応しています。

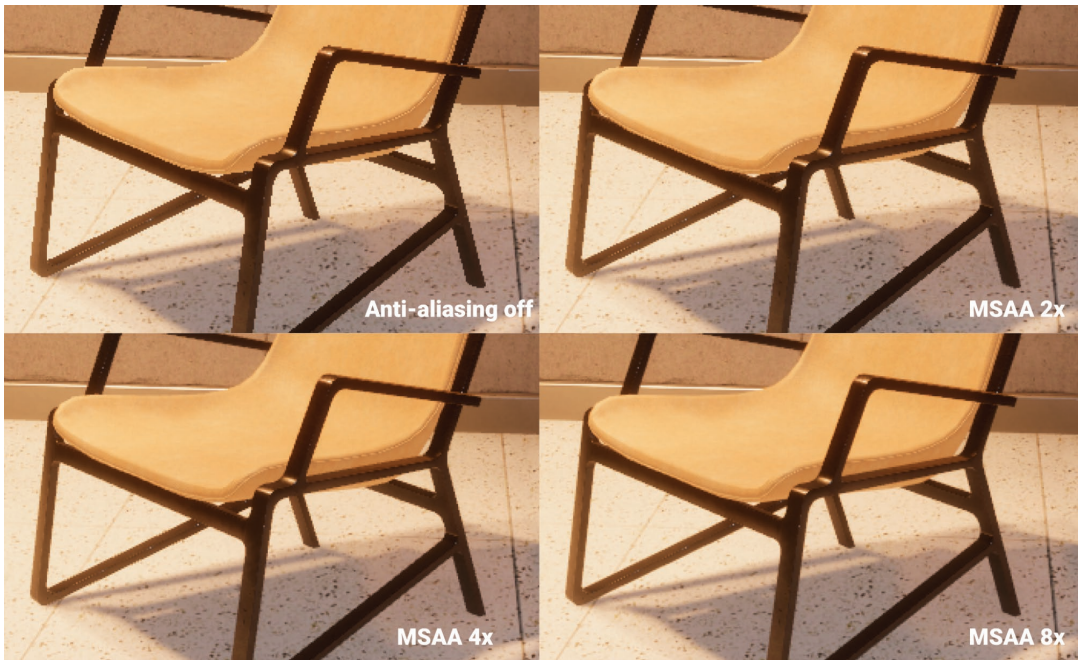
MSAA のクオリティー設定

アクティブなパイプラインアセットで、Lit Shader Mode を **Forward Only** に設定します。続いて、**Multisample Anti-aliasing Quality** で、**MSAA 2x**、**MSAA 4x**、**MSAA 8x** のいずれかを選択します。値が大きいほどアンチエイリアスの質が上がりますが、動作は遅くなります。

これは、カメラビューでズームインしたときにより顕著になります。



元のシーン



MSAA 設定を適用した画像

次の制約事項にご注意ください。

— MSAA は、シーンのジオメトリをテクスチャーに保管するディファードシェーディングの G-buffer では利用できません。そのため、ディファードシェーディングでは、いずれかのポストプロセスでのアンチエイリアス手法 (後述) を使用する必要があります。

— MSAA では、ポリゴンのエッジのエイリアシングのみを処理するため、シャープなスペキュラーライトが届く特定のテクスチャーやマテリアルに対するエイリアシングについては防ぐことができません。これが問題になる場合は、MSAA と他のポストプロセスのアンチエイリアス手法 (後述) とを組み合わせる必要があります。

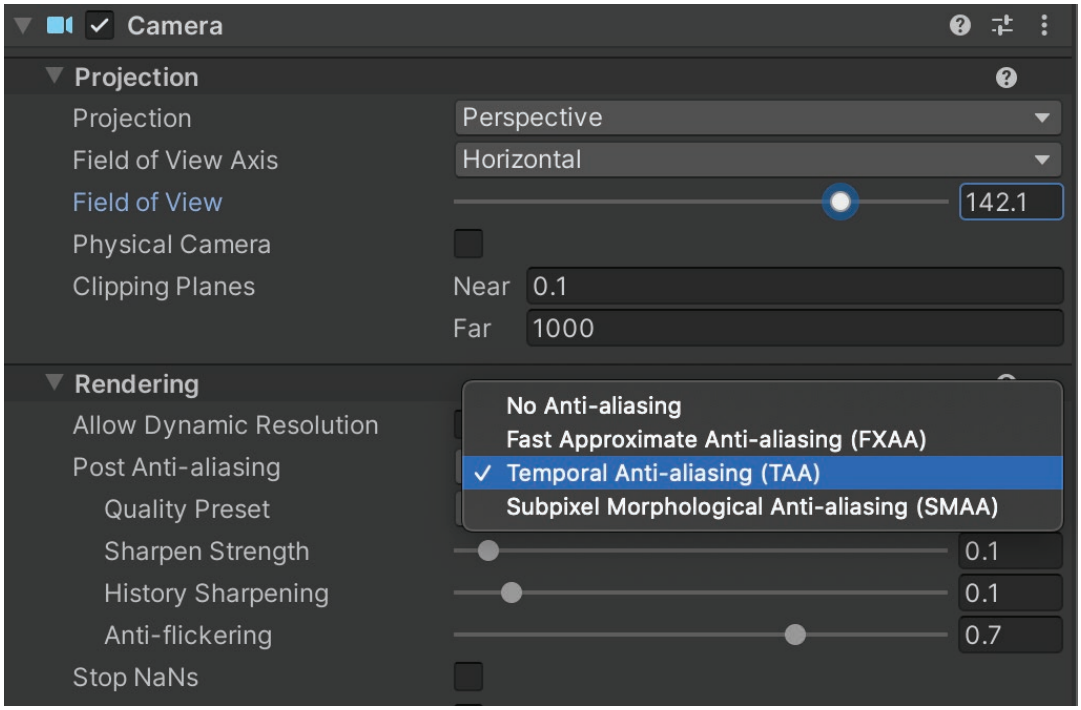
ポストプロセスでのアンチエイリアス

Post Anti-aliasing (ポストアンチエイリアス) を次のように設定して、ポストプロセスの手法でカメラにアンチエイリアスを適用することもできます。

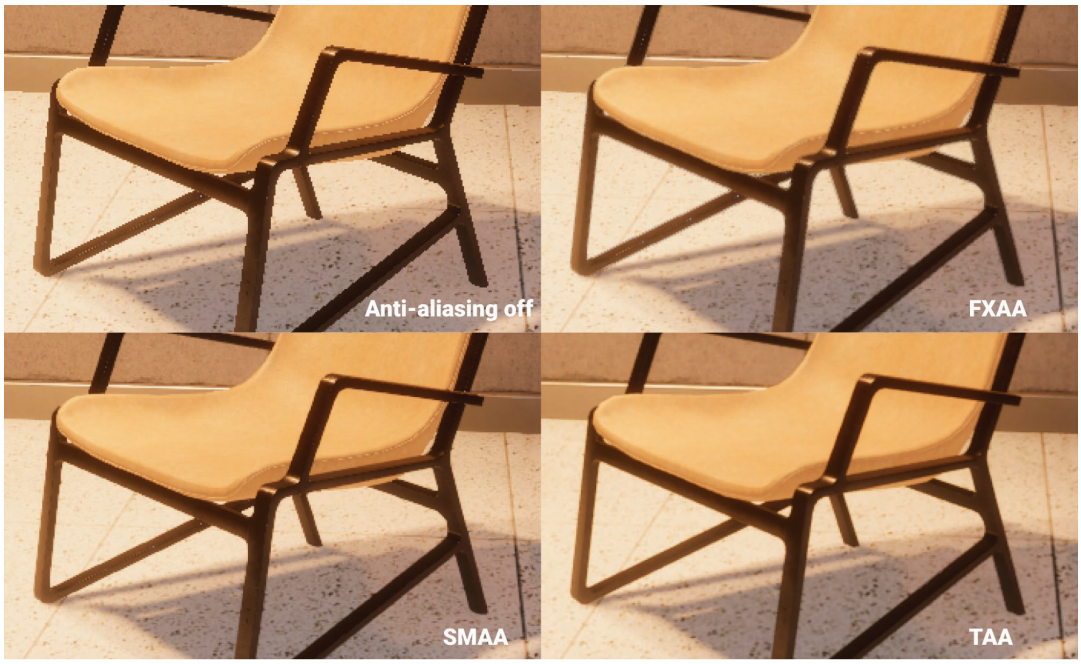
— **Temporal Anti-aliasing (TAA)** では、過去と現在のフレームからの情報を組み合わせ、現在のフレーム内のジャギーを取り除きます。この設定を使用するためには、モーションベクトルを有効にする必要があります。大半の場合 TAA は良い成果をもたらしますが、状況によってゴーストアーティファクトが発生することがあります (コントラストのある表面の手前でゲームオブジェクトが高速で移動した場合など)。HDRP10 では、一般的な TAA アーティファクトを削減するための改良が加えられています。Unity の実装ではゴーストの発生が抑制され、シャープネスが改善されたほか、他のソリューションで見られるちらつきが防止されています。

— **Fast Approximate Anti-aliasing (FXAA)** は、ハイコントラストの領域間のピクセルをブレンドするスクリーンスペース アンチエイリアス アルゴリズムです。比較的高速で、大規模な演算能力も不要ですが、画像の全体的なシャープネスは落ちる場合があります。

— **Subpixel Morphological Anti-aliasing (SMAA)** では、画像内の境界を検出してから、特定のパターンを見つけてブレンドします。これによって、FXAA よりシャープな結果が生成されます。フラットスタイルやカートゥーン調のスタイル、クリーンなアートスタイルに適しています。



ディファードシェーディングを使用する際は、カメラのポストアンチエイリアシングを調整します。



ポストプロセスのアンチエイリアス: FXAA、SMAA、TAA 各設定の結果を比較。

注意:ポストプロセスでのアンチエイリアスとマルチサンプルアンチエイリアスを組み合わせる際は、レンダリングコストに注意してください。いつでもビジュアルクオリティーとパフォーマンスのバランスがとれるように、プロジェクトは常に最適化を心がけるようにしましょう。

Post-processing

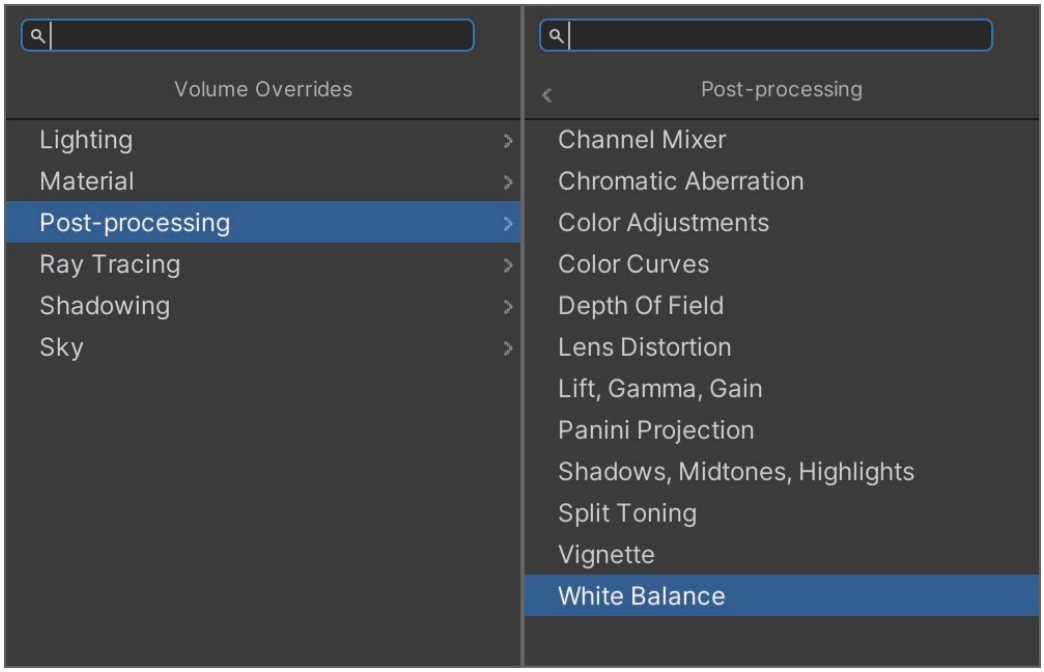
最新のハイエンドグラフィックスにおいては、ポストプロセスを行わずして完成することはないとも言えます。必ずしもプロセッシングで問題を修正できるわけではありませんが、シネマティックな効果を高めるフィルターやフルスクリーンのイメージエフェクトを使用せずに画像をレンダリングすることは、ほとんど考えられません。そのため、HDRP には組み込みのポストプロセスエフェクトがバンドルされています。

ポストプロセスエフェクトを使用すると、よりシネマティックなレンダリングが可能になります。

HDRPのプロセッシングでは、ボリュームシステムを使用してイメージエフェクトをカメラに適用します。オーバーライドを追加する方法を理解すれば、ポストエフェクトを適用するプロセスも理解しやすくなるはずです。

Post-processing のオーバーライド

色やコントラストを制御するこうした Post-processing のオーバーライドの多くは、機能が重複しています。適切な組み合わせを見つけるためには、ある程度の試行錯誤が必要です。



Post-processing のオーバーライド

すべての エフェクトを利用する必要はありません。目標とするビジュアルを生み出すために必要なオーバーライドのみを追加し、その他は無視します。

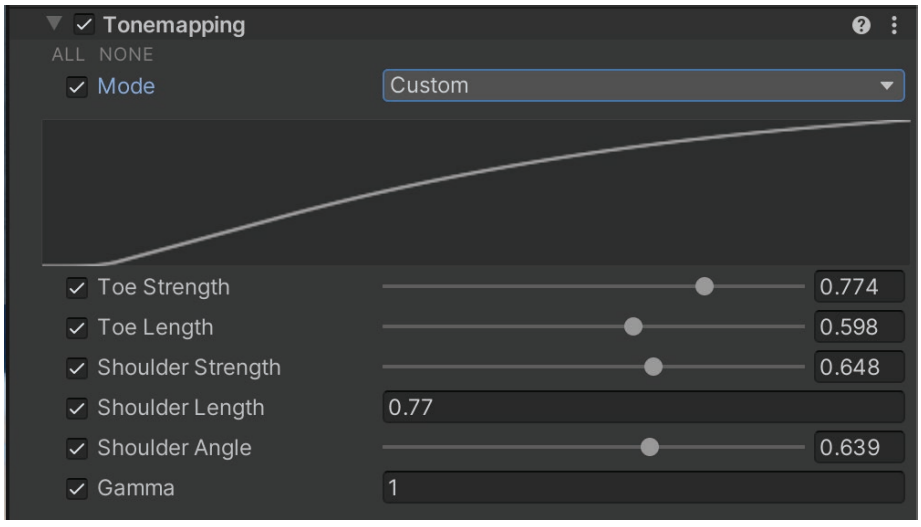
使用例については、HDRP サンプルシーンの Volumes (Unity Hub で入手可能) を参照してください。

トーンマッピング

トーンマッピングとは、ハイダイナミックレンジ（HDR）の色を、使用する画面の狭い ダイナミックレンジ にマッピングする手法です。レンダリングでのコントラストやディテールを改善できます。

ACES と Neutral のトーンマッピングの比較

見た目をフィルム調にしたい場合は、**Mode** を業界標準の **ACES** (Academy Color Encoding System) に設定します。彩度やコントラストを落とす場合は、**Neutral** を選択します。経験豊富なユーザーであれば、**Custom** を選択し、トーンマッピングカーブを自分で定義することもできます。



カスタムカーブを使用したトーンマッピング

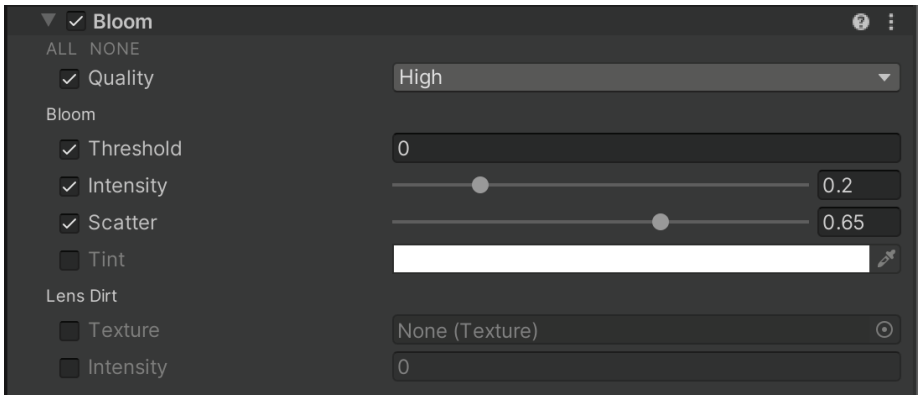
Shadows、Midtones、Highlights

Shadows、Midtones、Highlights のオーバーライドでは、レンダリング対象のシャドウ、中間調、ハイライトそれぞれの色調と色の範囲を制御できます。それぞれのトラックボールを使用して、画像内のそれぞれの特性を調整できます。その後、色補正をクリップまたはプッシュしすぎないように、**Shadow** と **Highlight Limits** を使用します。

Shadows、Midtones、Highlights

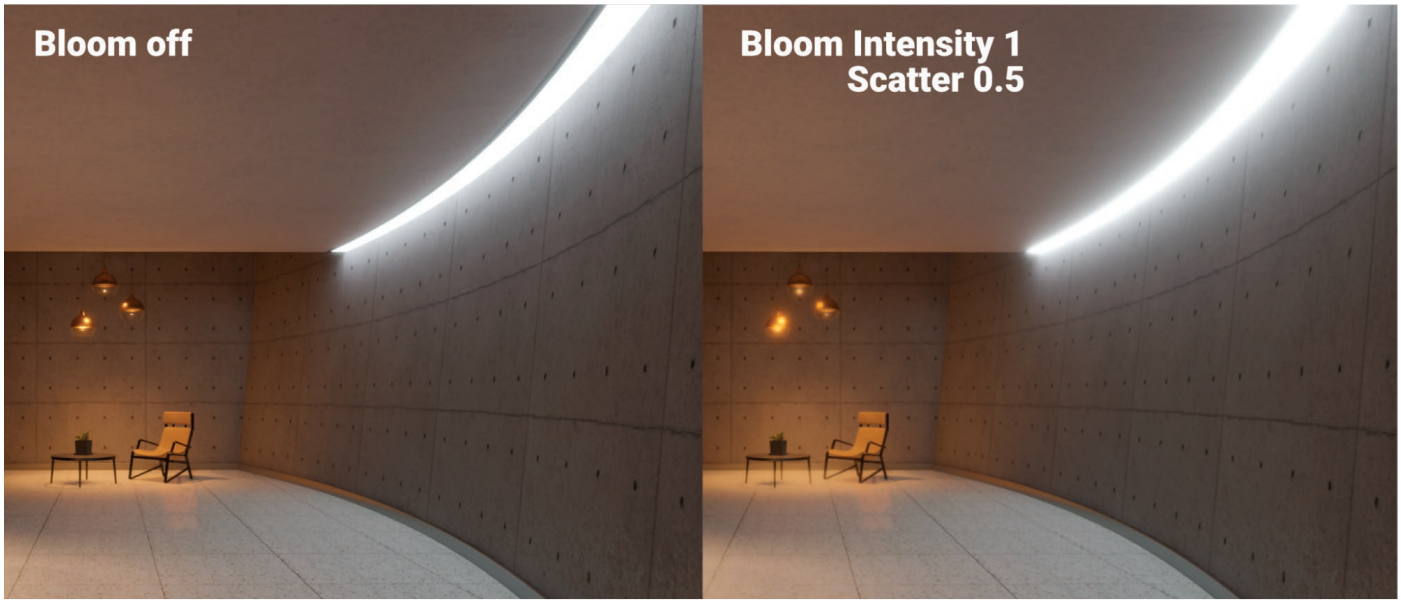
ブルーム

Bloom では、光源の周囲に発生する光のにじみの効果を生成できます。これにより、強烈に輝く、まばゆい光の印象を生み出すことができます。



Bloom のオーバーライド

Intensity と**Scatter** を設定することで、ブルームのサイズと明度を調整します。**Lens Dirt** は、ブルーム効果を散乱させる汚れやちりのテクスチャを適用します。明るくないピクセルの鋭さを維持するには **Threshold** を使用します。



Bloom の効果

被写界深度

Depth of Field (被写界深度) は、実際のカメラレンズの焦点特性のシミュレーションを行います。カメラの被写界深度より近い、または遠いオブジェクトは、ぼやけて見えます。

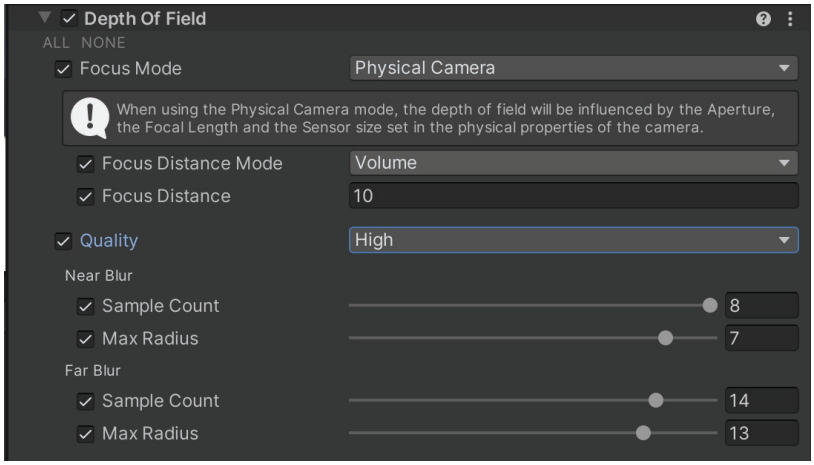
焦点距離の設定は以下より行えます。

Manual Ranges フォーカスモードのボリュームオーバーライド: ここでは、ボリューム自体が焦点距離を制御します。例えば、場所によって意図的にぼかすのに使えます (例: 水中シーンなど)。

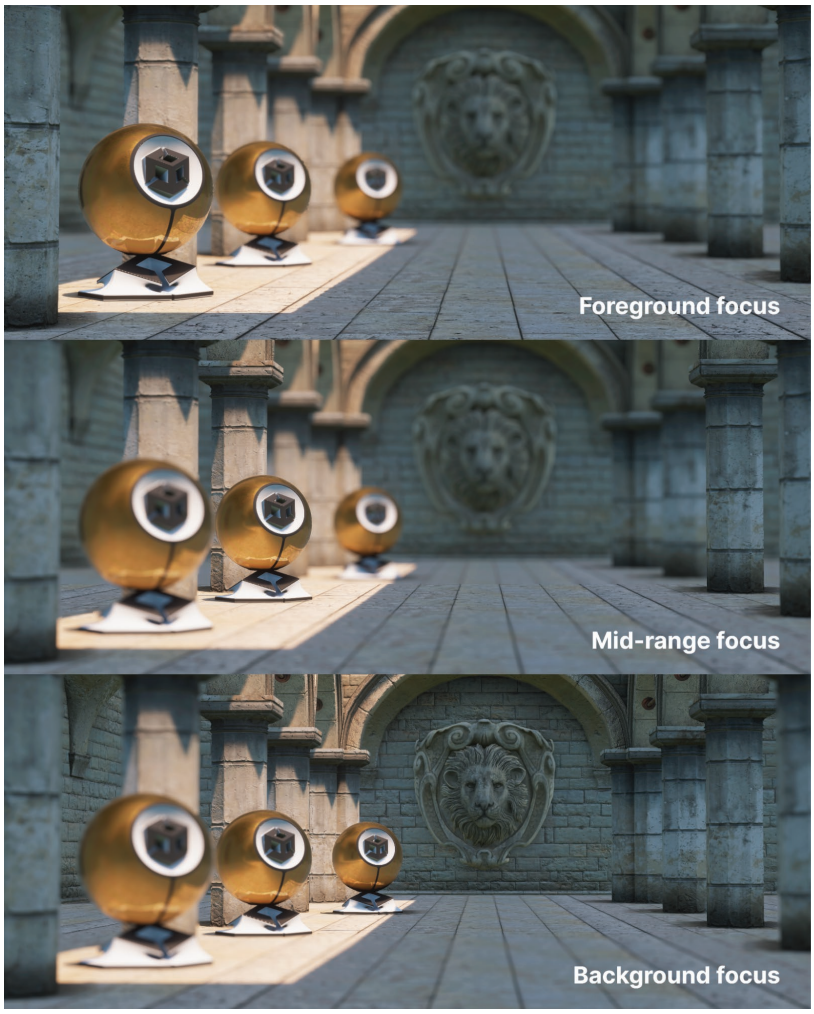
— Cinemachine カメラを Volume Settings 拡張機能 で使用:これを使うと、ターゲットを追跡して自動フォーカスすることができます。

— **Physical Camera** フォーカスモードでの **Physical Camera** プロパティ:これにより、Camera コンポーネントの **Focus Distance** パラメーターをアニメーション化できます。

Depth of Field が有効な場合は、ピントが合っていない ボケ と呼ばれるエフェクトが、画像の明るいエリアを中心に現れます。カメラの開口形状を調整し、ボケの見え方を変えることができます (前述の Physical Camera のその他のパラメーターを参照)。

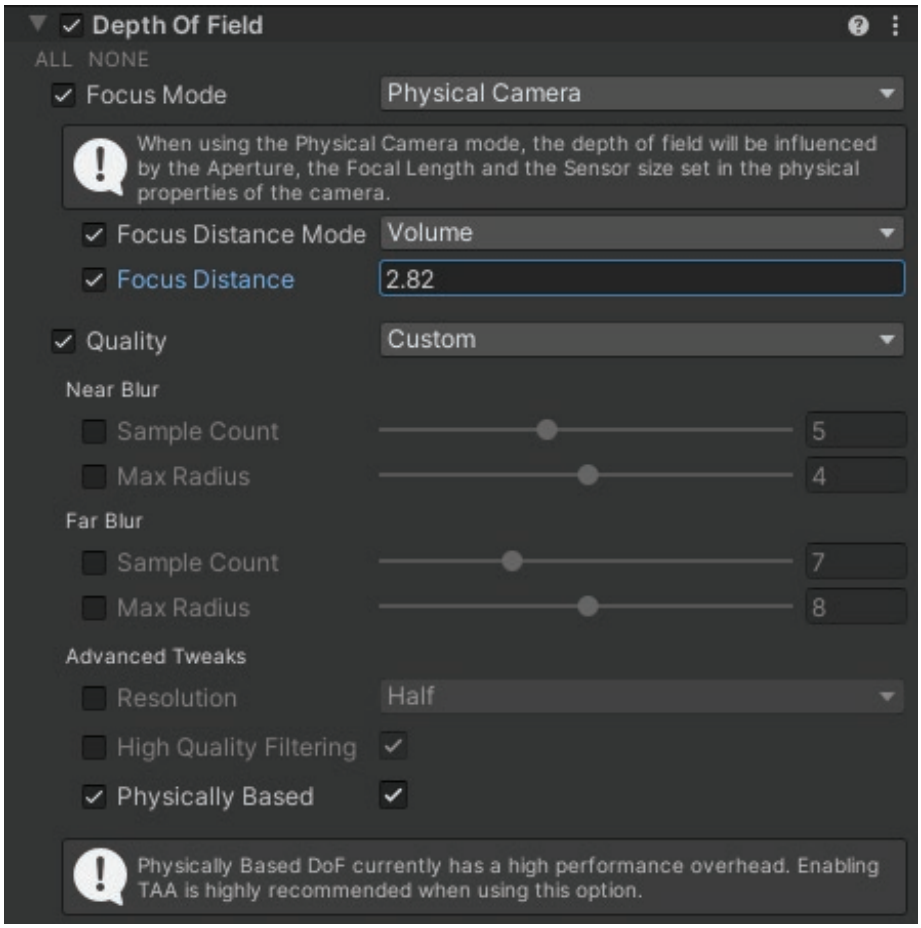


Depth of Field のオーバーライド



Depth of Field では、実際のカメラの被写界深度のシミュレーションを行います。

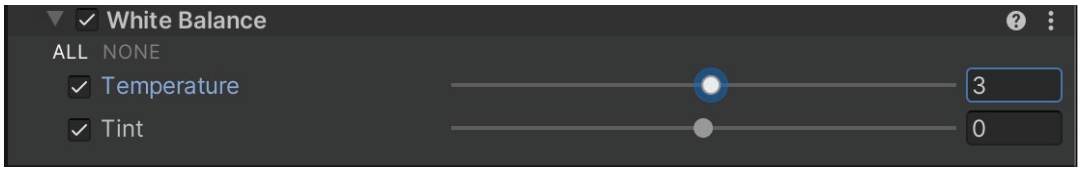
シネマティクスやオフラインレンダリングでは、Additional Settings と **Custom Quality** を有効にすることで、より高コストな **物理ベース** の被写界深度を選択することができます。



White Balance

White Balance のオーバーライドを使用すると、最終的な画像で白色が適切にレンダリングされるようにシーンの色を調整できます。**Temperature** で、黄色（暖色）と青（寒色）の間を調整することもできます。**Tint** では、緑とマゼンタの間で色かぶりを調整します。

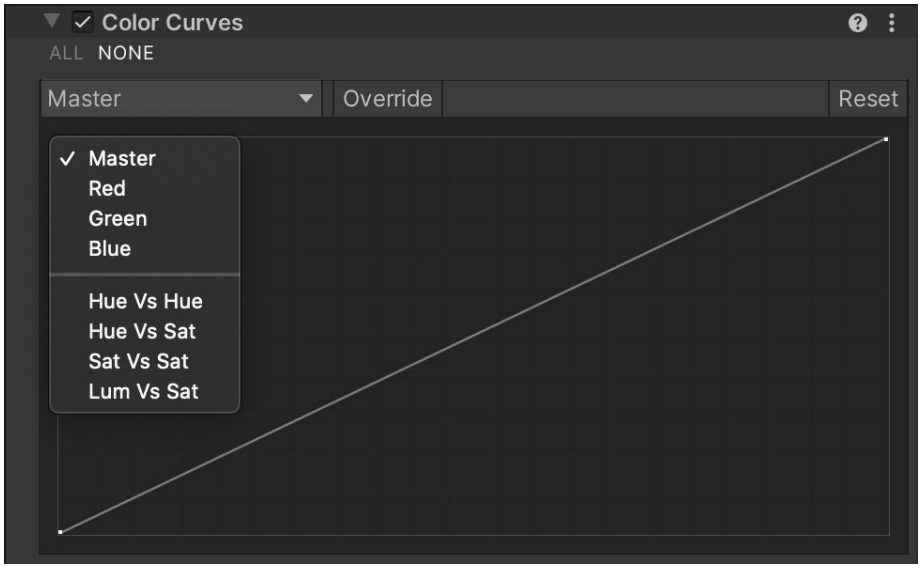
この HDRP サンプルプロジェクトでは、各部屋のローカルボリュームに White Balance のオーバーライドが含まれています。



White Balance

カラーカーブ

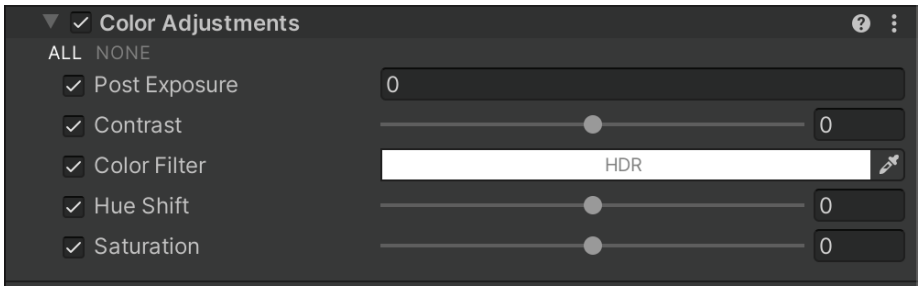
グレーディングカーブを使用して、色相、彩度、明るさの特定の範囲を調整できます。8　つのグラフのいずれかを選択し、色とコントラストをリマップします。



カラーカーブ

色の調整

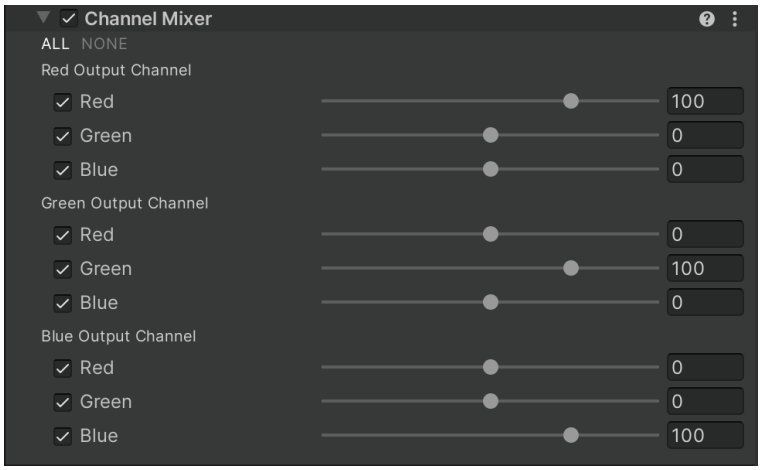
この効果を使用して、最終的にレンダリングされる画像の全体的なトーン、明度、色相、コントラストを調整します。



色の調整

Channel Mixer

Channel Mixer では、カラーチャンネルの “組み合わせ“ に、さらに別のカラーチャンネルを適用できます。RGB 出力を選択してから、いずれかの入力の影響度を調整します。たとえば、Red Output Channel で Green の影響度を上げると、画像内のすべての緑色の部分が赤みがかって表示されます。

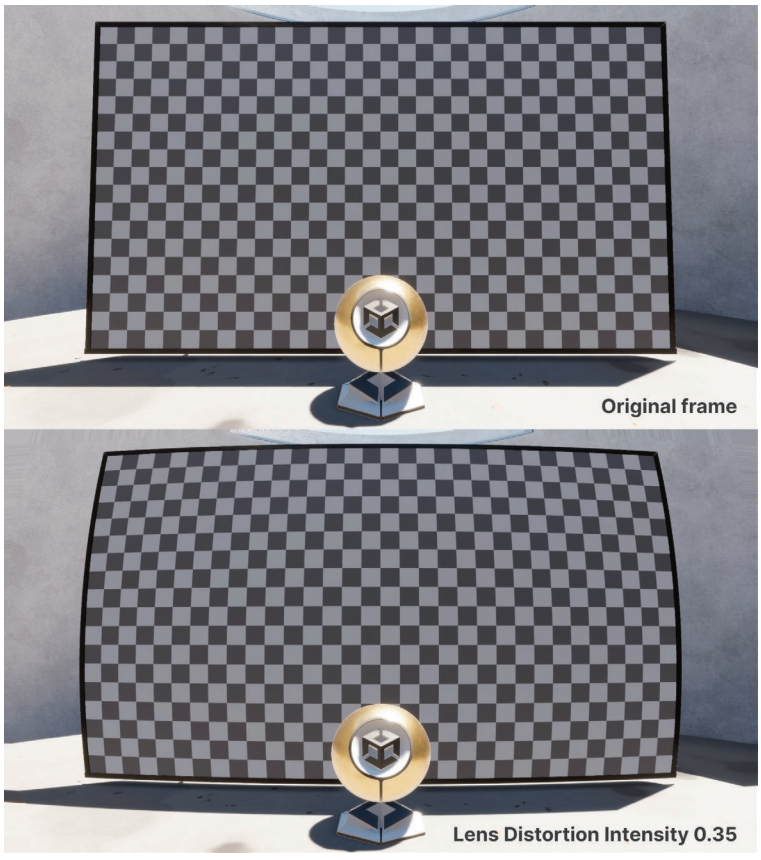


Channel Mixer

Lens Distortion

Lens Distortion では、現実世界のレンズの製造過程で発生する歪みによって生じる放射パターンのシミュレーションを行います。これを設定すると、特にズームレンズや広角レンズを使用したときに、直線がわずかに曲がって見えます。

この “魚眼“ 効果は、特定のムードやスタイルを作り出すのに役立ちます。例えば、フラッシュバックのカットシーケンスを強調したり、精神状態の変化を表すのに使うことができます。また、緊張感や方向感覚の喪失を強調することもできます。



Lens Distortion を使用すると、放射パターンの歪みを画像に適用します。

ビネット

ビネットでは現実世界の写真に生じる効果を再現し、画像の端部の明度や彩度を下げます。この現象は、広角レンズを使用した場合や、レンズフードや重ね付けしたフィルターリングによって光が遮られた場合に発生することがあります。見る人の視線を画面中央に集めるために、この効果が使用されることもあります。

ビネットを使用すると、フレームの端部が暗くなります。

モーションブラー

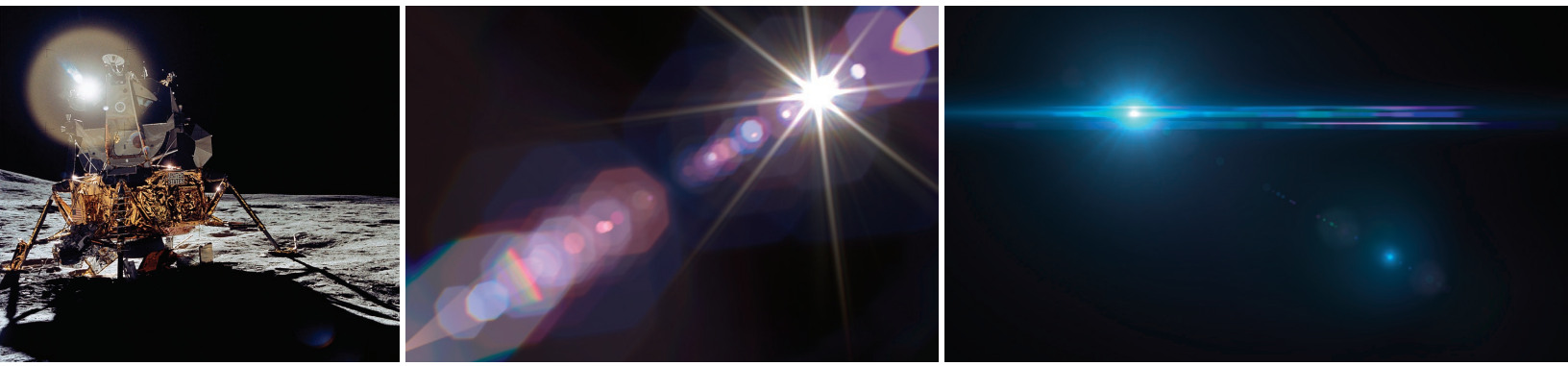
現実世界では、カメラの露出時間よりも速く移動する物体を撮影すると、線状になったりぼやけたりした状態で写ります。モーションブラーのオーバライドで、このような効果のシミュレーションを行うことができます。

パフォーマンスコストを最小限に抑えるためには、Sample Count を減らして Minimum Velocity を上げ、Maximum Velocity を下げます。Additional Properties で Camera Clamp Mode のパラメーターを下げることもできます。

レンズフレア

レンズフレアは、カメラのレンズに明るい光が当たったときに現れるアーティファクトです。1 つの明るい輝きとして現れることもあれば、カメラの絞りと同じ多角形をした複数色のフレアとして現れることもあります。

現実世界において、フレアは好ましくない効果を持ちますが、ナラティブや芸術面の演出で役に立つことがあります。例えば、強いレンズフレアは、プレイヤーの注意を引いたり、状況や場面の雰囲気を変えたりするのに使えます。レンズフレアの物理的な側面については、こちらの Wikipedia の記事 を参照してください。



実際の写真のレンズフレアです (出典:Wikipedia)。

レンズフレアはレンダリングプロセスの後段でレンダリングされるため、実質的にはポストプロセスエフェクトです。

HDRP には、**Lens Flare (SRP)** (レンズフレア (SRP)) コンポーネントと Unity 6 の新機能である **Screen Space Lens Flare** (スクリーンスペースレンズフレア) という 2 種類の補完的なレンズフレアが用意されています。

Lens Flare (SRP) コンポーネント

Lens Flare (SRP) コンポーネントは、シーン内の特定のオブジェクト (通常は光源や明るい要素) から位置ベースのフレアを作成します。これを使用するには、任意のオブジェクトに Lens Flare (SRP) コンポーネントを追加し、Lens Flare Data (レンズフレアデータ) アセットを割り当てます。

各 Lens Flare Data アセットは、ポリゴン、円、カスタム画像などの形状を持つ複数の要素で構成されています。これらの形状を重ねて複雑なエフェクトを作ることができます。ライトにアタッチすればライトの色を継承できるため、同じフレアアセットを異なる光源で再利用できます。

Lens Flare (SRP) は、強度、スケール、オクルージョンのパラメーターを制御します。

レンズフレアのサンプルには、リアルなものから様式化されたものまで、さまざまなプリセットが用意されています。

Screen Space Lens Flare

Unity 6 では、新しいポストプロセス手法として、Screen Space Lens Flare オーバーライド (SSLF) が導入されました。これは、明るいスペキュラーハイライトやエミッシブメッシュなど、明るい表面からフレアを生成できます。

SSLF は、Volume オーバーライドとして使用できます (**Post-processing > Screen Space Lens Flare**)。この種のフレアエフェクトは、発光が強い表面、鏡面反射、画面上のライトなど、シーン内の明るい領域で機能します。HDRP はこれらの明るい領域を抽出し、伸び、ぼかし、色収差などのエフェクトを適用してフレアを作成します。

Screen Space Lens Flare は、ブルームエフェクトと同じバッファを使用して数種類のフレアを作成します。

- 通常フレアは、画面の明るい領域から生じる変形したハイライトです。
- 反転フレアは、通常フレアを上下逆にして反転させたものです。
- 歪曲フレアは、通常フレアを極座標変換して、円形のカメラレンズを模倣したものです。
- ストリークは、アナモルフィックレンズを模倣するために一方向に引き延ばされたフレアです。

Screen Space Lens Flare を有効にするには、まず HDRP アセットと Frame Settings でストリークをアクティベートします。次に、**Screen Space Lens Flare** オーバーライドを Volume に設定し、その Intensity を 0 より大きく設定します。

すでにアクティブな Bloom オーバーライドがある場合、Screen Space Lens Flare が表示されるには Intensity (強度) が 0 より大きい必要があることに注意してください。

Screen Space Lens Flare を Lens Flare (SRP) コンポーネントと組み合わせて、より細かく制御できます。どちらのシステムもポストプロセスエフェクトとしてレンダリングされます。レンズフレアのオクルージョンは、雲、ボリュメトリックフォグ、水などの環境要素と統合されます。

Screen Space Lens Flare は、ストリーク (筋) を加えることができます。

スタートガイド

独自のレンズフレア作成を始めるには、Package Manager から High Definition RP のサンプルをインストールするのが最適です。これにより、定義済みの Flare アセットのセットがプロジェクトに追加されます。サンプルにはレンズフレアをざっと見るためのテストシーンもあるので、レンズフレアを自分でビルドする際にも役立ちます。

定義済みのフレアデータアセットは、よく使用されるフレアエフェクトを再現します。**Directional Light Sun** と **Point Light** のシーンで、各レンズフレアがどのように組み立てられているかを詳しく見てみましょう。

— 各光源には、独自の Lens Flare (SRP) コンポーネントがあります。

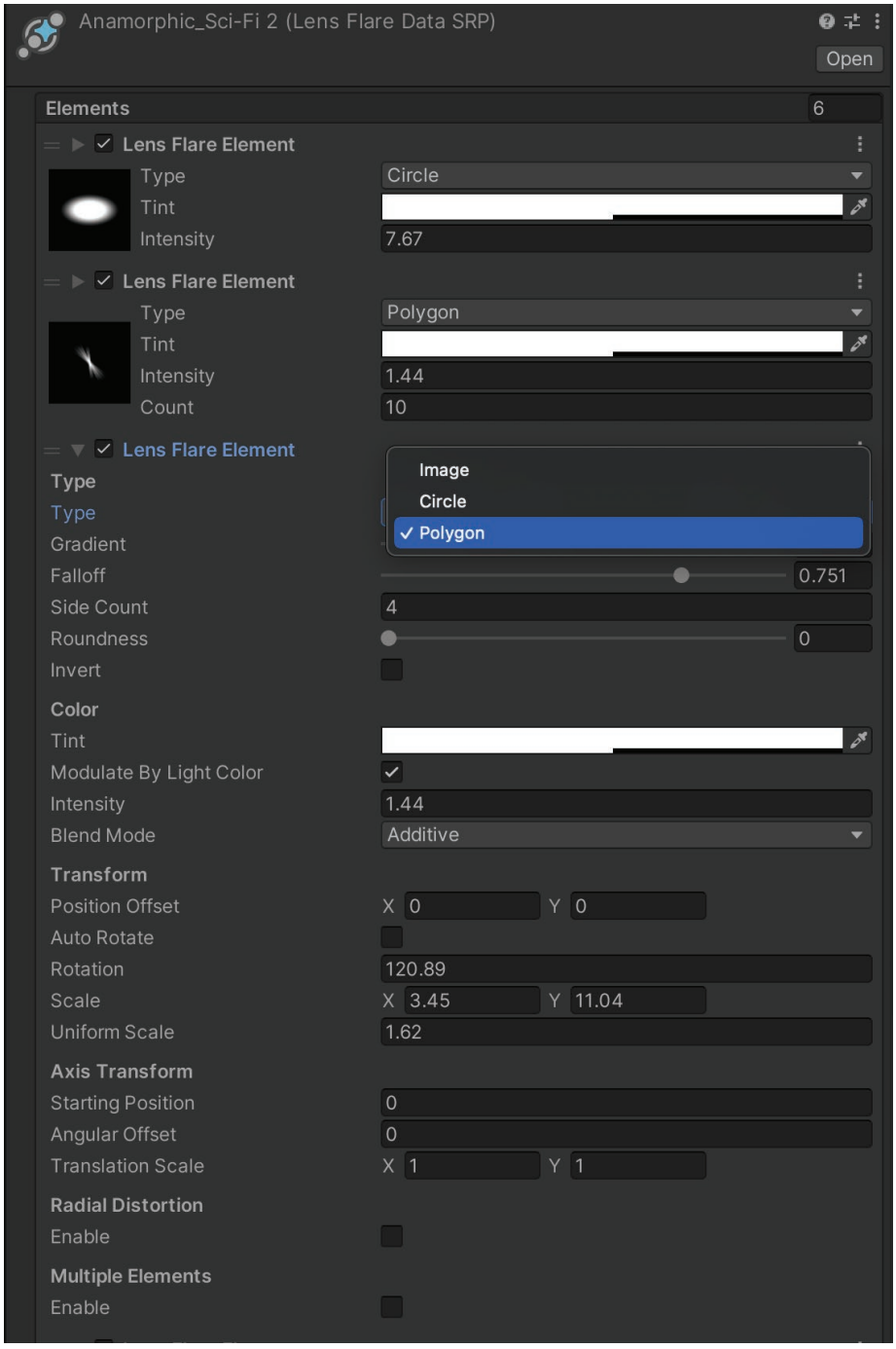
— このコンポーネントは、プロジェクト内のレンズフレアのデータアセットを参照します。

レンズフレアのシーンを閲覧し、プリセットに基づいて独自のフレアを構築するために活用できます。リアルなものから様式化されたものまで幅広く用意されています。

プロジェクトで **Directional Light Sun** を選択すると、ライトに接続されたコンポーネント Lens Flare (SRP) と、レンズフレアのデータアセットが見つかります。アセットを変更すると、さまざまなフレア効果を観察できます。

レンズフレアを組み合わせて、思い通りのエフェクトを実現しましょう (説明のために強度を上げています)。

レンズフレアがどのように機能するかについては、こちらの SIGGRAPH の講演(日本語字幕あり) で詳しく学ぶことができます。



カスタムフレアエフェクトの独自のライブラリを作成します。

動的解像度

アップスケーリングは、ゲームを低解像度でレンダリングしてから高解像度にスケーリングすることで、パフォーマンスを向上させる技術です。これにより、見栄えを保ちながら GPU の負荷が軽減され、ディテールを大きく損なうことなく、より高いフレームレートを実現できます。

動的解像度 は、GPU のパフォーマンスに基づいてレンダリング解像度をリアルタイムで調整する HDRP の機能です。有効にすると、可能な限り最大の解像度でレンダーターゲットが割り当てられ、さまざまな解像度でレンダリングするようにビューポートが再スケーリングされます。各フレームの最後に、レンダリングされた画像がディスプレイのネイティブ解像度に合わせてアップスケールされます。

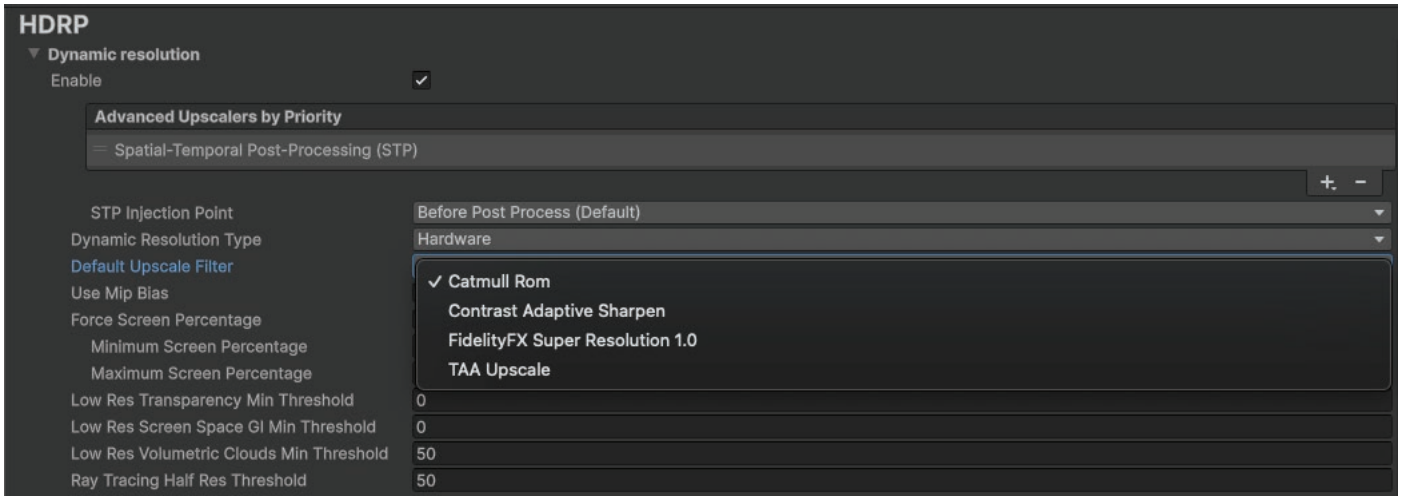
HDRP はハードウェアとソフトウェアの両方の動的解像度をサポートしています。ハードウェア動的解像度は、コンソールや、DirectX 12、Vulkan、Metal を使用する PC などのプラットフォームで利用できます。プラットフォームがハードウェア動的解像度をサポートしていない場合、HDRP はデフォルトでソフトウェア動的解像度に設定されます。

動的解像度は、HDRP アセットの **Rendering > Dynamic Resolution** で有効にします。次に、カメラの **Rendering** 設定にある **HDRP Dynamic Resolution** を有効にします。

HDRP では、さまざまなハードウェアでのアップスケーリング動作の制御が改善されています。

— **Advanced Upscalers by Priority**:このオプションでは、高度なアップスケーラー (DLSS、FSR、STP) の優先順位を指定できます。HDRP は、プラットフォームでサポートされている最も優先度の高いアップスケーラーを選択します。使用可能なものがない場合は、**Default Upscale Filter** にフォールバックします。

— **Default Upscale Filter**:このオプションにより、高度なアップスケーラーが利用できない場合のフォールバックアップスケーリング方式 (TAAU、Catmull-Rom、CAS など) が設定されます。



新しいユーザーインターフェースでは、アップスケーリングのフォールバック設定を指定できます。

Unity HDRP には、いくつかのアップスケールフィルター方式が用意されています。

Catmull-Rom

Catmull-Rom は 4 つのバイリニアサンプルを使用します。これは最小限のリソースで済みますが、アップスケーリング後に画像がぼやけることがあります。Catmull-Rom は依存関係を持たず、ポストプロセスパイプラインの最後に実行されます。

Contrast Adaptive Sharpen (CAS)

Contrast Adaptive Sharpen (CAS; コントラスト適応シャープニング) は、AMD FidelityFX CAS を使用します。この方式では、積極的なシャープニングのステップで鮮明な画像が生成されます。動的解像度の画面比率が 50% 未満の場合は、このオプションを使用しないでください。AMD FidelityFX™ Super Resolution (FSR) およびコントラスト適応シャープニングの詳細については、AMD FidelityFX Super Resolution (FSR) を参照してください。

コントラスト適応シャープニング (CAS) は依存関係がなく、ポストプロセスパイプラインの最後に実行されます。

NVIDIA DLSS (NVIDIA RTX GPU および Windows 用)

NVIDIA DLSS は、Unity 6 の HDRP でネイティブにサポートされています。NVIDIA DLSS は、AI を使用して追加フレームを生成し、ネイティブ解像度に匹敵する品質の画像を構築する一連のニューラルレンダリング技術です。その際、従来の方法でレンダリングするピクセルは全体のごく一部です。

リアルタイムレイトレーシングと NVIDIA DLSS により、NVIDIA RTX GPU 上でより高いフレームレートと解像度で動作する美しいワールドを生み出すことができます。DLSS はまた、従来のラスタライズされたグラフィックスのパフォーマンスを大幅に向上させます。詳しくは、NVIDIA の Unity 開発者ページ と Unity のドキュメント を参照してください。

DLSS テクノロジーにより、24 Entertainment の『Naraka:Bladepoint』のようなゲームが 4K で動作します。

AMD FSR (クロスプラットフォーム)

AMD FidelityFX™ Super Resolution (FSR) には、HDRP に対するサポートが組み込まれています。HDRP アセットとカメラで動的解像度を有効にし、アップスケールフィルターオプションで **FidelityFX Super Resolution 2.0** を選択すると、FSR を使用できます。

AMD FidelityFX Super Resolution (FSR) は、低解像度入力から高解像度フレームを生成するための、オープンソースかつ高品質なソリューションです。特に高品質なエッジを作成することに重点を置いた一連のアルゴリズムを使用しており、ネイティブ解像度で直接レンダリングした場合に比べてパフォーマンスが向上します。

FSR は、ハードウェアレイトレーシングのような、高コストなレンダリング処理において、実用的なパフォーマンスを可能にします。詳しくは、AMD の ウェブページ と Unity フォーラム を参照してください。

Steam で入手可能な *Spaceship* デモで FSR 技術を試してみることができます。

TAA アップスケール (クロスプラットフォーム)

Temporal Anti-Aliasing (TAA) アップスケールは、HDRP における通常のアンチエイリアスと並行して動作し、時間的統合を利用して画像の鮮明さを向上させます。HDRP はこのアップスケールフィルターをポストプロセスの前に、TAA と同時に適用します。

TAA アップスケールは他のアンチエイリアス方式と互換性がないため、この機能を有効にするときは TAA を使用する必要があります。また、動的解像度が有効になっていると、画面解像度が 100% であってもアクティブのままになります。HDRP アセットでアップスケール方式を設定したり、コードで動的に調整したりできます。

詳細については、Notes on TAA Upscale (TAA アップスケールに関する注記) を参照してください。

TAA アップスケールを使用した左側の画像 (A) は、よりシャープで高精細に見えます。Catmull-Rom アップスケール手法を使用した右側の画像 (B) は、よりソフトであまり精細ではありません。

Spatial-Temporal Post-Processing (クロスプラットフォーム)

Spatial-Temporal Post-Processing (STP) は、Unity 6 の新しいビルトインアップスケーラーです。STP は低解像度でレンダリングした後、空間データと時間データを使用して高解像度の画像を再構築します。これにより、視覚的忠実度を維持しながら GPU の負荷を軽減できます。

STP では、空間データと時間データの *両方* を使用してギザギザのエッジを滑らかにするアンチエイリアス手法が使用されます。1 フレーム (空間的) 内で細部をシャープ化し調整しながら、複数フレーム (時間的) にわたって動きを安定させることができます。これにより、アーティファクトの少ない鮮明な画像が得られます。

STP は、画質を維持したまま GPU 負荷を軽減します。

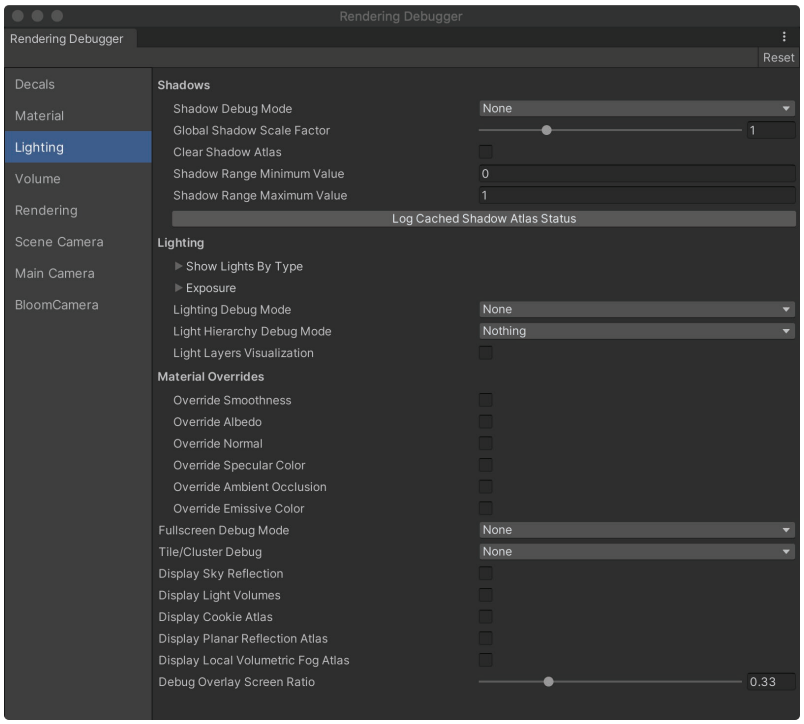
STP は、プラットフォームに基づいて設定を自動的に調整します。高性能なデバイス (PC やコンソール) では、アップスケール時により高品質な画像フィルタリングとデリンギングロジックを使用します。モバイルデバイスでは、より高速なフィルタリング方式を選択してパフォーマンスへの影響を最小限に抑えながら、DLSS2、FSR2、XeSS と同等の視覚的な品質を実現します。

HDRP アセット で **Render Scale** を調整することで、ターゲットプラットフォームに応じて STP のパフォーマンスと品質のバランスを微調整できます。

STP には Shader Model 5.0 が必要で、TAA (Temporal Anti-Aliasing) でのみ機能することに注意してください。ハードウェア動的解像度がサポート可能な場合を除き、STP は動的解像度をサポートしません。ハードウェア動的解像度が利用できない場合は、Render Scale を固定値に設定する必要があります。

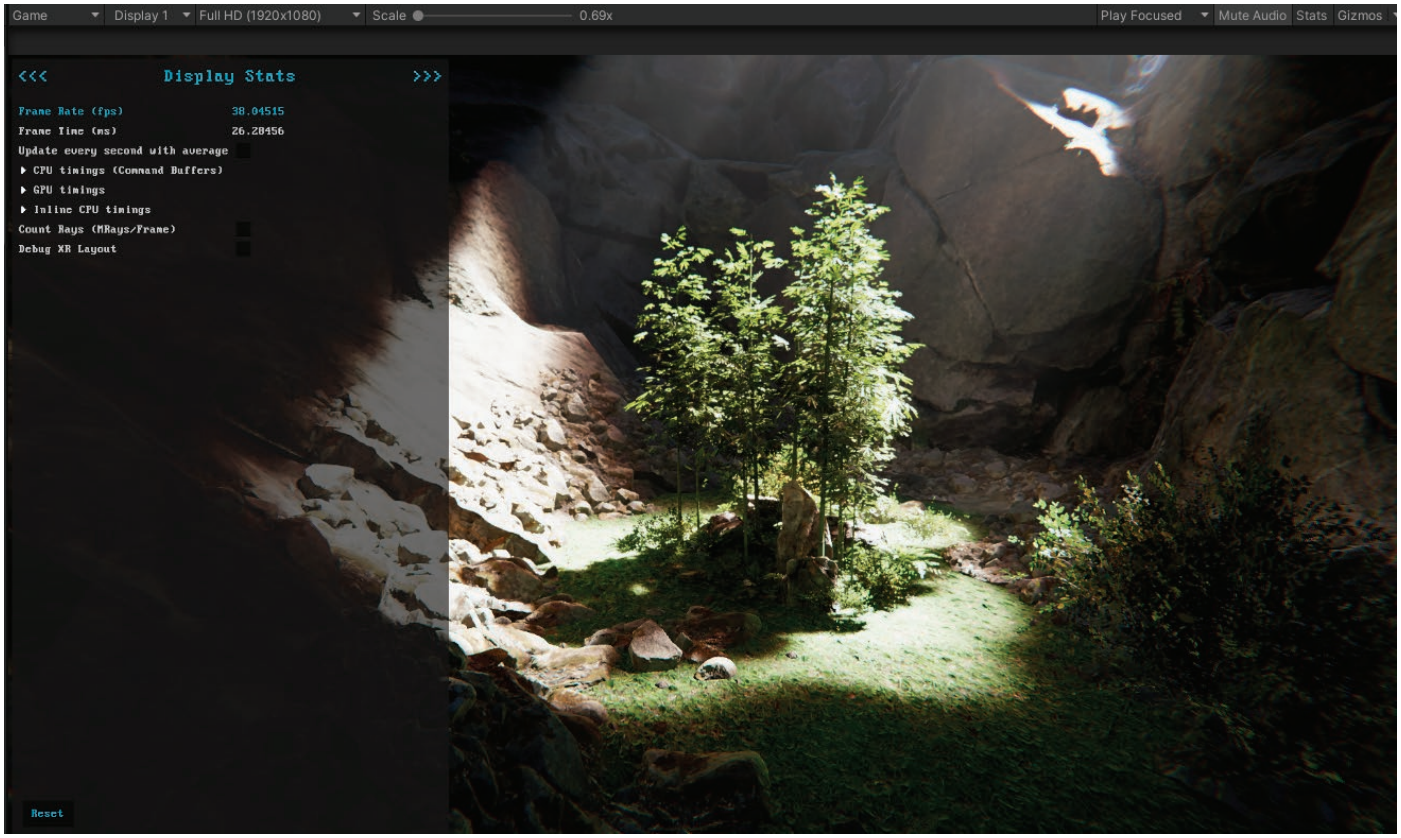
レンダリングデバッガー

Rendering Debugger ウィンドウ (**Window > Analysis > Rendering Debugger**) には、Scriptable Render Pipeline 用のデバッグツールと可視化ツールがあります。左側はカテゴリ別に分類されています。各パネルでは、ライティング、マテリアル、ボリューム、カメラなどの問題を分割できます。



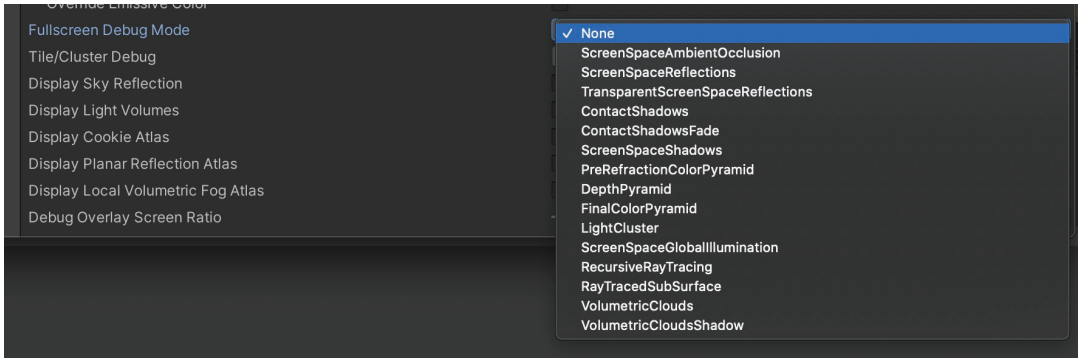
レンダリングデバッガー

Rendering Debugger は、ゲームビューの再生モード、または開発ビルドでビルドされたプレイヤーの実行時にも使用できます。メニューを開くには **Ctrl+Backspace** を使うか、ゲームコントローラーの 2 本のスティックを押してください。



ゲームビューまたはプレイヤーの Rendering Debugger ウィンドウ

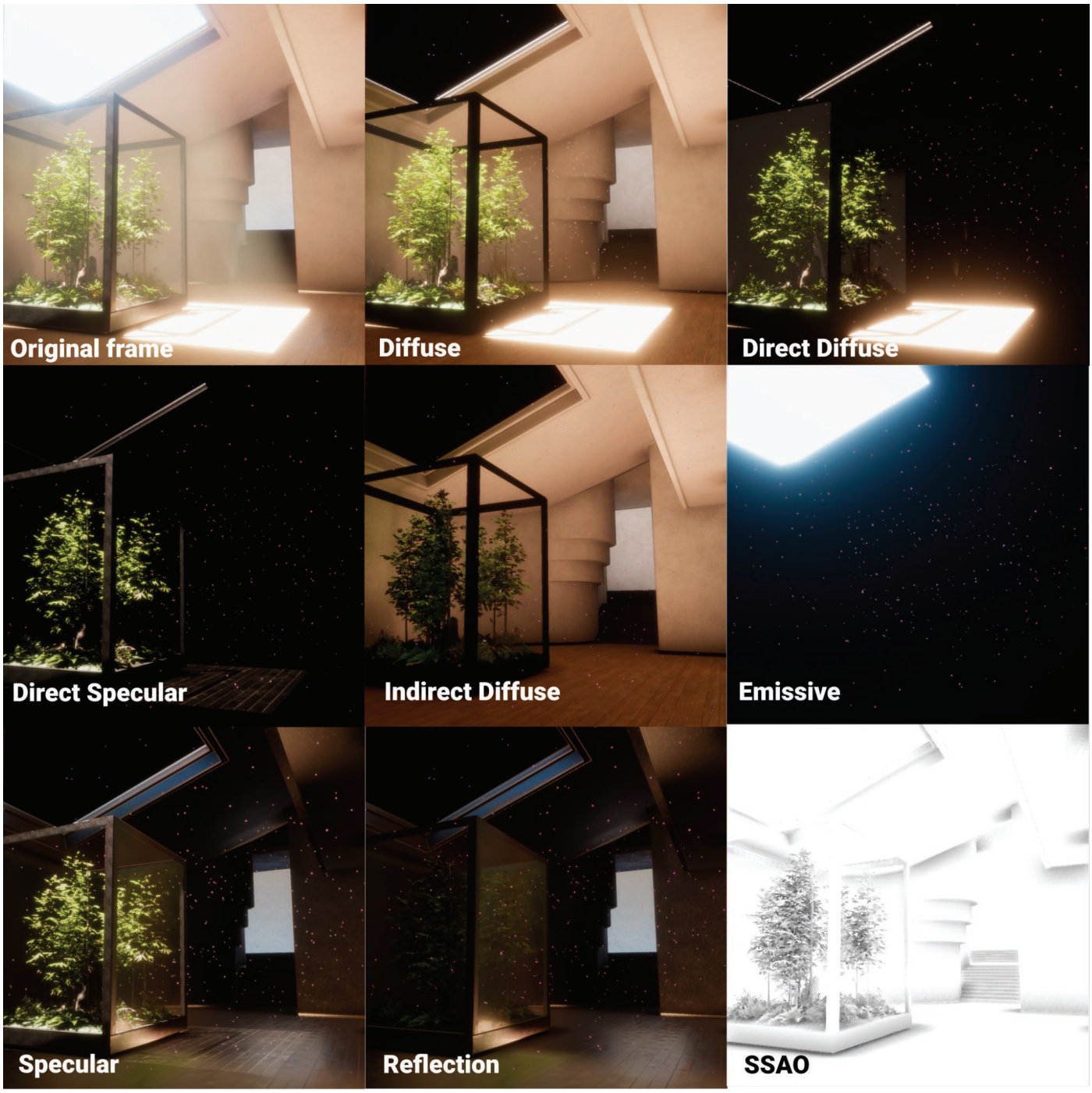
このデバッガーは、特定のレンダーパスのトラブルシューティングに役立ちます。Lighting パネルで **Fullscreen Debug Mode** と入力し、デバッグ機能を選択できます。



Fullscreen Debug Mode オプション

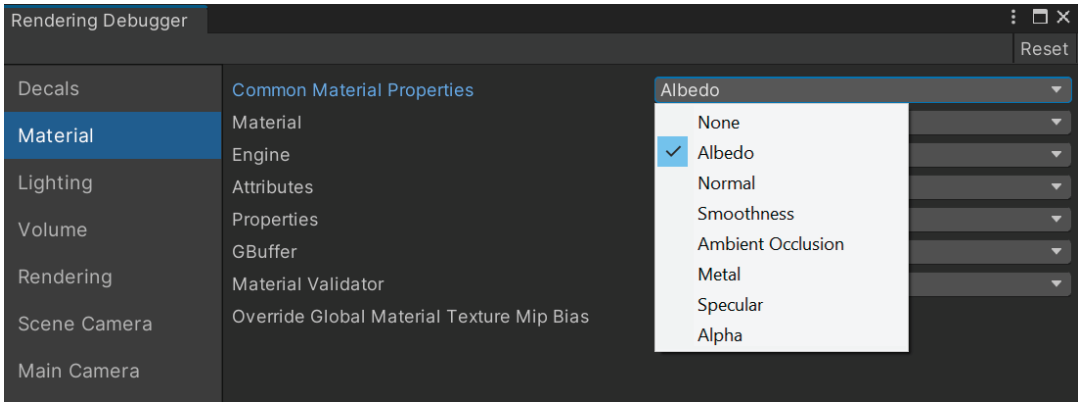
デバッグモードでは、さまざまな情報を手掛かりにして、ライティングやシェーディングの問題の原因を具体的に突き止めることができます。左側のパネルには、カメラ、マテリアル、ボリュームなどの重要な統計情報が表示されており、レンダリングの最適化に役立てることができます。

フルスクリーンのデバッグモードが有効な場合、シーンビューとゲームビューが切り替わり、特定の機能が一時的に可視化されます。この機能を利用して、効率的に診断を進めることができます。

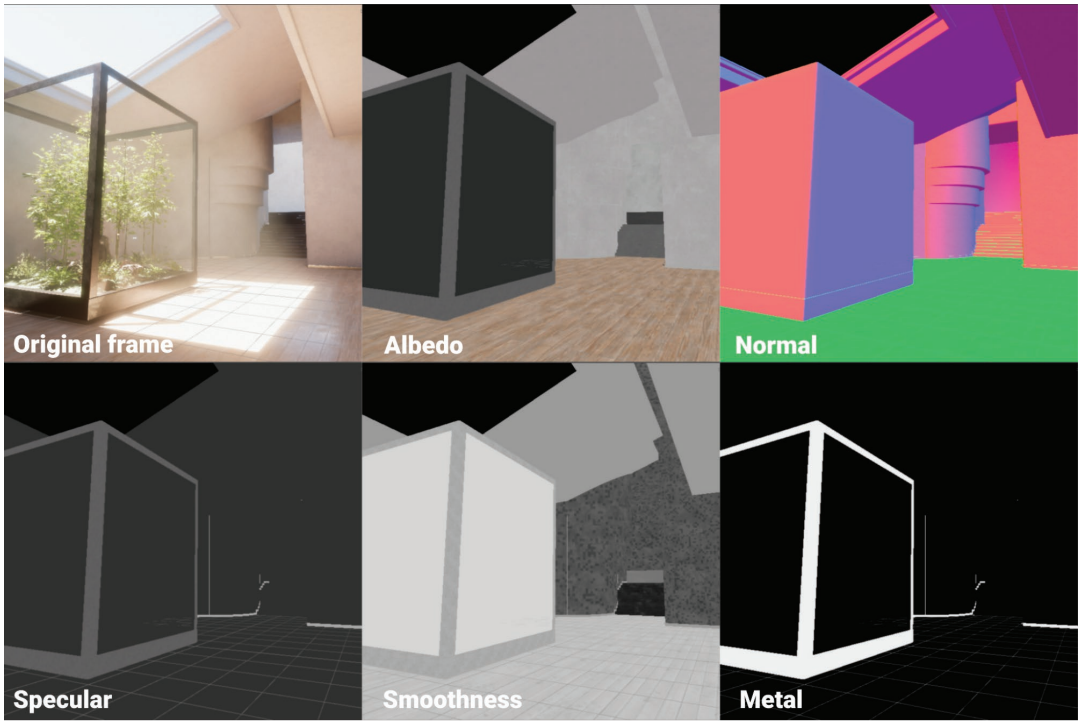


Lighting Debug Mode と Fullscreen Debug Mode は、シーン内の光源を把握するのに有効です。

一般的なマテリアルプロパティーのデバッグを行うこともできます。Material スクリーンで、Common Material Properties の中から、Albedo、Normal、Smoothness、Specular などを選択します。



マテリアルの基本プロパティ



レンダーパイプラインデバッガーを使用すると、マテリアルのトラブルシューティングを行えます。

カラーモニター

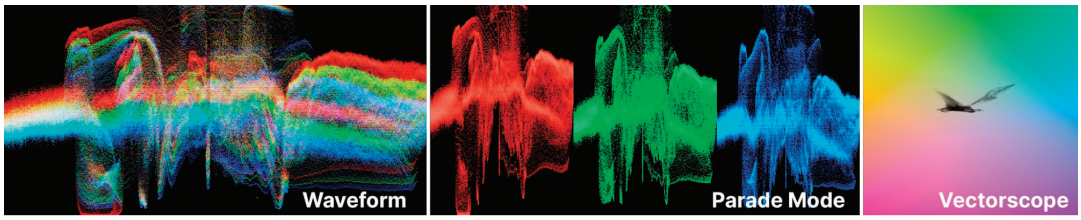
Unity には、シーンの全体的な外観と露出を制御するために使用できる業界標準の一連のカラーモニターが含まれています。これらのモニターは **Rendering Debugger** ウィンドウ (**Windows > Analysis > Rendering Debugger**) の Rendering タブでアクセス可能です。

Waveform モニターは、レンダリングされた画像のルミナンス (輝度) 値をグラフィカルに表示します。

Parade Mode は、波形画像を赤、緑、青の各チャンネルに分割し、色のアンバランスや不一致の特定と修正を簡単に行えるようにします。

Vectorscope モニターには、画像の色相 (色) と彩度 (色の強度) を測定する円形のグラフがあります。このツールは、肌の色調が正確であることを確認し、支配的なカラーキャストを識別するにあたって特に便利です。

これらのモニターをワークフローに組み込むことは、色の正確さとバランスの確保に役立ちます。



カラーモニター。

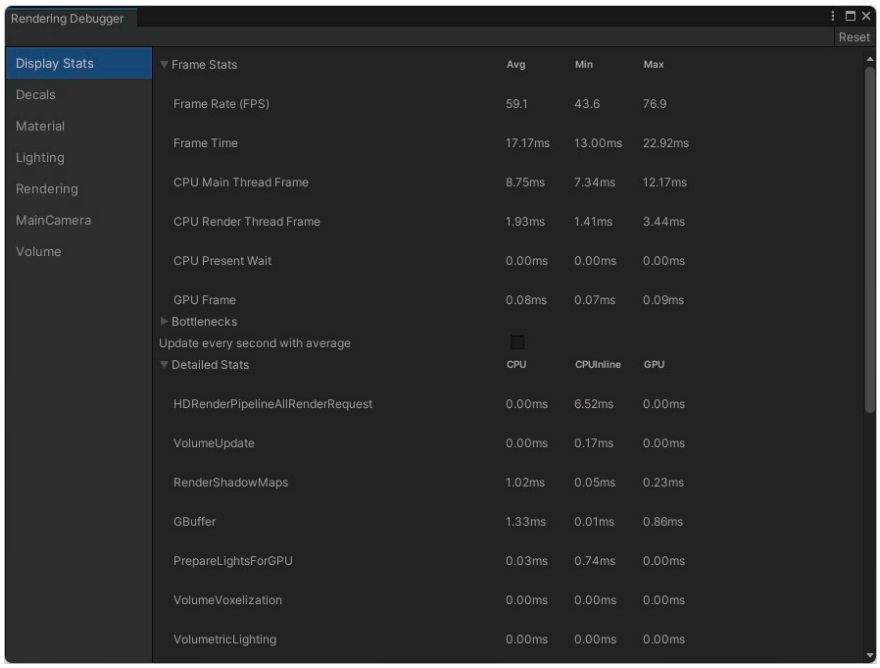
Runtime Frame Stats

SRP は Profiler の Runtime Frame Stats パネルにアクセスできます。このツールはエディターとプレイヤーの両方で利用でき、何が GPU の使用時間の大半を占めているのか簡単に把握することができます。

アーティスト、テクニカルアーティスト、開発者にとっては、HDRP　のフレームごとのパフォーマンスについてより詳しく知るのに役立つツールです。

注意:正確さを確保するため、エディターではなくプレイヤービルドでプロファイリングを行うことをすすめます。

Player Settings で **Frame Timing Stats** を有効化します。その後、再生モードで **Rendering Debugger** (エディターでは **Window > Analysis > Rendering Debugger**、プレイヤーでは **Ctrl + Backspace**) から Display Stats 画面にアクセスできるようになります。



Runtime Frame Stats (再生モードの場合)

詳細については、Render Pipeline Debugger (レンダーパイプラインデバッガー) に関するドキュメントを参照してください。

HDR ディスプレイのサポート

標準的なダイナミックレンジのディスプレイとは異なり、ハイダイナミックレンジ (HDR) ディスプレイは、人間の目が自然環境で知覚するものに近い、より広い範囲の輝度レベルを再現することができます。Unity 6 では、エディターとスタンドアロンプレイヤーの両方で クロスプラットフォーム HDR 出力 がサポートされるようになりました。

ハイダイナミックレンジ (HDR) ディスプレイが増えている中、Unity はこれらのディスプレイの利点を活かし、より優れたコントラスト (ハイライト/シャドウ) と彩度で画像を再現できます。

HDRP は以前からハイダイナミックレンジ画像をレンダリングすることができたものの、以前は HDR ディスプレイ向けの特別な出力を行うための専用のトーンマッパーがありませんでした。出力が最適化されていないと、画像の最も明るい部分と最も暗い部分のディテールが失われる可能性があります。

Unity 6 では、HDR ディスプレイのサポートがすべてのレンダーパイプラインと対応プラットフォームにおいて拡張され、ネイティブの色空間レンダリングと HDR トーンマッピングが可能になりました。HDR 出力では、10 ビットまたは 16 ビットの色深度、最大 680 億色の色範囲を使用できます。

エディターとスタンドアロンプレイヤーでは、すべての SRP レンダーパイプラインおよび対応しているプラットフォームで、HDR トーンマッピングとディスプレイが完全にサポートされるようになりました。HDR レンダリングでは、より幅広い色値を格納するために浮動小数点フレームバッファが使用されます。これにより、標準レンジのレンダリングと比較して、ライティング、ポストプロセス、全体的な視覚的精度が向上します。

プロジェクトに HDR を実装すれば、新しい HDR キャリブレーションテンプレートを使用して HDR キャリブレーションメニューを組み込むこともできます。これにより、露出不足や露出オーバーなどの一般的な問題を回避し、画面上のテキストを見やすく保つことができます。

HDR 出力を有効にするには、**Project Settings > Player > Other Settings** に移動し、次の設定を有効にします。

— **Allow HDR Display Output**

— **Use HDR Display Output**

Use HDR Display Output は、メインディスプレイで HDR 出力を使用する必要がある場合にのみ有効にします。

HDR 出力はゲームビューとプレイヤーでのみアクティブになります。現在、この機能は DirectX 12 でのみ利用可能です。

HDR では、このオプションをオンにすることで、HDR ディスプレイ上で URP によるより高品質な画像レンダー出力を利用できます。その結果、これらのデバイスでは、自然なライティング条件をより効果的に再現した色やコントラストのある画像をより忠実に表示できます。

Unity 2022 で利用可能になった既存のデスクトップおよびコンソール向けサポートに加え、Unity 6 では以下のプラットフォームでのモバイル向けサポートが導入されました。

— iOS プレイヤー (iOS 16+、iPadOS 16+)

— Vulkan および GLES を使用している Android プレイヤー (Android 9+、デバイスの能力による)

HDR ディスプレイをサポートする一般的なモバイル機器には、iPhone X 以降、Samsung の Galaxy S10 以降、Galaxy Note 10 以降、Galaxy Tab S6 以降などがあります。

詳細については、このブログ記事 や High Dynamic Range (HDR) Output (ハイダイナミックレンジ (HDR) 出力) を参照してください。

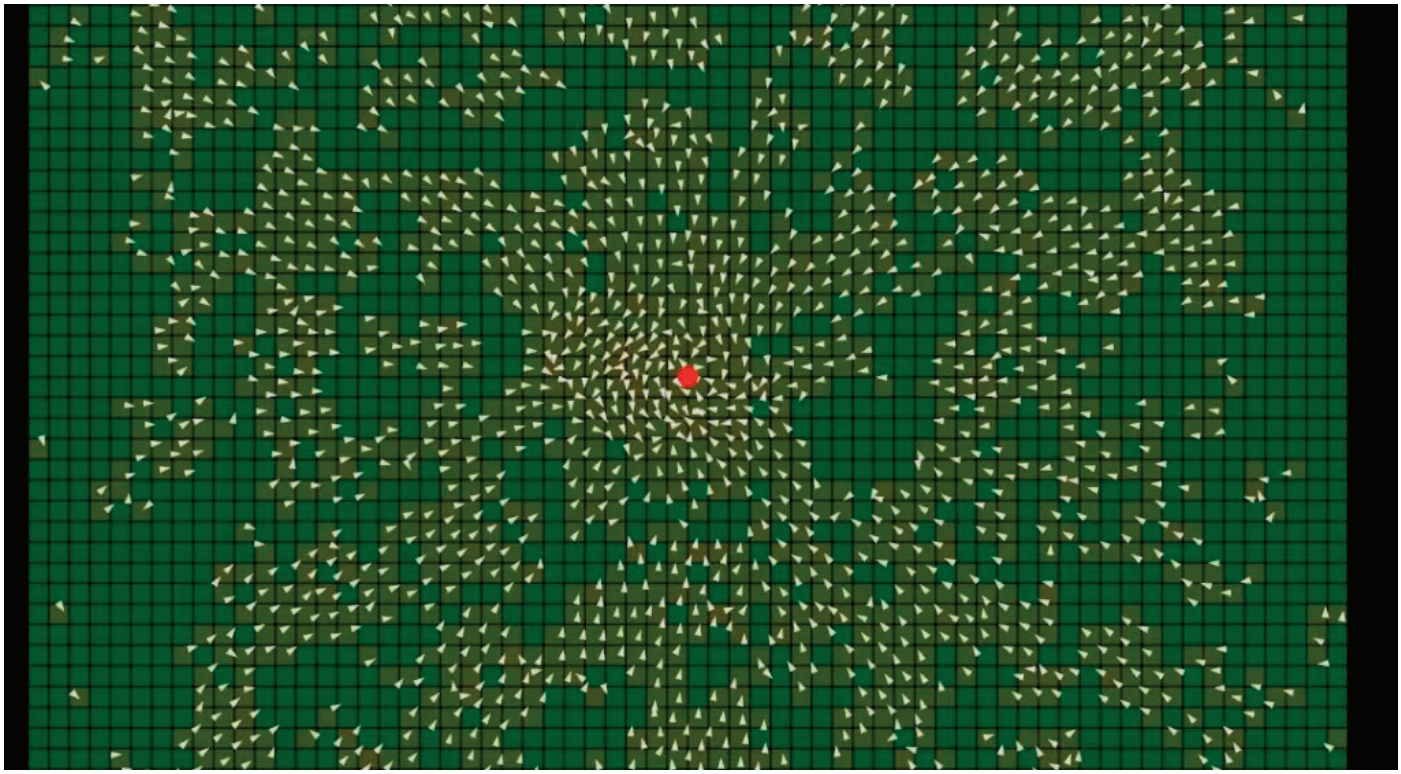
VFX Graph のサポート

Unity 6 には HDRP で動作する VFX Graph 機能も搭載されており、柔軟性と高度な視覚効果 (VFX) が強化されています。VFX Graph は、アーティストが HDRP のレンダリング機能を活用したリアルタイムの視覚効果を作成できるようにすることで、HDRP を補完します。

カスタム HLSL ブロックとオペレーター

Unity 6 には、カスタム HLSL コードを VFX Graph に直接記述できる新しい ブロック と オペレーター が含まれています。これを使用して、1 つのノードで複雑な動作をスクリプト化できます。ノードにカスタム HLSL コードを埋め込むことも、外部ファイルからロードしてコードを再利用することもできます。

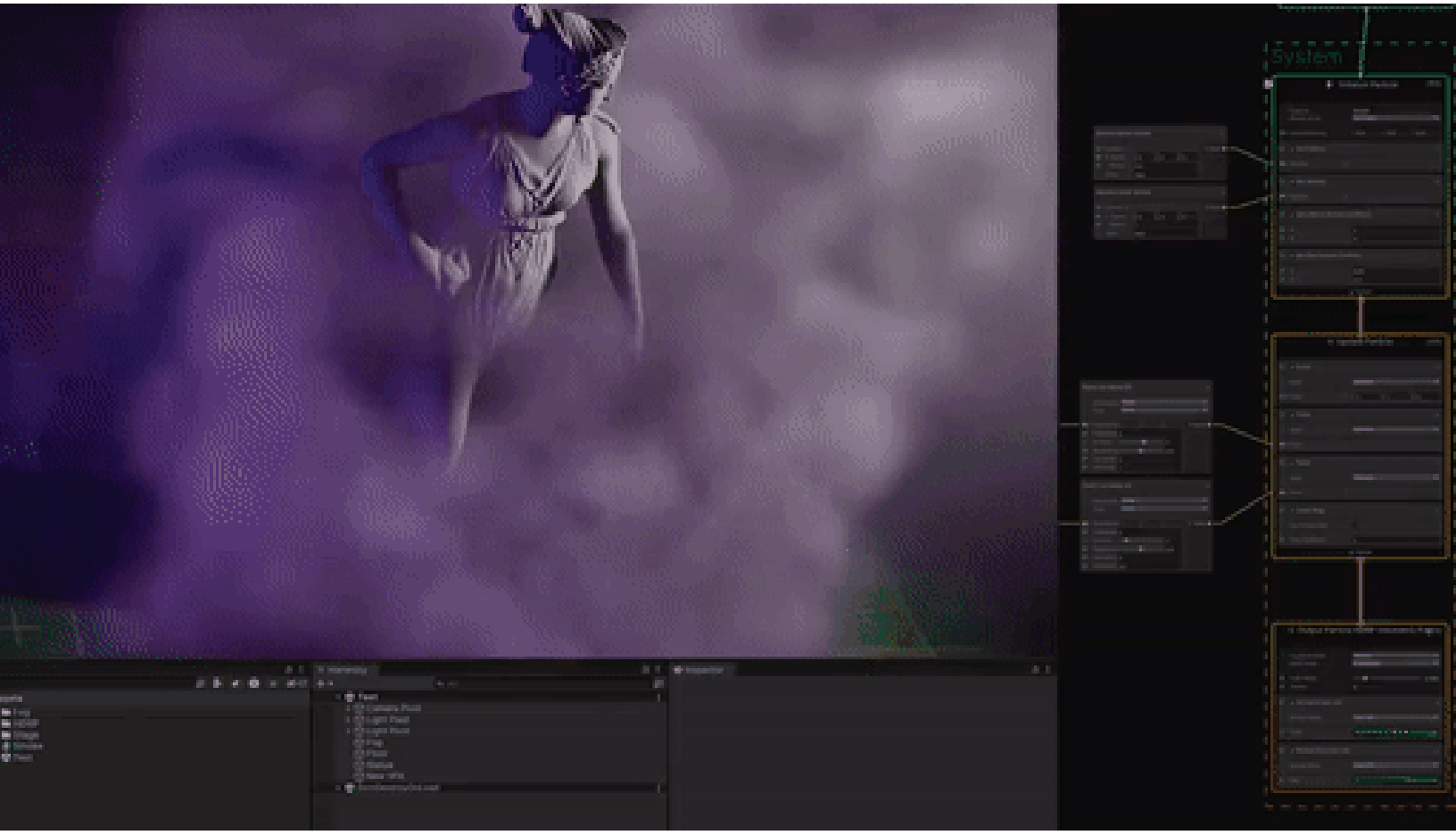
これを使用してカスタムシェーダーを記述したり、Unity のビルトイン機能を拡張して独自のエフェクトを作成したりできます。例えば、近傍探索アルゴリズムを使用してカスタムのフロッキング動作を実装したり、バッファリードバックを使用して視覚効果をオーディオトリガーと同期させたりすることができます。



HLSL ブロックで作成されたエフェクト。

Volumetric Fog の出力

Output Particle HDRP Volumetric Fog コンテキストは、HDRP で ローカルボリュメトリックフォグ をサンプリングし、VFX Graph がフォグをボリュメトリックライトに直接組み込めるようにします。その結果、VFX Graph パーティクルとシーンライティングのインタラクションが改善され、現実感が高まります。これを使用して、動的なフォグエフェクト、リアルな雲、煙、炎を作成できます。



Output Particle HDRP Volumetric Fog 出力ノード

レイトレーシングのサポート

Unity 6 の VFX Graph パーティクルが、HDRP で以下のレイトレーシング機能をサポートするようになりました。

- レイトレーシングによるリフレクション
- 屈折、シャドウ
- アンビエントオクルージョン
- グローバルイルミネーション

クアッド (四角形)、三角形、八角形を使用する VFX Graph でレイトレーシングを有効にできます。HDRP シーンでレイトレーシングを使用するには、Getting started with ray tracing (レイトレーシングの使用を開始する) を参照してください。注意:VFX Graph は、メッシュやストリップを使用したレイトレーシングをサポートしていません。

VFX Graph を HDRP プロジェクトに統合する方法については、e-book *The Definitive Guide to Creating Advanced Visual Effects in Unity (Unity 6 Edition)* (Unity で高度なビジュアルエフェクトを作成するための決定版ガイド、Unity 6 版) を参照してください。

最適化

複雑なゲーム環境では高いパフォーマンスが求められるため、レンダーパイプラインの最適化はアプリケーションにとって不可欠です。モジュール式要素やプレハブを使用するゲームでは、さまざまな手法を用いることで、CPU 負荷を軽減し、レンダリング効率を向上させることができます。

スクリプタブルレンダーパイプラインバッチ

SRP Batcher は、ドローコールを準備してディスパッチするために必要な CPU オーバーヘッドを軽減することで、ドローコールを最適化します。SRP Batcher は、ドローコールの *数* を減らす代わりに、ドローコール間の *レンダーステートの変更* を最小限に抑えます。

SRP Batcher グループは、GPU コマンドを **SRP バッチ** にバインドして描画し、同じシェーダーバリアントを共有するマテリアルのパフォーマンスを向上させます。

最適なパフォーマンスを得るには、各 SRP バッチにできるだけ多くの バインド コマンドと 描画 コマンドを入れることが望ましいです。使用するシェーダーバリアントを少なくすることで、バッチ処理の効率を最大化しつつ、複数のマテリアルが同じシェーダーを使用できるようにします。

SRP Batcher の機能の詳細については、こちらの ドキュメントページ を参照してください。

BatchRendererGroups

BatchRendererGroup (BRG) は、スクリプタブルレンダーパイプライン（SRP）と SRP Batcher を使用するプロジェクトにおいて、多数のオブジェクトを効率的に処理するために用意された Unity のレンダリング API です。

従来のゲームオブジェクトベースのレンダリングに依存するのではなく、BRG はレンダリングデータを直接制御できます。そのため、従来のゲームオブジェクトではコストがかかりすぎる DOTS Entities、プロシージャル環境、カスタム地形に最適です。

BRG は、複数のオブジェクトを 1 回のドローコールにまとめることでパフォーマンスを最適化し、CPU オーバーヘッドを軽減します。永続的な GPU バッファを保持し、頻繁な CPU 介入を必要とせずにオブジェクトデータを保存および更新します。

BRG を使用するには、プロジェクトが **SRP Batcher** に対応している必要があります。また、BRG シェーダーバリアントを保持し (**Project Settings > Graphics**)、**Player Settings** で Allow "unsafe" Code を有効にする必要があります。

BRG は、密集した植生、画面上の多数のオブジェクトのレンダリング、動的な LOD 処理など、大規模なインスタンシングを必要とするシーンで効果を発揮します。これは、新しい GPU Resident Drawer と統合されています。

GPU Resident Drawer

Unity 6 の GPU Resident Drawer は、ゲームオブジェクトのレンダリング効率を向上させます。この新機能は、BatchRendererGroup API を使用します。これは、より優れたインスタンシングを処理する特殊な高速パスを介して MeshRenderer をバッチ処理およびレンダリングできます。GPU Resident Drawer は、オブジェクトごとの処理コストを下げることにより、ドローコールの多いシーンでの CPU ボトルネックを軽減できます。

この機能はスクリプタブルレンダーパイプライン (URP と HRDP の両方) に対応しており、コンピュートシェーダーをサポートするグラフィックス API を必要とします (OpenGL/GLES では利用できません)。

GPU Resident Drawer を使用したパフォーマンスの向上は、シーンの複雑さに応じて増加します。シーンのディテールやインスタンス化されたオブジェクトが多いほど、メリットは大きくなります。

GPU Resident Drawer を有効にするには、以下を行います。

— **Project Settings > Graphics** に移動し、**BatchRendererGroup Variants** を **Keep All** に設定します。

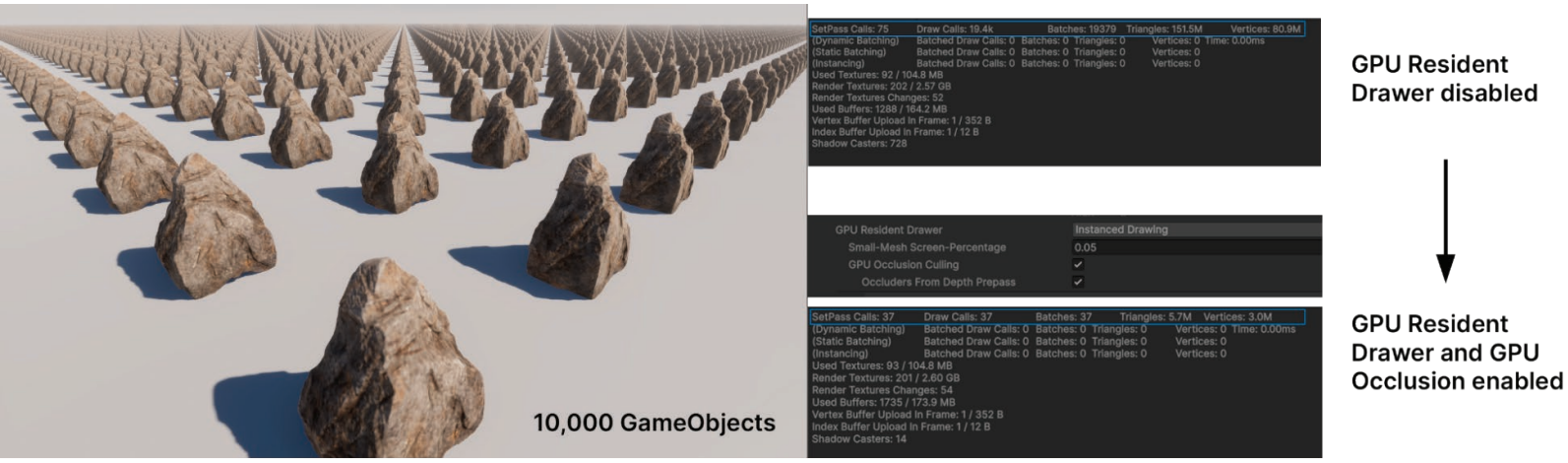
— 次に、**HDRP Render Pipeline Asset** を探します。**GPU Resident Drawer モード** を **Instanced Drawing** に設定します。

— ゲームオブジェクトのマテリアルで **Enable GPU Instancing** がオンになっていることを確認します。

GPU Resident Drawer を最適化するには、プロジェクトで **Static Batching** (静的バッチ処理) を無効化にする (**Edit > Project Settings > Player**) か、Inspector でインスタンスごとに無効にします。静的バッチ処理が有効になっていると、GPU Resident Drawer は機能しません。

GPU Resident Drawer を使用する場合、追加のシェーダーバリアントをコンパイルするため、ビルド時間が長くなる可能性があることに注意してください。この機能は標準のメッシュレンダラーで機能しますが、スキンメッシュレンダラー、VFX Graph、パーティクルシステムなどのエフェクトレンダラーに対応していません。

GPU Resident Drawer を有効にすると、複数のゲームオブジェクトが同じメッシュを共有するシーンのドローコールを節約できます。ここに示すのは、インスタンスが 10,000 回作成されたプレハブのプロファイラーでの比較です。

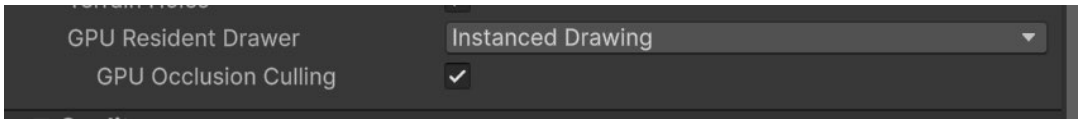


GPU Resident Drawer を使用した場合と使用しなかった場合の統計を比較します。

GPU オクルージョンカリング

シーンで GPU Resident Drawer を使用している場合は、**GPU オクルージョンカリング** を活用して、実際には画面に表示されない GPU インスタンスを削除することもできます。これにより、設定 1 つでオーバードローの多いシーンでの GPU パフォーマンスを向上させることができます。

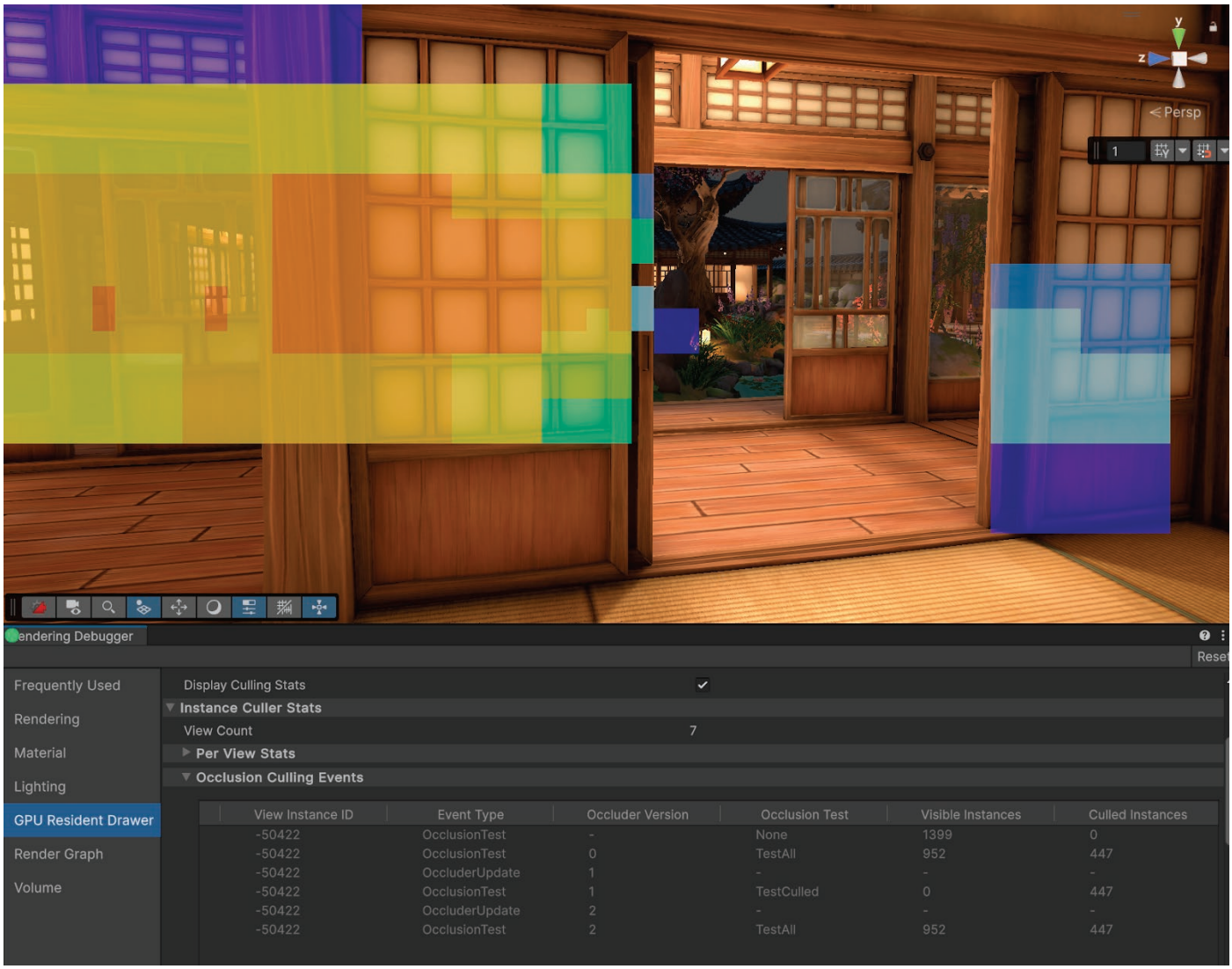
これを有効にするには、必ず GPU Resident Drawer モードを有効にしてから、HDRP アセットを探します。**Rendering** セクションに移動し、**GPU オクルージョンカリングを有効にします。**



GPU Occlusion Culling (GPU オクルージョンカリング) を有効にします。

シーンで GPU オクルージョンカリングが有効かどうかを確認するには、**Window > Analysis > Rendering Debugger** に進み、**GPU Resident Drawer > Occlusion Test Overlay** を選択します。

RenderDebugger は、シーンのどの部分が最も恩恵を受けるかを示すヒートマップで画面に色を付けます。**青** で示されている領域はカリングされたインスタンスが少ないことを示し、**赤** はカリングされたインスタンスが多数あることを示します。その間は色のグラデーションが使用されています。



GPU Resident Drawing は URP でも動作します。

詳細については、別のガイドブック *Introduction to URP for advanced creators* (上級 Unity クリエイター向けユニバーサルレンダーパイプライン (URP) 入門 - Unity 6 版) を参照してください。

DX12 Graphics Jobs のエディターでのサポート

DX12 を使用する場合、Unity エディターで Graphics Jobs がサポートされるようになりました。これにより、シーンビューと再生モードのレンダリングが改善され、複雑な環境をエディターで作業する際のパフォーマンスが向上しています。

この動作を有効にするには、**DirectX 12** を **Graphics API** として設定します。次に、**Player** 設定で **Graphics Jobs** にチェックを入れ、**Jobs** 設定 (**Preferences/Settings > Jobs**) で **Allow Graphics Jobs in Editor** を有効にします。

DirectX 12 (DX12) のような最新のグラフィックス API は、コマンドバッファの記録と送信をマルチスレッドで行うことを可能にし、CPU 使用率を向上させます。Unity の Graphics Jobs システムは、ドローコールを処理する際の CPU ボトルネックを減らすことで、このプロセスを最適化します。

メインスレッドが描画コマンドをキューに入れ、レンダリングスレッドがワーカースレッドを生成して変換し、GPU コマンドバッファとして送信します。このアップデートにより、高パフォーマンスのレンダリングや大規模なシーンにおいて、CPU と GPU の負荷分散が改善されています。

DX12 Graphics Jobs は、DirectX 12 を使用する Windows プラットフォームのエディターでサポートされています。ランタイムビルドでは、Graphics Jobs は Windows (DX12) およびコンソールプラットフォームでサポートされています。

可変レートシェーディング (Unity 6.1)

Variable Rate Shading (VRS; 可変レートシェーディング) を使用すると、画面のさまざまな部分にわたってシェーディングの細かさを調整できます。ピクセルレベルまたはオブジェクトレベルでシェーディング品質を制御することで、見栄えに大きな影響を与えることなく GPU パフォーマンスを向上させることができます。

例えば、レーシングゲームでは、プレイヤーの車や周囲の細部は高品質のシェーディングを使用でき、モーションブラーのある道路などの目立たない部分にはリソースを節約するために低品質のシェーディングを適用します。

Unity 6.1 では、可変レートシェーディング 用の新しい API が導入されており、Windows (DirectX 12)、Vulkan、および対応するコンソールをサポートしています。

Shading Rate サンプルプロジェクト で、可変レートシェーディングの仕組みを示す実践的なデモを確認してください。詳細については、こちらのディスカッションページ を参照してください。

可変レートシェーディングは、画面のさまざまな部分でシェーディングの品質を調整します。

次のステップ

このガイドが、皆様が次のプロジェクトで HDRP に取り組むきっかけとなれば幸いです。さらに詳しく知りたい方は、Unity Hub からいつでも入手できる **3D Sample Project** を利用してください。このガイドに関するご質問やフィードバックは、フォーラムのスレッド でお知らせください。

以下の追加リソースも必ず確認してください。また、Unity ブログ や HDRP コミュニティフォーラム では、いつでもヒントを得ることができます。

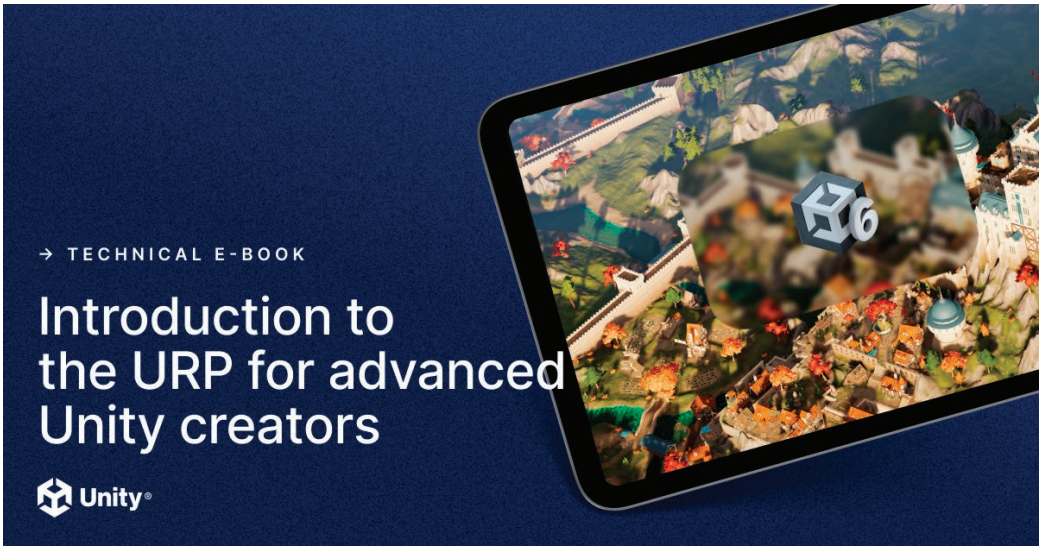
Unity は、リアルタイムコンテンツを作成するための最高のツールを用意して、アーティストや開発者の皆様を支援したいと考えています。ゲームの世界や環境を作ることは芸術であり、科学です。芸術と科学を融合させることで、魔法のような効果が生まれます。

その他のリソース

- HDRP の機能を紹介する動画については、Achieving high-fidelity graphics with HDRP (HDRP で高品質なグラフィックスを作成する方法) を視聴することをすすめます。
- ビルトインレンダーパイプラインから移行する場合は、こちらのチャート を参照してください。2 つのレンダーパイプラインの機能の違いが詳細にまとめられています。
- HDRP に関するドキュメントには、このパイプラインのあらゆる機能の詳細が含まれています。
- パフォーマンス強化のための HDRP 設定について、詳しくはこちらのブログ記事をご覧ください。

— 新しい水システムについての詳細は、この ブログ記事 または水システムの ドキュメント を参照してください。また、この 紹介動画(自動翻訳機能推奨) で機能の概要を確認してください。

— Unity の開発 と テクニカルアートについてもっと知りたいですか?Unity ベストプラクティスハブでは、短時間でより多くの成果を上げるための実用的なヒントやベストプラクティスが満載の記事や電子書籍が用意されています。



プログラマー、アーティスト、テクニカルアーティスト、デザイナーのための、技術面についての e -book でさらに学びましょう。


Unity®