テクニカルアーティスト向け UNITY

機軸となるツールセットとワークフロー

2021 LTS EDITION

Unity®

目次

はじめに

HD レンダーパイプラインで作成された都市環境

本ガイドでは、Unity のツールセットとシステムについて概説しています。テクニカルアーティスト（TA）の方は本ガイドを参照することで、自身のチームのゲーム制作におけるビジュアル要件を満たすために Unity を活用できるようになるでしょう。

テクニカルアーティストは、技術的な要件と芸術的な要件の両方について解決策を提供できるため、チーム内でアーティストとプログラマーの橋渡し役になることが多くなっています。

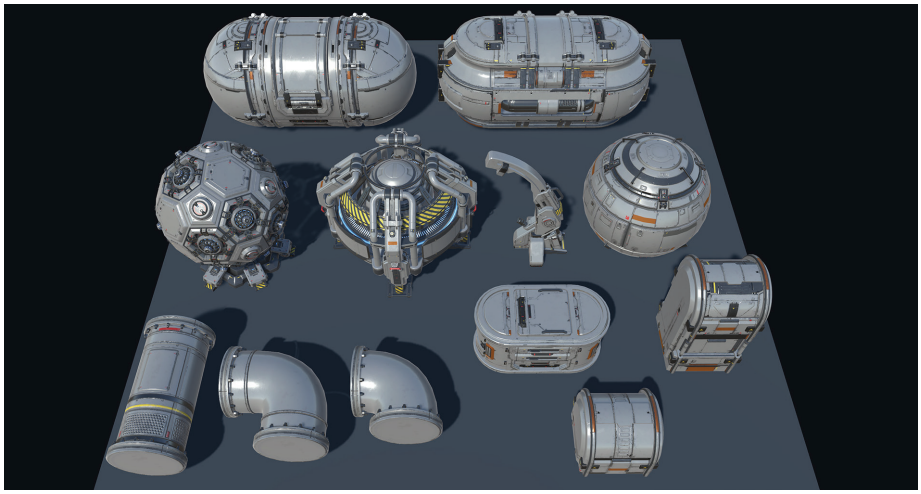
自分たちが使用しているデジタルコンテンツ制作（DCC）ツールやゲームエンジンによって、さまざまなターゲットプラットフォームで何が実現できるのか、テクニカルアーティストは幅広く理解しています。このため、ターゲットハードウェアに関する制限事項や機会があれば、アートディレクターやアーティストに知らせることができます。

多くの TA は、チーム内で大変に複雑なアート面でのニーズに真っ向から取り組んでおり、キャラクターリギングやシェーダーの記述のほか、プロセスを高速化する新しいワークフローや制作ツールの提案までも行っています。TA がゲームエンジン技術を利用する業務へ密に携わり、レンダリングの完成品としての成果をリアルタイムに確認することで、チームが目的として据えた基準を満たすゲーム、その他のアプリケーションのビジュアル品質に対して、確信が持てるものへと導く重要な役割を果たします。

Unity の TA は大規模なゲーム制作における長年の経験から、ライティング、アニメーション、シネマティクス、ビジュアルエフェクト、シェーダー、グラフィックスプログラミングについて高度な知識を備えています。Unity でも特に経験豊富な、R&D チームの TA の数名に加え、ストックホルムに拠点を置くデモチーム、『*Adam*』、『*Book of the Dead*』、『*The Heretic*』、『*Enemies*』のクリエイターたちから得た知恵をまとめたのが本ガイドです。

Unity 2021 LTS 以降のバージョンで利用可能なオーサリングツールと API についての入門書として本ガイドを参照し、効率的にコンテンツ制作パイプラインを設定する方法を学んで、チームのメンバー全員が最高の成果を上げられるように後押ししていきましょう。

プレハブ



環境の構成要素となっているプレハブの例

Unity のプレハブシステムを使用すると、**ゲームオブジェクト**を作成して設定、保存できます。このゲームオブジェクトは、コンポーネント、プロパティ値、子ゲームオブジェクトがすべて揃ったアセットとして再利用可能です。**プレハブアセット**は、シーンビューに新しいプレハブインスタンスを作成するためのテンプレートとして利用できます。保存したプレハブは、シーン間で共有できるだけでなく、再設定の必要なく他プロジェクトと共有さえも可能です。

ゲームのプラットフォームや収集可能なアイテムなど、何度も使用されるオブジェクトにおいてプレハブは有用です。プレハブを作成するには、「**Hierarchy**」ウィンドウから「**Project**」ウィンドウにオブジェクトをドラッグします。

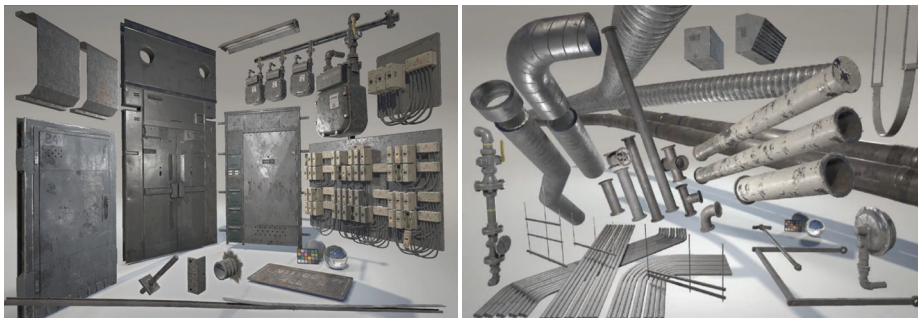
「Project」ウィンドウで 2 列のビュー（下）と 1 列のビュー（左）に表示されたプレハブ MOD_Trees2

プレハブは Unity 内のすべてのアセットと同様に編集が可能です。編集はオブジェクト単位で行うことができ、プレハブの単一のインスタンスを個別の目的に合わせて変更できます。また、プレハブのすべてのインスタンスに変更を適用することもできます。これにより、オブジェクトのエラーの修正、アートの入れ替え、その他のスタイル上の変更を行う際の効率性が向上します。

ネスト状のプレハブでは、より大きなプレハブを作成するために、あるプレハブを別のプレハブの親にすることができます。例えば、部屋や家具などの小さいプレハブで構成される、建物を作成するような場合です。これは、アセットの開発をチームの複数のアーティストや開発者で分割して行うのに便利で、コンテンツのさまざまな部分を全員同時に作業できることになります。

プレハブバリアントでは、オブジェクト指向プログラミングの継承とほぼ同じくして、あるプレハブから別のプレハブへ派生させることが可能です。バリアントに変更を加えるには、元のプレハブに影響を及ぼすことなく特定の部分をオーバーライドする必要があります。いつでも、すべての変更を削除してベースプレハブへ戻すことができます。

また、一度にすべてのプレハブバリアントを変更するために、ベースプレハブ自体に変更を直接適用することもできます。



大きなネスト状のプレハブを小さいプレハブで構成できます。例えば、左のプレハブパネルには多数の電源トランスプレハブ、右のプレハブパネルは異なるマテリアルのパイププレハブのバリアントが示されています。

その他のリソース

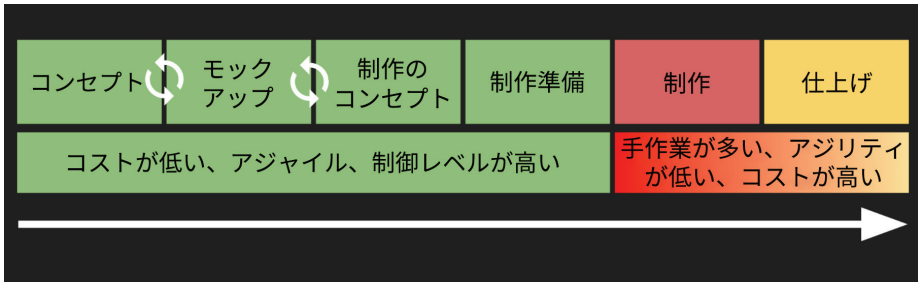
ネスト状のプレハブの紹介
プレハブを編集するワークフローの改善

アセットの操作

制作パイプライン

いかなるゲームエンジンおいても、基礎となるシステムは安定した**アセットパイプライン**です。これはプロジェクト内のアセットの状態を決定し、それに応じて依存関係にあるシステムの更新を随時行うものです。アセットパイプラインは、ユーザーが定義した制作パイプラインによって拡張することもできます。

プロジェクト用のアセットパイプラインが確立されていると、アーティストが技術要件を満たしていないアセット（割り当てられていた数または予定されていた数よりも三角形数が多い 3D モデルなど）をプロジェクトに追加してしまうリスクを抑えることができます。優れたアセットパイプラインというものは、チームが命名規則、ファイルパス、プレハブ、その他設定についての基準へ沿うように手を差し伸べるものです。そうすれば、アセットの重複、壊れた依存関係などの問題や、アセットをバンドルする際の諸問題を避けることができます。このセクションでは、効率的で安全なアセットパイプラインを作成するために Unity で使用できるツールを紹介します。



詳細なプリプロダクションプランがあれば、プロジェクト内で複数のアーティストが活動している場合に作業を並行進行できるため、未使用アセットや、生産性、パフォーマンスに影響ある諸問題を最小限に抑えられます。

非破壊的なアセットパイプライン

プリプロダクションプランニングの一環として、作成が必須のアセットのリスト、プロジェクトの技術要件に基づくアセットのバジェット、開発コストなど、アセットワークフローにおける様々な側面についてチームで意見をまとめる必要があります。

レベルにおける必須要素をグループ分けする手段の 1 つとして、以下のようなものがあります。

— **大きな要素：**固有のテクスチャを使用して作成することはできず、タイリングやモジュラー要素を使用しなければならない要素。壁、地面、崖、建物の屋根など。

— **中サイズの小道具：**固有のテクスチャシートを備えた小道具。樽、木箱、岩、ドアなど。

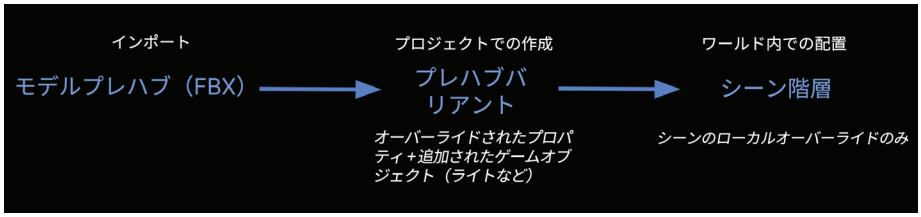
— **使用頻度の高い、小さな要素：**オブジェクトを地面に置いたりスケールを伝えたりするために使用するアイテム。小石、葉、ねじ、ボルトなど。

各アセットの時間、ポリゴン数、テクスチャ解像度の割り当てを決定し、それを踏まえてタスクを優先順位付けします。次に、必須ではないアセットを作成して、必須アセットの数を必要最小限まで減らします。

さらに、アセットインポートのワークフローには、3D モデルからのプレハブ作成という重要な側面があります。制作中に FBX モデルをプレハブに直接変換することは推奨されません。FBX ファイルに変更（ジオメトリでの階層の変更など）が加えられると、プレハブが壊れるからです。こうなるとシーン内のプレハブのインスタンスをすべて、更新されたバージョンへ置き換えなければならなくなります。この問題の解決策については、プレハブバリアントで記載されています。

プレハブバリアントを使用すると、シーンと依存関係のないアートアセット制作が可能となり、プロトタイピング中でもアートアセットを使用目的に適したものにできるようになります。プレハブバリアントは、ゲーム世界の構築時に共同で反復作業を行う際にお勧めです。元のモデルのバリエーションを作れるといった通常のプレハブのメリットがあるだけでなく、FBX モデルプレハブの「弱い」参照としても機能します。このため、プレハブを作り直すことなくモデル要素を便利に追加、分割、または削除できます。制作アーティストは FBX モデルに変更を加えることができ、プレハブバリアントはその変更に応じて更新されます。また、アーティストはプレハブバリアントを使用することで、プロジェクト自体は変更せずにコンテキスト内でアートの変更を確認できます。

それと同時に、シーンでコラボレーションを促進したい場合は、他でネスト状のプレハブを使用することも検討するようにしてください。

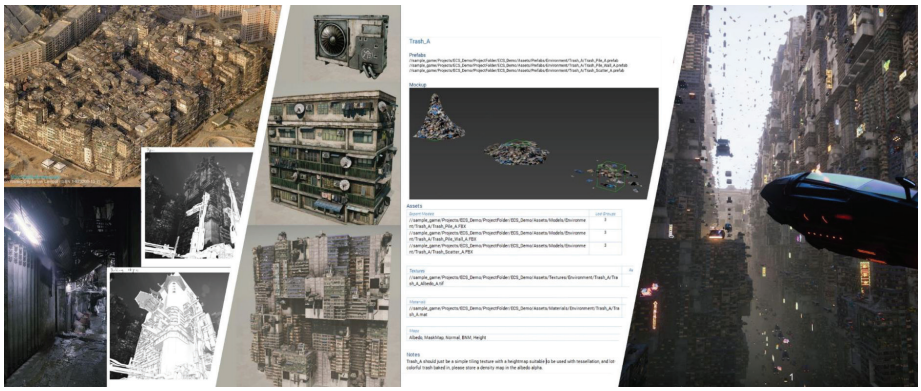


多数の共同作業者の制作において推奨のプレハブバリアントワークフロー

AssetPostProcessor によるアセット設定の自動化

数千ものアセットを扱わなければならない制作では、インスペクターから各アセットの仕様を手動で設定することは避けましょう。

アセットファイルの検証プロセスを自動化するには、**AssetPostProcessor API** を使用します。これは、インポートパイプラインにフックし、アセットのインポート前後にスクリプトを実行するのに便利です。必要な変更を加えるスクリプトにアセットを追加するだけで済みます。プロジェクト内のアセットに変更を自動的に再適用する単一のスクリプトを使用して、設定を変更できます。



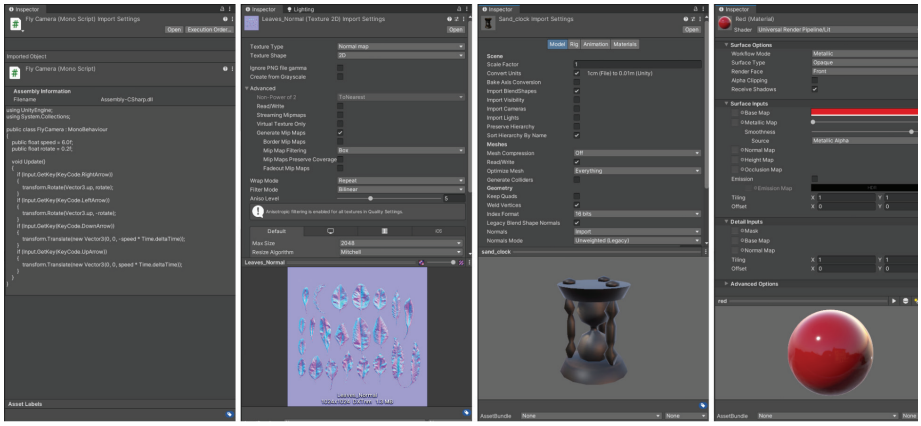
左から右へ：ムードボードの例、コンセプトアート、プレハブアセット、ディレクトリと命名基準、Unity 制作の『*Megacity*』デモの最終的な外観

アセットのインポート

DCC ツールで作成したアセットを、Unity が読み取り可能なプロジェクトのアセットフォルダーに直接取り入れることができます。新しい変更をファイルに保存すると、それを Unity が検出し、必要に応じてファイルを再インポートします。

スクリプトとインスペクター、どちらを使用するか否かにかかわらず、Unity でアセットの設定を変更した際には、元のソースファイルは変更されないままとなります。代わりに、ユーザーが選択したインポート設定に一致する、ゲームで使用可能なアセットの表現が Unity の内部で作成されます。Unity エディター内でのアセットの移動または名前の変更は、常に対応するメタファイルと同様に行うようにしてください。

インポート設定は、手元のアセットのタイプに応じて異なる表示になります。



インスペクターに表示される、さまざまなアセットタイプのプリビジュアライゼーションと設定。左から C# スクリプト、テクスチャ、FBX アセット、マテリアル

Unity では、最も一般的な 3D モデル、テクスチャ、オーディオファイルをはじめとして、多数のアセットタイプがビルトインインポーターとともにサポートされています。完全に網羅したリストについては、ドキュメントを参照してください。

Scripted Importer を使用して C# で独自のインポーターを作成することもできます。Scripted Importer を使用すると、Unity でネイティブにサポートされていないファイルを操作できます。**Alembic** インポーターや **PSD インポーター**など、パッケージマネージャーで例を確認できます。

アセットデータベース

Unity では、ほとんどのアセットタイプについて、アセットのソースファイルから得たデータをゲーム内で使用できる形式に変換する必要があります。これらの変換されたファイルと関連データは、アセットデータベースに保存されます。

変換プロセスが必要なのは、ほとんどのファイル形式がストレージスペースを節約できるように最適化されていることが理由です。しかしゲーム内においては、アセットデータはハードウェアに対応した、CPU、グラフィックス、オーディオハードウェアなどですぐに使用できる形式である必要があります。例えば、Unity で .PNG 画像ファイルをテクスチャとしてインポートするときは、実行時に元の .PNG 形式のデータは使用しません。そうではなく、テクスチャをインポートすると、その画像の新しい表現が Unity によって別の形式で作成され、**Library フォルダー >「Project」ウィンドウ**に保存されます。このインポートされたバージョンが Unity 内の **Texture** クラスで使用されます。Unity はこのバージョンをリアルタイムで表示するために GPU にアップロードします。

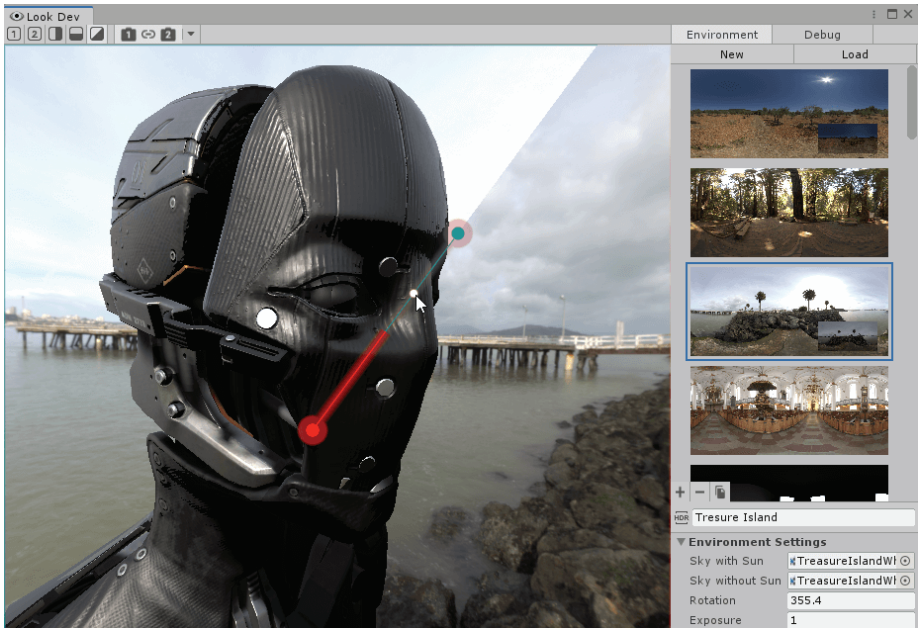
すでにインポート済みのアセットのソースファイルを後から変更する（またはその依存関係を一部でも変更する）と、Unity によってファイルが再インポートされ、インポートされたバージョンのデータが更新されます。このプロセスの詳細については、アセットデータベースの情報を更新するを参照してください。

アセットデータベースには、アセットへのアクセスのほか、インポートプロセスのコントロールやカスタマイズに使用できる **AssetDatabase API** も用意されています。

コンテキスト内での 3D モデルの可視化

一般的な制作では、アートディレクターなどさまざまな関係者が新しいビジュアルアセットを総合的にレビューして検証しなければならない場面が何度も訪れることでしょう。マテリアルとアセットがシーン内で確実にうまく機能するようにするには、それらの動作を事前にテストするほかに方法はありません。

Unity のルックデベロップメントソリューションは、**HD レンダーパイプライン（HDRP）**とともに提供される画像ベースのライティングツールであり、**「Window」>「Render Pipeline」>「Look Dev」**で利用できます。このソリューションには、アセットを確認および比較して、それらが各種のライティング条件下でどのように動作するかを検証できるビューアーが用意されています。ルックデベロップメントの初回使用時には、**環境ライブラリ**を新規作成またはロードする必要があります。なお、実際のライティングのシミュレーションを行うには、**高ダイナミックレンジ画像（HDRI）**の方が優れていることに注意してください。



ルックデベロップメントの HDRI 画像

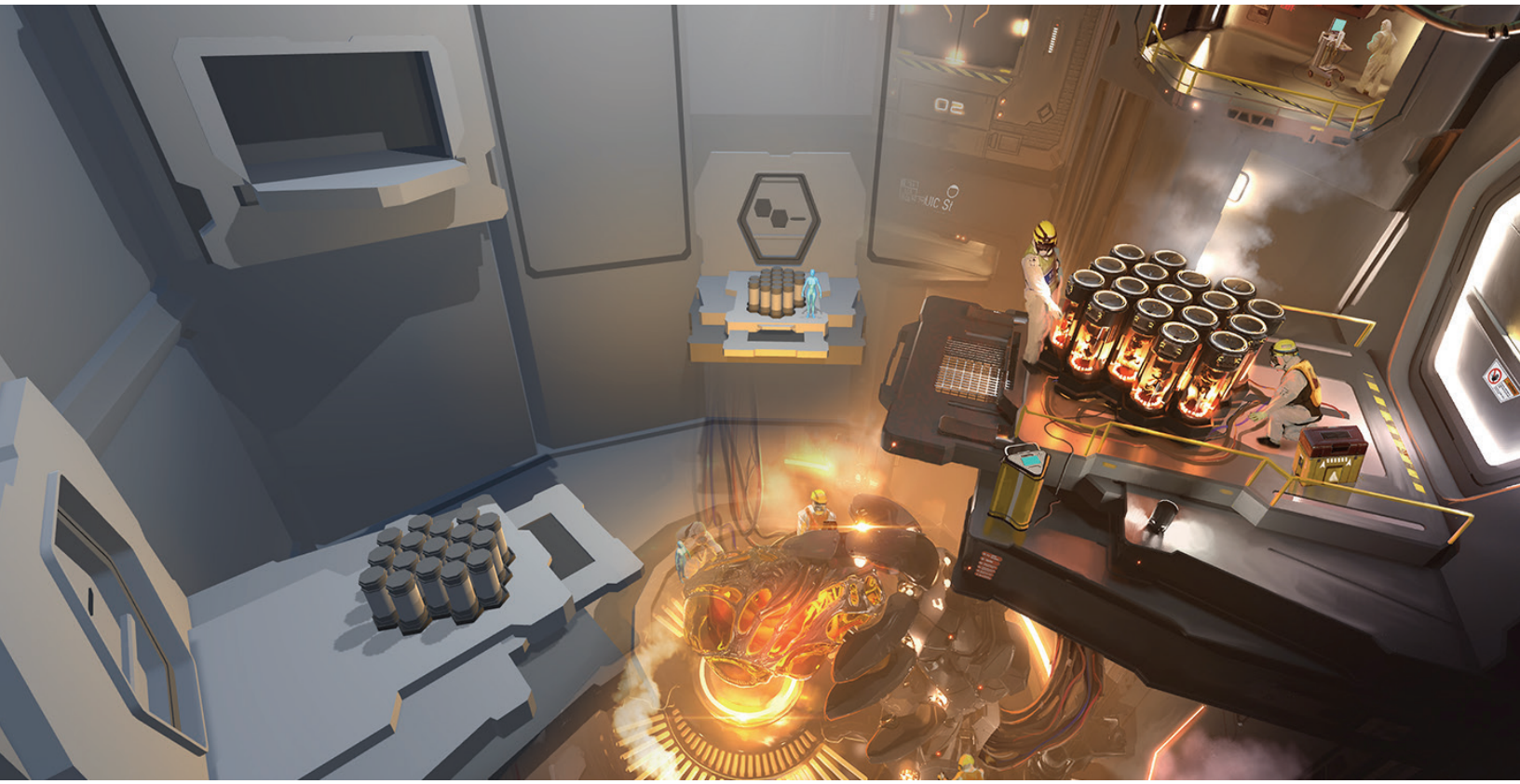
FBX Exporter

FBX Exporter パッケージは、Unity と Autodesk が共同で開発した製品の 1 つです。これを使用すると、Unity、Autodesk® Maya®、Autodesk® Maya LT™、Autodesk® 3ds Max® の間で 3D モデルを移動する際に、円滑なラウンドトリップワークフローが実現します。

FBX Exporter では、アーティストフレンドリーなインターフェースを使用して、Unity のシーンを FBX ファイルにエクスポートして、それらを Maya、Maya LT、または 3ds Max にインポートできます。さらに言えば、Unity に対応した FBX ジオメトリとアニメーションをエクスポートして、それらのアセットに変更を安全にマージして戻したうえで、Unity で作業を継続することができます。

例えば、Unity 内で任意のレベルをブロックアウトしたり、後にアーティストやアニメーターがロードして親しんだ DCC ツールで反復作業をすることが可能になる、任意のアニメーションやカットシーンのプロトタイプも作成したりすることができます。Unity FBX Exporter では、階層、ライト、メッシュ、マテリアル、テクスチャ、カメラのパラメーターがサポートされています。

ただし、パフォーマンス重視のゲームやグラフィックスの負荷が高いゲーム（オープンワールドなど）を扱っている場合は、早期段階で確実に、ゲームがその複雑さに対応できるようにします。プロトタイピングアセットは、制作アセットの最終的な仕様で作成できます。例えば、モックアップアセットは、最終的なアセットバジェットと同じ三角形数または同じシェーダー設定で作成できます。



左側のグレーボックスのアセットは Unity ProBuilder で作成されたものです。一方、右側の最終的なアセットは、DCC ソフトウェアで仕上げられ、Unity にインポートして戻されたものです。

その他の DCC ツールの操作

Unity の内部では FBX ファイル形式を使用します。そのため、可能な場合は制作中、このファイル形式を使用することをお勧めします。プロプライエタリモデルファイル形式は避けてください。唯一の例外は、Blender などの DCC ソフトウェアを使用している場合です。この場合は、Unity の 3D アセットを .Blend ファイルとしてプロジェクトのアセットフォルダー内に保存できます。ただし、制作チームの全員が同じバージョンの DCC ソフトウェアをインストールして使用する必要があります。

Unity では、メッシュとすべてのノードが、保存済みの位置、回転、スケールでインポートされます。また、ピボットポイントと名前も、頂点、ポリゴン、三角形、UV、法線、ボーン、スキンメッシュ、アニメーションと一緒にインポートされます。ユーザーがその他のサポート対象 DCC ソフトウェアでアセットに対して反復作業を行うと、ユーザーがファイルを保存するたびに、対応するゲームオブジェクトが Unity で更新され、ユーザーによる変更が Unity エディターに反映されます。

Blender などの DCC ソフトウェアとのラウンドトリップ以外では、**Cloud Build** などの Unity 機能との互換性を最大限に高めるために FBX ファイルを使用することをお勧めします。Unity でシーンのプロトタイプまたはグレーボックスを作成する際は、FBX としてエクスポートされたアセットをアーティストが自分の DCC ツールで操作できます。Unity の**プロジェクト設定**に移動して、**エクスポート**のデフォルト設定を**FBX オプション**に調整しておきましょう。

その他のリソース

Maya と Unity を使ったアーティストワークフロー
Unity と Autodesk の間を簡単に行き来する
アセットデータベースをより効果的に活用するためのヒント

レンダーパイプライン

Unity の柔軟なグラフィックス機能は、モバイルからデスクトップやハイエンドコンソールまで、さまざまなプラットフォームでシャープに最適化された外観を作成するための洗練された制御レベルを提供します。レンダーパイプラインとは、シーンのコンテンツを取得して画面上に表示する一連の処理を実行するものです。大まかに言うと、カリング、レンダリング、ポストプロセスの処理を行います。

Unity には、多様な目的に対応できるように 3 つのレンダーパイプラインが用意されています。1 つは旧式の**ビルトインレンダーパイプライン**で、2 つは新しい**スクリプタブルレンダーパイプライン（SRP）**です。後者は、デフォルトのレンダリングパイプラインとなる予定の**ユニバーサルレンダーパイプライン（URP）**と、**HD レンダーパイプライン（HDRP）**です。また、カスタムの SRP を作成することもできます。レンダーパイプラインの選択は、ターゲットプラットフォームや自分が求めている現実感のあるビジュアルのレベル、カスタマイズの内容に応じて決まります。

レンダリングパス

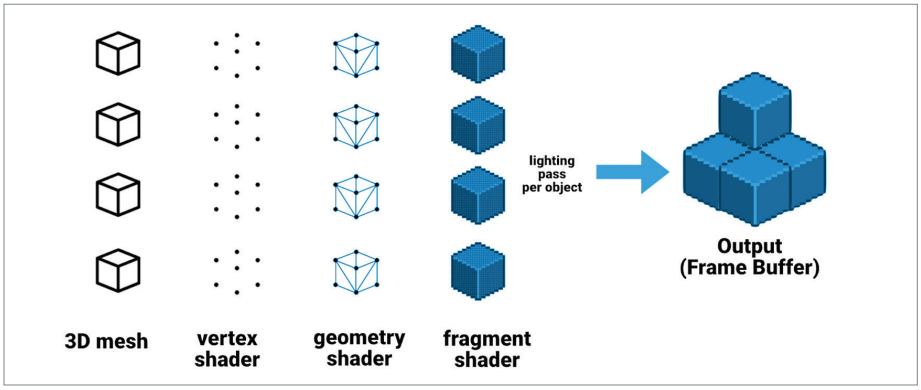
レンダリングパスは、ライティングとシェーディングをレンダリングするために使用される一連の処理です。さまざまなレンダリングパスがあり、それぞれが異なる機能やパフォーマンス特性を備えています。Unity のレンダーパイプラインでは、以下のレンダリングパスが使用できます。

フォワードレンダリング

フォワードレンダリングとは、ビルトインレンダーパイプラインと URP におけるデフォルトのレンダリングパスです。オブジェクトのピクセルへ影響を与えるリアルタイムライトの数を制限する、汎用的なレンダリングパスです。

オブジェクトに影響しているライトには、それぞれコストがあります。これらが一定の数を超えると、ライトは近似によってオブジェクトへ影響することになります（URP では、全ライトがピクセル単位で計算されます。また、この制限は変更可能）。プロジェクトで多数のリアルタイムライトを使用するわけではない場合は（モバイルまたは VR のプロジェクト）、このレンダリングパスが良い選択と思われます。理由としては、シーンのすべてのライトについてフルスクリーンパスが実行されることがないためです。この場合、タイルベース GPU に対するコストが高くなってしまいます。

ただし、HDRP のタイル / クラスターアーキテクチャであっても、多数のライトがフォワードレンダリングで各オブジェクトに影響する可能性があります。そのため、大抵 2 つのレンダリングパス間で同等の機能を提供しています。また、このアーキテクチャは、ディファード G-buffer レンダリングよりはやや精密なレンダリング品質を提供していることもあり、パフォーマンスよりも現実感のあるビジュアルが優先される HDRP プロジェクトに適しています。詳細については、HDRP のレンダリングパスのドキュメントを参照してください。

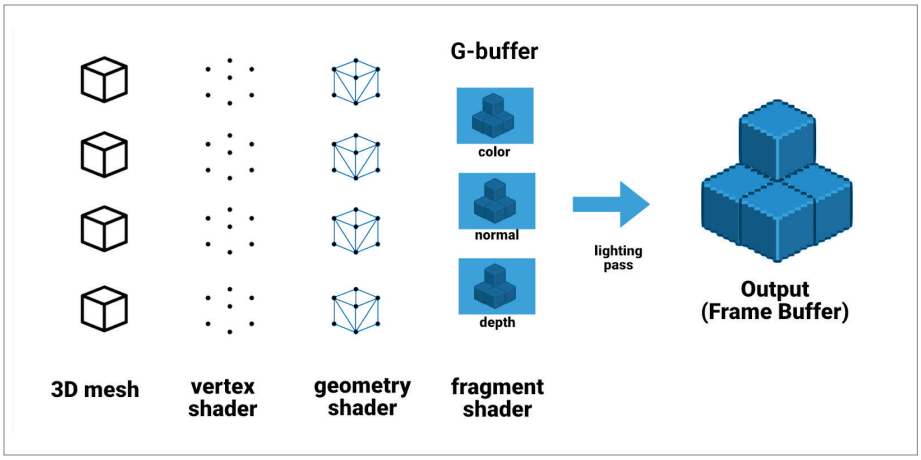


フォワードレンダリングパス

ディファードシェーディングレンダリング

ディファードシェーディングレンダリングでは、デスクトップや、コンソール、ハイエンドモバイルデバイスに大抵搭載されている GPU 機能に対して、より高いレベルのサポートが必要になります（ただし、一部のレンダリング機能には制限があります）。プロジェクトでハイエンドモバイルや、PC、コンソールプラットフォームをターゲットとしており、リアルタイムライトをいくつか、そしてライティングの忠実度を高度に求める場合に、このレンダリングパスは、ビルトインレンダーパイプライン、もしくは URP と併用することで最適なものとなるでしょう。

ディファードシェーディングレンダリングは、HDRP のデフォルトのパスです。これは、HDRP プロジェクトでは多数のリアルタイムライトとフルスクリーンエフェクトが使用されることが予想されるためです。HDRP では、ディファードシェーディングがパフォーマンスと品質のバランスを適正に保ちます。



ディファードシェーディングレンダリングパス

ユニバーサルレンダーパイプライン

URP は、モバイル、PC、コンソール、WebGL、VR など、幅広いプラットフォーム上でグラフィックスパフォーマンスを最大限に最適化します。この最適化が可能なのは、ライティングとシェーディングについて一部妥協しているためです。一定の制限がかかり、ローエンドのデバイスでサポートされていない機能は無効になります。これにより、開発者は自分の作業を最適化する方法について思いわずらう時間が減り、より多くのオーディエンスに届くようにプロジェクトでの開発にさらに専念することができます。

URP にはスケーラブルであるというメリットもあります。上級の開発者であれば、URP をカスタマイズし、プロジェクトで特定の目的に合わせてレンダリングを思うようにコントロールできます。前述のとおり、URP はビルトインレンダーパイプラインに代わって Unity のデフォルトのレンダーパイプラインになる予定です。

Unity での URP のデモ

URP ではデフォルトでライトループによるレンダリングを行うため、シングルパスでフォワードレンダリングを実行できます。一方、ビルトインレンダーパイプラインのフォワードレンダリングだと、範囲内でピクセルライトごとに追加のパスを実行します。つまり、URP を使用することでドローコールが少なくなるということです。

URP は、2D Renderer、2D Lights と 2D ライティングエフェクト、ピクセル化された外観のスタイルをプロジェクトに実装するための 2D Pixel Perfect Camera を備えており、Unity での 2D ゲーム開発に最適なソリューションです。

URP の使用を開始するには、ドキュメントを確認し、Unity 2021 LTS 以降の **Unity Hub** にアクセスしてください。

HD レンダーパイプライン

HDRP（HD レンダーパイプライン）は、コンピュートシェーダーと互換性のある最新のプラットフォームをターゲットとして使用される忠実度の高い SRP です。HDRP の機能を使うことで、アーティストや開発者は高解像度の外観を短時間で作りあげることができます。AAA 品質のゲーム、自動車のデモ、建築用アプリケーション、バーチャルプロダクション、映像、および高精細グラフィックスを必要とするその他のコンテンツには、HDRP を使用します。

『*I Am Fish*』。Bossa Studios Ltd. が HDRP を使用して制作。Steam、Xbox でリリースされています。

HDRP は、3 つの大原則に従います。それは、物理ベースレンダリング（PBR）、一元的かつ一貫したライティング、そして、独立したレンダリングパスです。新しいレンダリングプロセスとシェーダー、そしてシーン内でのライティングへの大規模な改善は、全般これらに基づき、作られています。

このパイプラインは、ディファード / フォワードとタイル / クラスターレンダラーを組み合わせて使用するため、その他のレンダーパイプラインを使用する場合よりも効率的にライティングをスケールできます。HDRP では、ディファードレンダリングとフォワードレンダリングは同等の機能を備え、効果の品質と正確さにほとんど違いがありません。HDRP は**カスタムパス**と**カスタムポストプロセッシング**エフェクトを使用してカスタマイズできます。

HDRP は、3D PC や、コンソールグラフィックス向けの最高品質で現実感のあるビジュアルを追求している、小規模な開発チームが利用することを考慮して設計されているものです。モバイルプラットフォームには適していません。

『*LEGO Builder’ s Journey*』。Light Brick Studios が HDRP と NVIDIA DLSS によるレイトレーシングを使用して PC およびコンソールバージョン向けに制作したゲーム。

HDRP の詳細な紹介については、ドキュメントとこのプレゼンテーションをご覧ください。

カスタムレンダーパイプラインの作成

プロジェクトのレンダリングに対する制御を強化するには、カスタム SRP または URP および HDRP のカスタマイズバージョンを作成します。

カスタム SRP を作成するには、Unity が提供する **SRP Core API** を使用することになります。この API には、プラットフォーム固有のグラフィックス API を操作するための定型コード、一般的なレンダリング処理用のユーティリティ機能、および URP と HDRP の両方が使用するシェーダーライブラリなど、独自のレンダーパイプラインを作成するのに役立つ再利用可能なコードが含まれています。

カスタム SRP を作成するには、レンダリングコマンドを設定し、スケジューリングする C# スクリプトを記述する必要があります。**ScriptableRenderContext** クラスは、C# スクリプトと Unity の低レベルのグラフィックスコードの間のインターフェースとして機能します。

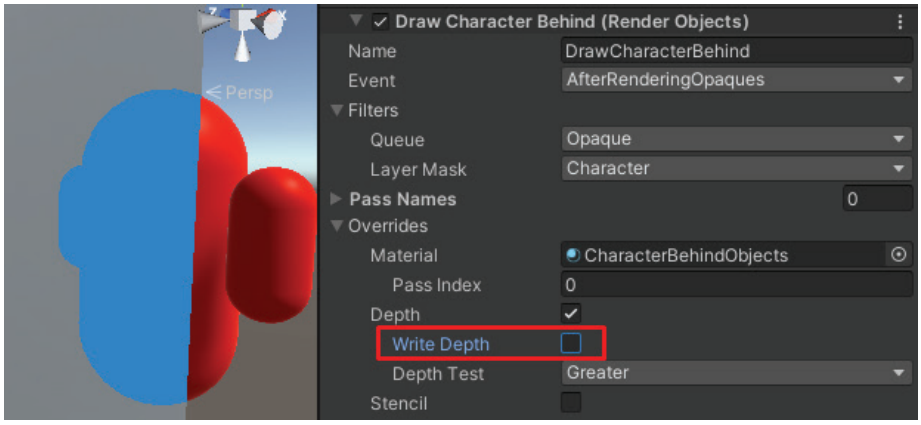
SRP レンダリングは遅延実行によって動作します。最初に ScriptableRenderContext を使用してレンダリングコマンドのリストを作成します。次に、Unity にそれらのコマンドを実行させ、そうして、ようやく低レベルのグラフィックスアーキテクチャがグラフィックス API に命令を送信します。

Unity のドキュメントには、URP と HDRP のカスタムバージョンを含む、カスタムレンダリングを作成するための命令が記載されています（次のセクションを参照）。SRP ソースコードは、GitHub でも入手できます。

URP のカスタムレンダーパス

URP では、**レイヤーマスク**を使用して、カスタムレンダーパスで特定のオブジェクトのセットをレンダリングする方法を定義することができます。

Unity 2021 LTS 以降のバージョンでは、URP 用の **Render Objects Renderer** において、レンダリングするゲームオブジェクトをフィルターするレイヤーをベースとした上で、カスタムレンダーパスにアクセスできます。このため、アーティストはコードを記述することなく、壁の向こうにいてカメラからは見えないキャラクターのシルエットをレンダリングするなど、高度なビジュアルエフェクトやゲームプレイメカニクスを作成できます。実際の例については、このビデオチュートリアルをご覧ください。



上の例では、「Opaque Layer Mask」でオブジェクトに異なるマテリアルが適用されています。

HDRP では、カスタムパスは少し異なる方法で実行されます。詳細な情報については、ドキュメントとサンプルプロジェクトを確認してください。

動的解像度とアップスケーリング手法

ゲームで一般的なパフォーマンス最適化の 1 つに動的解像度があります。Unity では、ビルトインレンダーパイプライン、URP、および HDRP において、カメラ設定として用意されています。GPU バウンドの結果としてアプリケーションのフレームレートがドロップしそうな場合、一定のフレームレートを維持するために解像度が徐々にスケールダウンされます。スクリプトを介して手動でこれを発動することもできます。

HDRP では、**NVIDIA ディープラーニングスーパーサンプリング（DLSS）**（サポートされる NVIDIA GPU のみ）、**AMD FidelityFX ™ Super Resolution（FSR）**、**Temporal Anti-Aliasing（TAA）Upscale** など、さらに高度なアップスケーリング手法を使用できます。URP では **AMD FSR** もサポートされています。

NVIDIA RTX GPU および Windows 向け NVIDIA DLSS

Unity 2021 LTS 以降のバージョンの HDRP では、NVIDIA DLSS がネイティブにサポートされます。NVIDIA DLSS は高度な AI レンダリングを使用しており、ほんの少しピクセルを従来同様にレンダリングするだけで、ネイティブ解像度に匹敵する画質を実現することができます。リアルタイムレイトレーシングと NVIDIA DLSS により、より高いフレームレートと解像度で動作する美しい世界を NVIDIA RTX GPU 上で作成することができます。また、DLSS は従来のラスタライズされたグラフィックスに対しても、大幅な性能向上を実現しています。詳細については、NVIDIA の Unity 開発者ページ、ブログ、ドキュメントを参照してください。

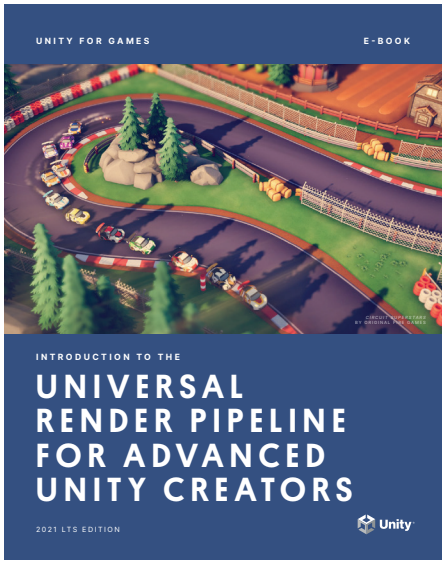
24 Entertainment の『*Naraka: Bladepoint*』などのゲームを 4K で動作させる DLSS 技術の内部的なしくみについて詳しくは、こちらのブログをご覧ください。

AMD FidelityFX Super Resolution（クロスプラットフォーム）

Unity 2021 LTS では AMD FidelityFX Super Resolution（FSR）を利用できます。これには、HDRP と URP の両方に対するビルトイン FSR サポートが含まれています。FSR を使用するには、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**オプションから「**FidelityFX Super Resolution 1.0**」を選択します。

FSR は、低解像度の入力から高解像度のフレームを生成するためのオープンソースの高品質ソリューションです。高品質のエッジを作成することに重点を置いたアルゴリズム群が使用されており、特にネイティブ解像度で直接レンダリングする場合と比較して大幅な性能向上が得られます。言い換えると、FSR は、ハードウェアレイトレーシングのようなコストのかかるレンダリング処理に対して「実用的なパフォーマンス」を実現します。詳細については、AMD の Unity ウェブページおよびフォーラムを参照してください。

Unity のデモ『*Spaceship*』で AMD FidelityFX Super Resolution（FSR）テクノロジーがどのように使われているかをご覧ください。このデモは、Steam から入手できます。



上級 Unity クリエイター向け URP 入門

このガイドは、プロジェクトで URP を利用したいと考えている Unity 初心者および経験者の両方に対応しています。URP の設定、レンダラーの機能、シェーダーコーディング、ビジュアルオーサリング、ライティング手法、パフォーマンスに関する考慮事項などのトピックについて説明しています。

e ブックをダウンロード

その他のリソース

URP のチュートリアルでゲームグラフィックス技術を磨く
URP と GPU ライトマッパーでライトを活用する
HDRP のチュートリアルで驚くように美しいグラフィックスを作り出す
新しい HDRP テンプレートを探検して、学習や制作に活用する

シェーダー

Unity では、シェーダーは主に次の 3 つのカテゴリに分けられます。

— **グラフィックスパイプラインの一部であるシェーダー：**画面上のピクセルの色を決定するための計算を実行する、最も一般的なタイプのシェーダーです（例については、サーフェスシェーダーを参照してください）。

— **コンピュートシェーダー：**通常のグラフィックスパイプラインからは、外にある GPU 上で計算を実行します。

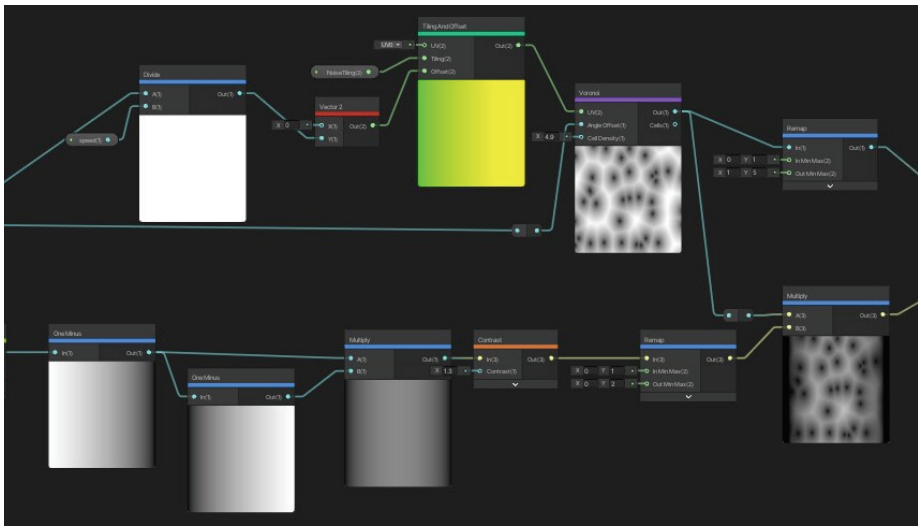
— **レイトレーシングシェーダー：**主にライティングを生成するために使用されます。

シェーダービジュアルオーサリング：シェーダーグラフ

Unity のシェーダーは **ShaderLab** と呼ばれる Unity 固有の言語で記述されますが、シェーダーグラフを使用して視覚的にシェーダーを作成することもできます。エディターでコードを作成して保存し、コンパイルしてテストする代わりに、シェーダーグラフのフレームワークでノードを作成して接続します。マテリアルの状態をリアルタイムで確認できるため、開発者はその場で即時に変更を行い、テストすることができます。

シェーダーグラフアセットには、さまざまなマテリアルの事前設定済みのオプションが用意されています。シェーダーグラフのノードは、マテリアルが適用されるオブジェクトに関するデータ（数学的関数、プロシージャルパターンなど）を表します。

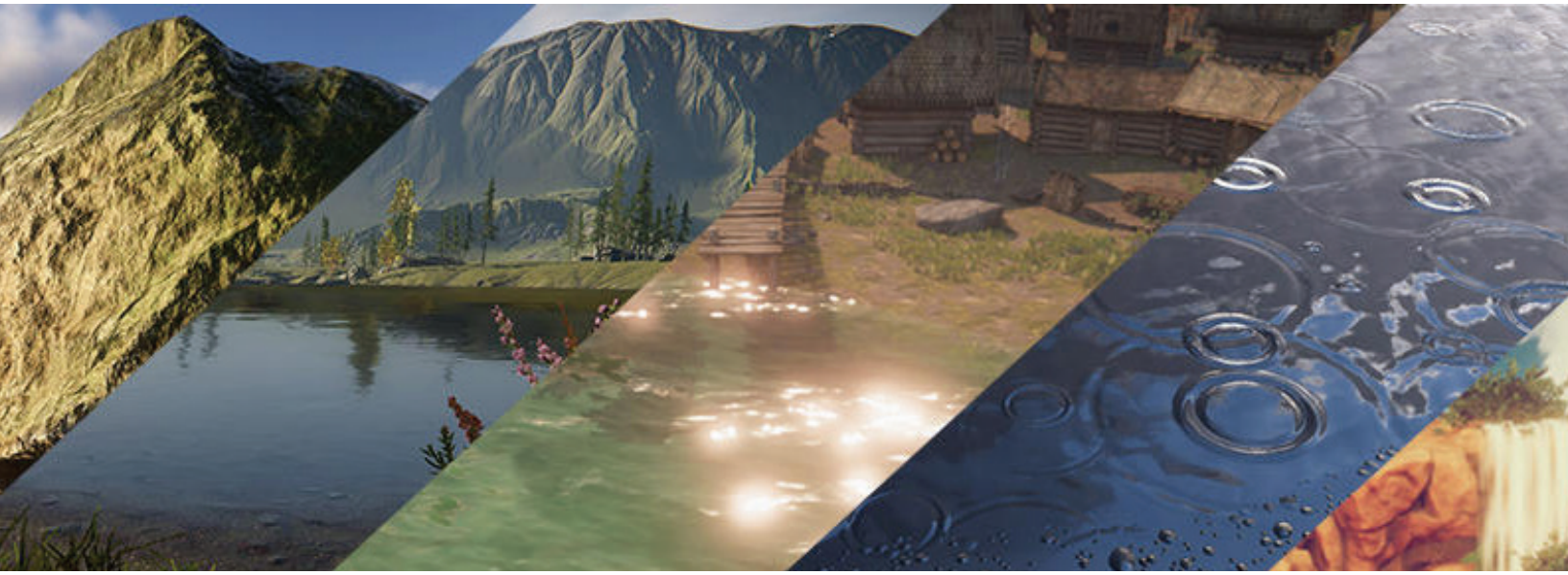
加えて、シェーダーグラフシステムは拡張可能であるため、プログラマーはカスタムのシェーダーグラフノードを開発できます。URP と HDRP（および 2021 LTS 以降のビルトインレンダーパイプライン）で機能するシェーダーを作成するにはシェーダーグラフを使用します。



シェーダーグラフでのノードベースのシェーダー作成

Master Stack は、シェーダーのサーフェスの最終的な見た目を定義するシェーダーグラフのエンドポイントです。これは、頂点ステージ（ポリゴンの頂点の属性が計算されるとき）とフラグメントステージ（頂点間のピクセルの見た目がどのようになるかを確認するための計算が行われるとき）で行われる処理の関係性を具象化するのに役立ちます。

HLSL コードの作成経験があるユーザーは、シェーダーグラフで**カスタム関数**ブロックを作成できます。



シェーダーグラフを使用すると、アーティストや他のチームメンバーは、グラフネットワークでノードをつなぐことでシェーダーを制作できます。

SRP のコンピュートシェーダー

コンピュートシェーダーは、通常のレンダリングパイプラインの外にある GPU 上で実行されます。超並列 GPU アルゴリズムや、レンダリングの部分的な高速化のために使用することができるものです。

コンピュートシェーダーを効率的に使用するには、GPU アーキテクチャと並列アルゴリズム、DirectCompute、OpenGL Compute、CUDA、OpenCL についての深い知識が必要です。これらのシェーダーは、互換性と汎用性が高く、すべての Unity レンダーパイプラインに実装できます。

ビルトインレンダーパイプライン向けのサーフェスシェーダー

名前が示すように、サーフェスシェーダーはマテリアルの物理特性を定義します。マテリアル内の各ピクセルの最終的な色を計算し、サーフェス上の各ピクセルのシェーディングを定義するライト計算を実行します。Unity の大部分のサーフェスシェーダーは、デフォルトの**標準サーフェスシェーダー**を拡張したものです。このため、作成プロセスがより直感的で、アーティストはサーフェスの外観をより自由に定義することができます。

その他のリソース

アーティスト向け最新ビジュアルエフェクトガイド
シェーダーグラフの Master Stack
シェーダーグラフで実験しよう：少ないリソースで多くのことを実現するシェーダーグラフの表面勾配フレームワークを使用した法線マップコンポジティング

UNITY のライティング

グローバルイルミネーションによるライティングエフェクト

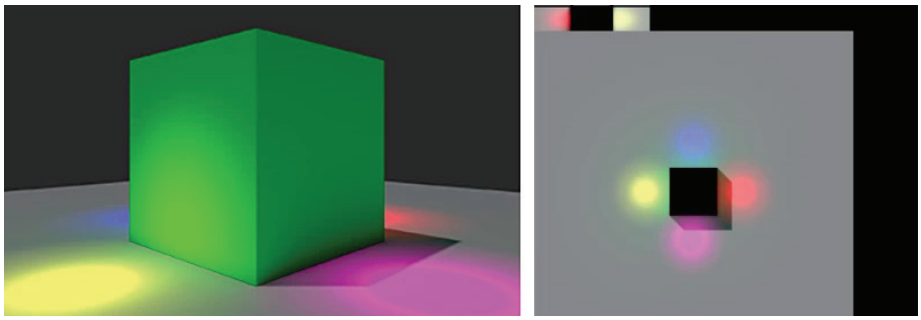
グローバルイルミネーション

現代のゲームのライティングは、グローバルイルミネーション（GI）をうまく利用しています。GI には、光が反射し、対象のワールドと相互に作用する際の複雑な動作のシミュレーションを試みるための、さまざまな手法と数学モデルが取り揃えてあります。グローバルイルミネーションのシミュレーションを正確に行うのは難しく、計算の負荷が高くなる場合があります。このためゲームでは、ゲームプレイ中ではなく事前にこういった計算を処理するさまざまなアプローチがよく利用されています。

一般に、Unity のライティングにはリアルタイム（直接光）と事前計算がありますが、両方のアプローチを組み合わせることで臨場感のあるライティングを作成できます。

ベイクしたグローバルイルミネーション

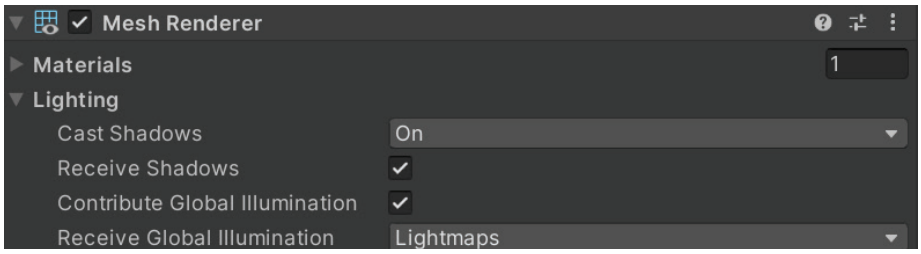
ライトマップをベイクすると、シーン内の静的オブジェクトに対するライトの効果が計算され、その結果がシーンジオメトリの上にオーバーレイされるテクスチャに書き込まれて、ライティングの効果が作成されます。



静的オブジェクトのライトマップ

Unity では、事前計算済みライティングはバックグラウンドで自動的に生成するか、手動で開始することになります。ライトマップに影響を及ぼすオブジェクトを移動または編集しない限り、これらのプロセスをシーンの背後で実行したまま、エディターでの作業を続けることができます（移動や編集をした場合は、ベイクプロセスが再度開始されます）。

GI システムは、**Mesh Renderer** コンポーネントで、「**Contribute Global Illumination**」プロパティを選択しているオブジェクトのみを考慮します。ライトマップでライティングするには、「**Receive Global Illumination**」プロパティの「**Lightmaps**」オプションを選択します。それ以外の場合、オブジェクトはライトプローブでライティングされます。



GI の主なオプションは、Mesh Renderer にあります。

ゲームオブジェクトがライトプローブから GI を受け取るよう設定した場合、ライティングの表現を正確にキャプチャできるよう、シーン内に**ライトプローブグループ**オブジェクトを作成し、そのプローブを複製してシーンを覆う必要があります。ライトプローブグループは、空間内の多数のポイント（プローブ）からライティングをキャプチャします。ライティングデータはディスクに保存されることになります。その後、実行時にプローブでライティングされる各オブジェクトは、最も近い 4 つのプローブからのライティングブレンドを使用してライティングされます。ライトプローブは、高品質なライティングを必要としない小さなオブジェクトもそうですが、特に動的オブジェクトで便利です。

ベイク時間を短縮し、ライトアップのために予約されるメモリの量を減らすため、シーン内ではプローブでライティングされるオブジェクトを最大限に駆使することをお勧めします。大きな静的オブジェクトや忠実度の高いライティングを必要とするオブジェクトのみ、ライトマップから GI を受け取るようにしましょう。

プログレッシブライトマッパー

プログレッシブライトマッパーは、エディター内のベイクしたライトマップとライトプローブをプログレッシブに更新する、パス追跡ベースのライトマッピングシステムです。エディター内のシーンまたはゲームビューでライトマップをプログレッシブに改良し、表示するため、アーティストは短時間で反復作業を行うことができます。そして、ベイク時間がもっと予測しやすくなります。これは、プログレッシブライトマッパーがベイク中に推定時間を返してくれるからです。

プログレッシブライトマッパーは、アップサンプルスキーム、イラディアンスキャッシュ（放射照度のキャッシュ）、その他のグローバルデータ構造を使用せず、テクセルごとに個別に GI をライトマップ解像度でベイクします。ライトマップの選択した部分をベイクでき、テストとイテレーションがより迅速になるため、強力な機能です。

プログレッシブライトマッパーは、ビルトインレンダーパイプライン、URP、および HDRP で利用できます。

CPU および GPU ライトマッパー

プログレッシブライトマッパーには 2 つのバックエンドのいずれかを選択できます。プログレッシブ CPU ライトマッパーは、コンピューターの CPU とシステム RAM を使用します。プログレッシブ GPU ライトマッパーは、GPU と VRAM を使用します。

プログレッシブ GPU ライトマッパーは現在プレビュー版です。つまり、開発中であり、変更の可能性があります。最新の開発ステータスについては、ドキュメントを確認してください。

リアルタイムグローバルイルミネーション

リアルタイム GI は、時間的な性質を根拠とするので、空を動く太陽や、閉ざされた廊下でゆっくり脈打つようなライトなど、ゆっくりと変化し、コンテンツに大きな視覚的影響を与えるライトに有用です。リアルタイム GI は、パフォーマンスコストや遅延のため、高速で動くライトや特殊効果に使うのは効率的ではありません。ミッドレンジからハイエンドの PC システムおよびコンソールのゲームに適しています。また、リアルタイムライトマップ用の低解像度の小さなシーンに使用する場合であれば、ハイエンドモバイルデバイスのゲームにも好適です。

Enlighten

Enlighten は、「Lighting」ウィンドウで設定する、Unity のリアルタイム GI のバックエンドです。このシステムは、静的オブジェクトに対してシーンの事前計算を必要としますが、同様にリアルタイムでのライトとマテリアルのシームレスな補完を行う際のサポートをしてくれます。シーンの事前計算が完了した後は、ライティングのイテレーション時間が大幅に短縮されます。

ビルトインレンダーパイプラインは Enlighten を提供しており、Unity 2021 LTS の時点では HDRP と URP も Enlighten をサポートしています。

レイトレーシングされたグローバルイルミネーション

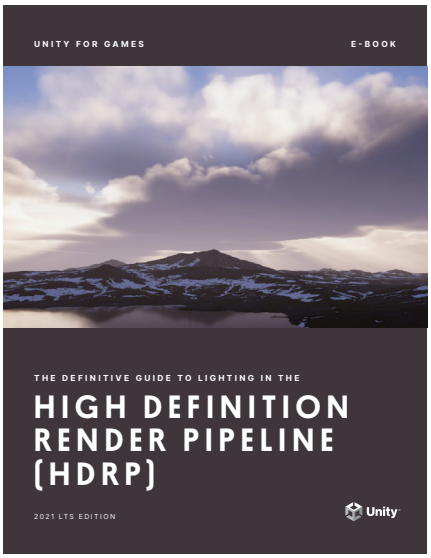
リアルタイム GI のもう 1 つの形は、レイトレーシングを通じて実現できます。現在 HDRP のみがこの機能を提供しており、レイトレーシングと互換性のある GPU が必要です。

Enlighten や従来のベイクしたライトマップと比較した場合、レイトレーシングされた GI には事前計算が不要という利点があります。さらにライティングがピクセル単位であるため、特定のテクセルの解像度に依存せず、最適なライティングを実現するための特定の UV レイアウトも必要ありません。イテレーション時間は非常に短くなりますが、レイトレーシングされた GI の GPU 要件は非常に高くなっています。

そのため、この手法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をサポートする特定のハードウェアでのみ使用できます。

『*LEGO Builder’s Journey*』。Light Brick Studio が PC プラットフォーム向けに制作し、HDRP、レイトレーシング、ディープラーニングスーパーサンプリング（DLSS）を使用してリアルタイムにレンダリングします

レイトレースされた GI、リフレクションなど、Unity で利用できるすべてのレイトレーシングエフェクトに関する詳細については、この動画をご覧ください。



HD レンダーパイプライン（HDRP）のライティングを有効活用するための最も信頼のおけるガイド

ゲーム開発においてライティングは最も複雑なトピックの 1 つです。HDRP の機能を利用する方法を学習することで、高度に最適化されたフォトリアリスティックなゲームを作成できます。このガイドでは、カメラ、ライト、露出、リアルタイムライティングエフェクト、シャドウの調整の概念について説明しています。

e ブックをダウンロード

その他のリソース

HDRP の設定に詳しくなってパフォーマンスを上げよう
Unity の HDRP のレイトレーシング機能に関する NVIDIA ウェビナーのご案内
グローバル Environment Lighting の設定
ライティングとレンダリングの概要
ライトマップの設定

ワールド構築

Unity は、リッチでスケーラブルな 3D および 2D ワールドを構築し、デザインするための完全なツールセットを提供しています。このセクションでは、プロトタイプから洗練されたアートへと移行する際に役立つ 3D ツールについて説明します。

プロ向けアーティストリグループなど多くのチームが、ゲーム開発の最も一般的なニーズをサポートする新しいツールセットを継続的に開発しています。この取り組みに関する詳細については、本書末尾のページをお読みください。

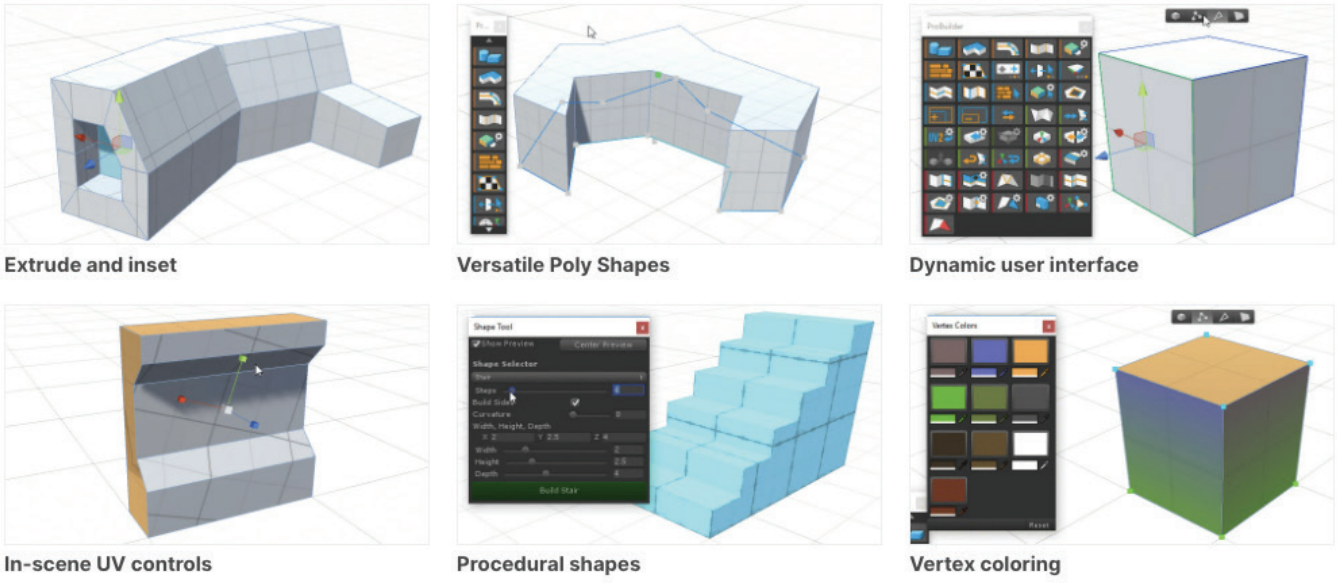
プロトタイピング

ProBuilder

ProBuilder は、3D モデリングツールとレベルデザインツールを組み合わせた他にはないハイブリッドツールです。シンプルなジオメトリ用に最適化されており、詳細な編集機能と UV 展開機能を備えています。

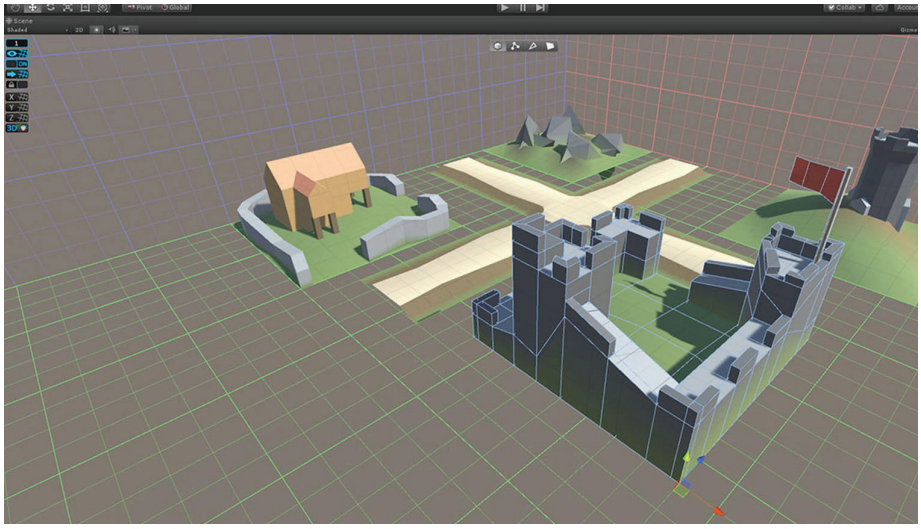
ProBuilder はパッケージマネージャーから入手可能です。ProBuilder を使用すると、構造物や複雑な地形の形状、車両、武器、カスタムの衝突ジオメトリ、トリガーゾーン、およびナビメッシュのプロトタイプ作成を短時間で行うことができます。

ProBuilder には独自のエクスポートオプションがあり、FBX Exporter で Unity のラウンドトリップ機能を使用することもできます。このため 3D アーティストは DCC ソフトウェアでプロトタイプからアセットを完成させることができます。



ProBuilder の 3D メッシュ作成機能の一部

好みのツールで、プロトタイプの形状とサイズの参照を使用し、モデルにディティールを加え、磨きをかけます。Shape ツールでは、事前定義済みの形状を作成できます。このツールには、レベル作成の共通オブジェクトに対応する、標準のジオメトリ形状と複雑なジオメトリ形状の両方のライブラリが含まれています。また、2D 形状を作成して押し出したり、Bezier Shape ツールを使ってスプラインのラインを包むカーブしたメッシュを作成したりすることもできます。

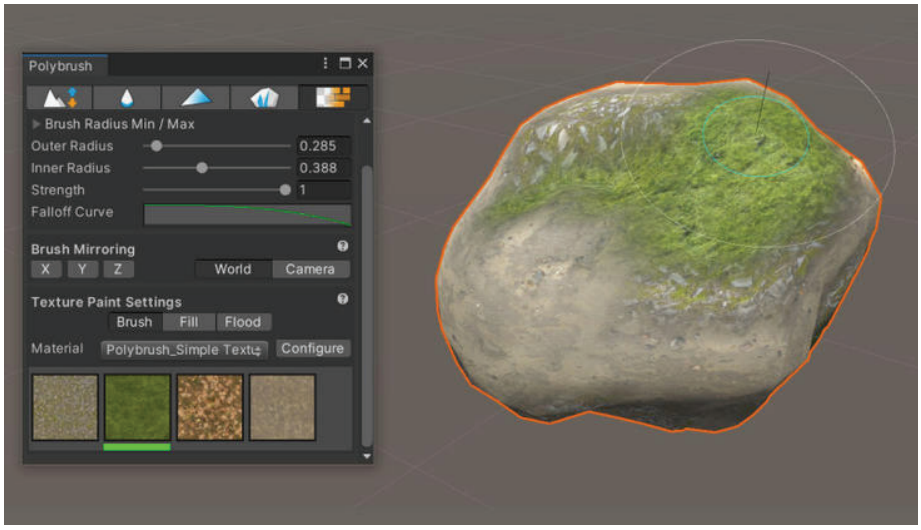


Unity で ProBuilder を使用して作成されたレベルのプロトタイプ

加えて、ProBuilder で新しいメッシュを作成したり、既存のメッシュを変更したりできます。メッシュ全体または選択したフェースにマテリアルを適用して、ゲームプレイ中またはグレーボックス化の際に、よりリアルな外観のサーフェスを作成することができます。例えば、ゲームプレイのテストで床のタイルや壁のレンガを使用する場合などがあります。UV をアンラップまたは編集するには、ツール内で UV エディターを使用します。

Polybrush

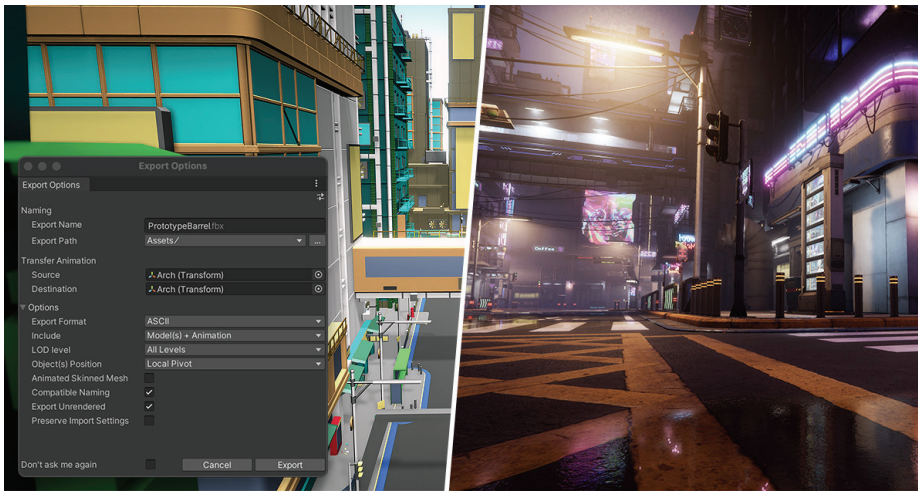
同じくパッケージマネージャーから入手可能な Polybrush を使うと、Unity エディターで直接、テクスチャと色をブレンドしたり、メッシュをスカルプトしたり、オブジェクトを散布させたりすることができます。ProBuilder と組み合わせると、Polybrush は完全にエディター内で完結するレベルデザインソリューションとなります。



Polybrush を使用して、ProBuilder で作成したアセットにディテールを追加します。

微調整に FBX Exporter を使用する

ワークフローで ProBuilder、Polybrush、および FBX Exporter を組み合わせると、モデルとレベルをグレーボックス化でき、プロトタイプ作成と機能テストの両方が容易になります。FBX Exporter を使用すると、微調整のために DCC ソフトウェアにエクスポートする前に、アセットを必要なサイズに調整できます。



ProBuilder で作成したプロトタイプおよび開いた状態の FBX Exporter パネル（左）とアーティストによる最終的な環境（右）

Unity Terrain エディターで作成された地形

Terrain スカルプティングツール

Unity の Terrain エディターを使用すると、ディテールを加え、高度に最適化されたリアルな地形を作成できます。Terrain ツールを使用するには、「**Hierarchy**」ウィンドウで **Terrain** オブジェクトを作成または選択します。

URP および HDRP による新しいデモシーン。作成方法についてはこのブログ記事を参照してください。

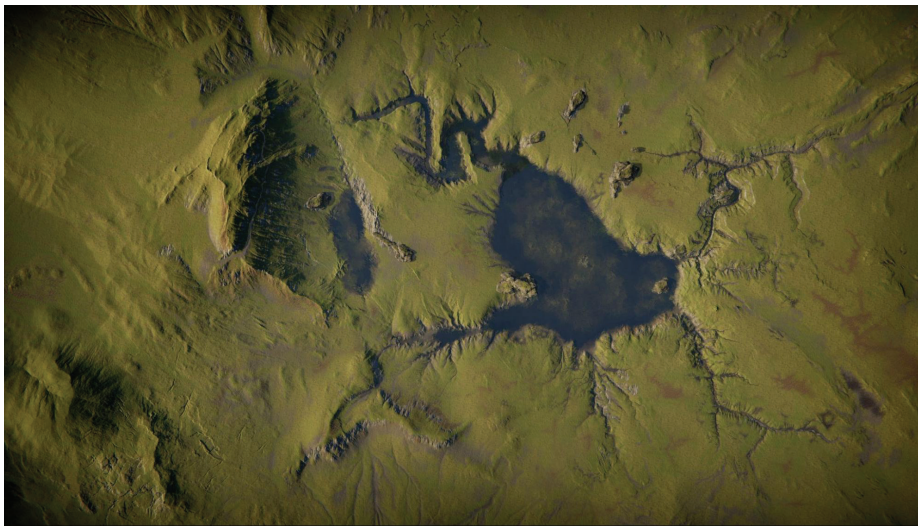
Terrain コンポーネントは、Paintbrush ツールで地形のハイトマップをペイントする際に、地形を上げ下げするのに使用できるブラシを提供しています。このコンポーネントは、地形の一部を隠したり、現在のハイトマップの上にスタンプブラシを追加したり、単に地形を作り込んだりするのに使用します。

このコンポーネントを利用して、地形のジオメトリにサーフェス変更を適用するテクスチャをペイントできます。Terrain エディターには、植生、樹木、地形の描画距離を定義する設定に加えて、メッシュ解像度、ライティング、および風の効果を制御する設定も含まれています。

Terrain Tools パッケージは、息づくような地形のアセットを作ることができる、地形スカルプティングブラシとツールをプロジェクトへ増補してくれるだけでなく、ワークフローも容易にしてくれます。他の Unity ツールと同様に、**Terrain API** はカスタムエディターツールを作成するのに役立つため、チームに最適な方法で機能を使用できます。

Terrain Tools パッケージに含まれている複数のスタンプ

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に関する詳細については、この Unity Learn チュートリアルシリーズをご覧ください。



ブラシとスタンプを使用した地形の作成。

樹木と植生

Unity は Terrain コンポーネントの一部としてブラシを提供しています。このブラシを使用すると、エディター内で樹木エリアや地形のディテールをデザインできます。これは、ディテールを加えた大規模な森林やジャングルを作成するのに便利です。色やサイズのバリエーションを追加することで、環境はより自然な外観になります。

ディテールブラシは、樹木のペイントツールと同じように機能します。これは主にディテールを追加するために使用されます。芝生や石など、マテリアルが 1 つの、シンプルなメッシュを思い浮かべてください。このツールは、**SpeedTree** とも互換性があるため、LOD のスムーズな遷移、高速なビルボーディング、自然な風のアニメーションなど、高度なビジュアルエフェクトを使用した樹木を作成できます。

地形のディテールを持つインスタンス化されたメッシュ

Unity はその他のアセットを処理するのと同じ方法で **SpeedTree アセット**を認識しインポートします。SpeedTree Modeler 7 を使用している場合、**Modeler** の Unity バージョンを使用して、`.spm` ファイルを再び保存します。SpeedTree Modeler 8 を使用している場合は、`.st` ファイルを **Unity プロジェクト**フォルダーに直接保存します。

Terrain を使用した SpeedTree Modeler 8 の葉

地形に風の効果を作成するには、**Wind Zone** コンポーネントを使用して、1 つ以上のゲームオブジェクトを追加します。ウインドゾーン内の樹木、草、その他の植生がリアルなアニメーションのように曲がり、風自体は脈打つように動いて、木々の間に自然な動きのパターンが作り出されます。

Terrain システムのブラシとツールは、ターゲットプラットフォームとアートディレクションに適したパフォーマンスと美観のバランスを取るために使用できる多数のオプションを備えており、これらの効果を作成するのに役立ちます。

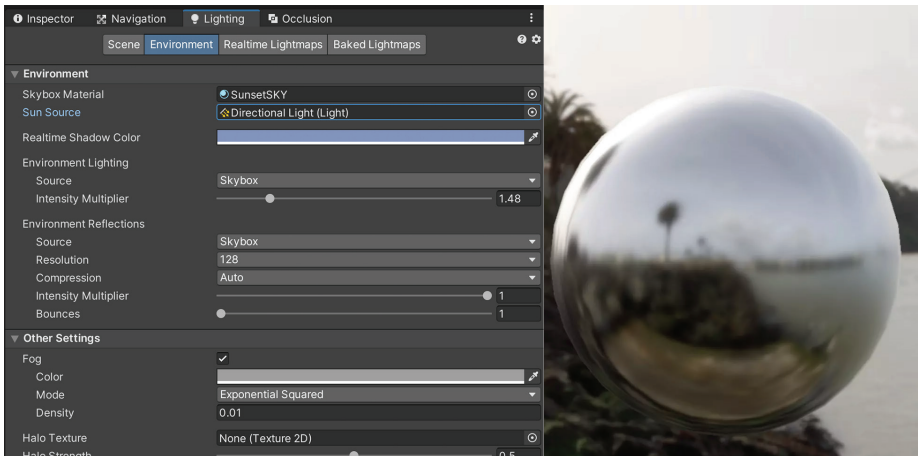
空、フォグ、雲

SRP には、大気エフェクトを設定するツールと設定が含まれています。URP は全体的なパフォーマンスを優先するのに対し、HDRP はフォトリアリスティックなグラフィックスを目的にしています。

URP

「URP Lighting Settings」>「Environment」タブに、大気エフェクトを作成するためのツールセットがあります。

URP は**スカイボックスシェーダー**を使用して、シーン内の環境光に環境テクスチャまたは手続き型で生成された画像を投影します。同じウィンドウで、「**Fog**」、「**Realtime Shadow Color**」、および「**Intensity**」を制御できます。



URP の「Lighting」メニューの「Environment」設定を確認してください。ハロー効果は代わりにレンズフレア（SRP）を使用して実現できるようになりました。

HDRP

HDRP の**ボリュームフレームワーク**を通じて、シーンのプロパティ値をオーバーライドできます。デフォルトでは、「**Project**」設定の「**HDRP Global Settings**」セクションの設定が使用されます。

ボリュームは、**グローバル**にするか、**ローカル**境界を指定することができます。例えば、**ローカルボリューム**を使用して、フォグの色や密度などの「**Environment**」設定を変更すると、シーン内のさまざまな領域の雰囲気を変更できます。「**Visual Environment**」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。

HDRP には、空を生成するための手法として HDRI Sky、Gradient Sky、および Physically Based Sky の 3 つがあります。

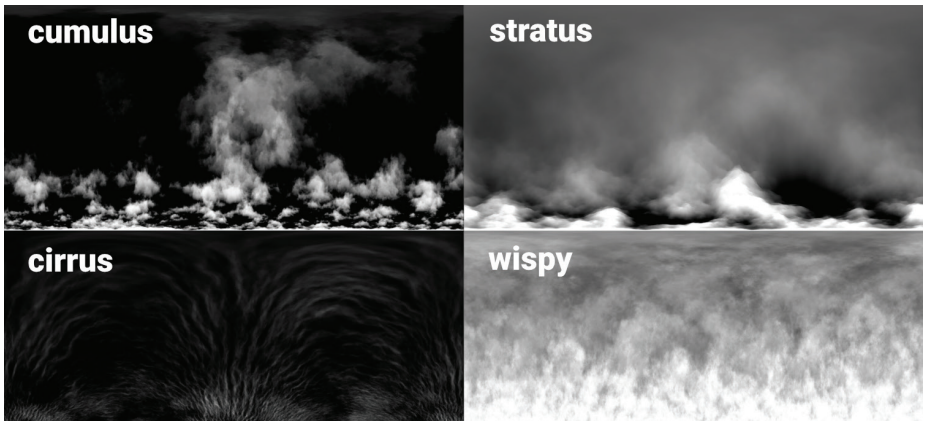
手続き型で生成された空

HDRP 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。また、近くのライトでフォグをライティングする**ボリューメトリックフォグ**機能も実装しています。「Volumetric Fog Distance」では、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、大気は浮遊物質のシミュレーションを行うレイヤーで満たされ、範囲内のゲームオブジェクトが部分的に隠れます。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」を利用できます。

HDRP でのボリューメトリックフォグ

Unity 2021 LTS 以降では、**クラウドレイヤー**システムで自然な外観の雲を生成することにより、Sky と Visual Environment のオーバーライドを補間することができます。

クラウドレイヤーは、フローマップを使ってアニメーション化できる 2D テクスチャです。ベクトルのディスプレイスメントが、赤色と緑色のチャンネルを使用して制御されます。再生モードで空にわずかな動きを加えることで、背景をより動的にすることができます。クラウドレイヤーは空の手前に配置され、オプションで地面に影を付けることができます。



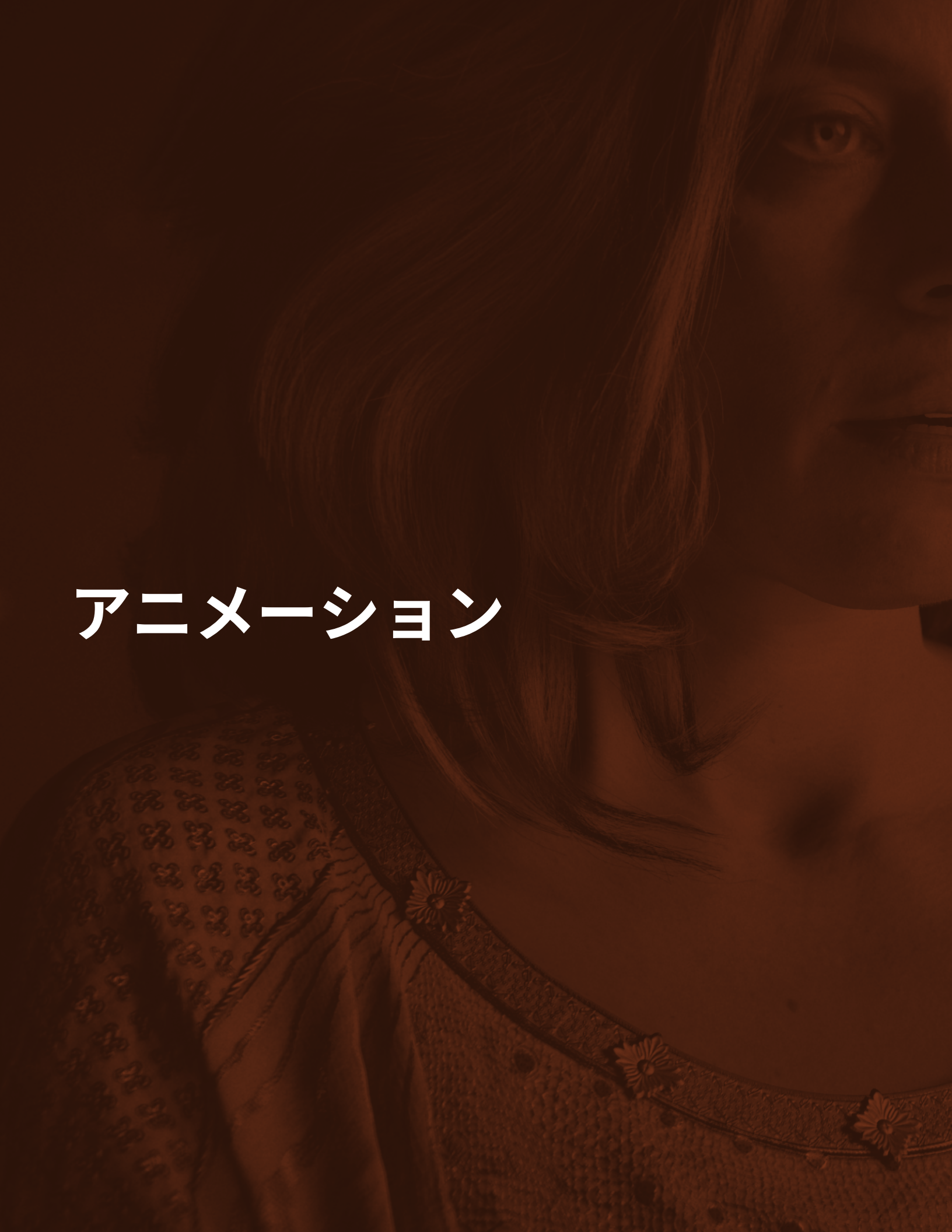
2D テクスチャは円柱投影を使用しており、RGBA チャンネルにはそれぞれ異なる雲のテクスチャ（積雲、層雲、巻雲、薄雲）が含まれています。

ボリュメトリッククラウドは、ライティングや風に反応する、厚みのあるテクスチャを持ったリアルな雲を作り出します。

雲と光を相関させる必要がある場合は、**ボリュメトリッククラウド**を使用します。影をレンダリングしたり、フォグを活用したり、ボリュメトリックライトシャフトを作成したりすることができます。これらをクラウドレイヤーの雲と組み合わせたり、別々に追加したりします。

その他のリソース

Unity ゲームデザイナープレイブック
ProBuilder や Polybrush でレベルデザインを高速にする
FBX Exporter、ProBuilder、Polybrush によるアセット管理 HDRP と URP に対応した新しい Unity Terrain Demo シーンを体験する
新しいライティング機能：ボリュメトリッククラウド、レンズフレア、ライトアンカー
HDRP 2021 LTS のライティングを有効活用するための最も信頼のおけるガイド『*The Heretic*』のメイキング：環境アート


アニメーション

キャラクターのリグは、一組のコンストレイントを使用するもので、環境に対して反応します。

> **ノート：**Unity のアニメーションシステムをよくご存知の方は、このセクションはスキップして「Animation Rigging」に進んでください。

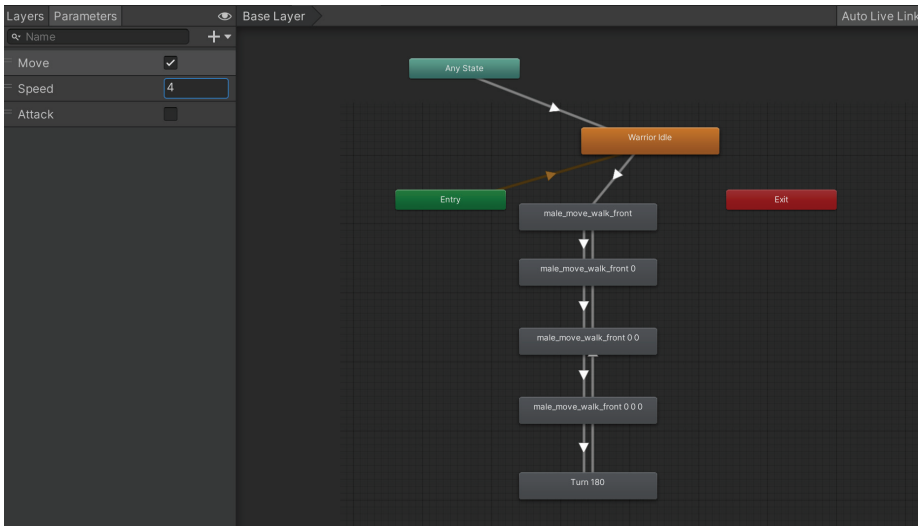
アニメーションシステム

一般に Unity プロジェクトのアニメーションは、モーションキャプチャから作成されるか、Blender、Autodesk Maya、または Autodesk 3ds Max などのソフトウェアを使用して作成されます。Unity のアニメーションシステムは、このようなアニメーションを調整し、磨きをかけ、手続き型で適合させてブレンドすることにより、ゲームやインタラクティブ体験で生き生きとしたアニメーションを実現してくれます。

Unity のアニメーションシステムは、**アニメーションクリップ**の概念に基づいています。これは、特定のオブジェクトの位置や回転などのプロパティが時間の経過に応じてどのように変化するかについての情報を含んでいるものです。各クリップはそれぞれが単一のリニア（linear）で行った収録物とも考えられます。上で述べたような外部ソースからのアニメーションクリップは、Unity に取り込まれた後（ラウンドトリップのセクションを参照）、**アニメーターコントローラー**と呼ばれる構造化されたフローチャートシステムに編成されます。

アニメーションステートマシン

Unity ではアニメーターコントローラーを使用することで、キャラクターまたはオブジェクトのアニメーションクリップや、関連する**アニメーション遷移**をセットで変化をつけたり、望ましい状態で保存しておけます。ほとんどの場合、複数のアニメーションを用意し、一定のゲーム条件が発生した場合に切り替えるのが普通です。例えばスペースキーが押されるたびに歩くアニメーションクリップからジャンプするアニメーションクリップに切り替えることができます。アニメーターコントローラーは、その中で使用されているアニメーションクリップへの参照を保持しており、アニメーションステートマシンを使用して、さまざまなアニメーションクリップおよびアニメーション遷移を管理しています。ステートマシンは、アニメーションクリップとアニメーション遷移のフローチャート、または Unity 内のビジュアルプログラミング言語で記述されたシンプルなプログラムと考えることができます。



アニメーターコントローラー内のアニメーションクリップとパラメーターの流れ

ブレンドツリーは、複雑な部分を隠しておくのに効果的です。ブレンドツリーはステートを持たず、コードのコールバックも行いません。単純に定義されたパラメーターに応じて、さまざまなクリップをブレンドするだけです。これは、ゲームの他の部分を壊す心配をすることなく、ブレンドツリーを反復処理できるため意義の大きいことです。加えて、ブレンドツリーのアニメーションの大半には動作を関連付けることができないため、複雑なステートの関係を隠しておいておけるので、将来のバグを防止できます。

Unity は、入り組んだステートマシンを管理するための**アニメーションレイヤー**を提供しています。例えばアニメーションレイヤーを使用すると、歩行およびジャンプの下半身レイヤーと、オブジェクトを投げる動作および発射する動作の上半身レイヤーを作成することができます。アニメーションステートは、ビジュアルアニメーションに加えて、サウンドエフェクトや C# コードを発動できます。

「Animation」ウィンドウ

「Animation」ウィンドウを使用すれば、Unity 内で直接アニメーションクリップの作成と変更をすることが可能です。このウィンドウは、外部 3D アニメーションソフトウェアの代わりとして機能し、開発中は必要に応じてシンプルなアニメーションを作成できるよう設計されています。このウィンドウには、アニメーションに必要なツールとして標準的なものは一揃い用意されています（**キーフレーム**、**再生ヘッド**、**アニメーションタイムライン**、**アニメーションカーブ**など）。

動きをアニメーション化するだけでなく、さまざまなマテリアルとコンポーネント（ほとんどすべてのゲームオブジェクトプロパティ）をアニメーション化し、**アニメーションイベント**でアニメーションクリップを拡張します。アニメーションイベントは、つまるところ、エディター内でタイムライン上の特定のポイントで呼び出される関数のことです。

Unity の「Animation」ウィンドウでは、以下もアニメーション化可能です。

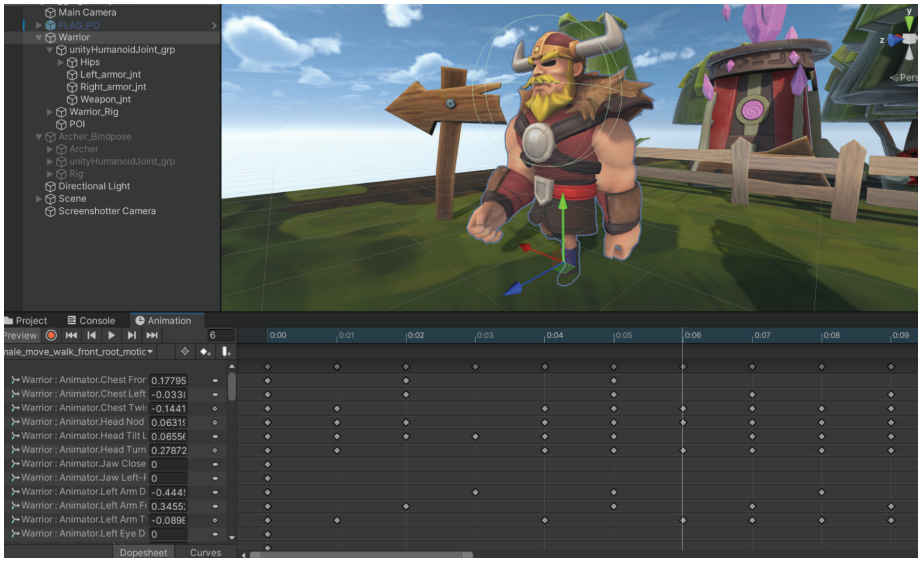
— ゲームオブジェクトの位置、回転、スケール

— マテリアルの色、ライトの強度、音量を含むコンポーネントプロパティ

— Float、Integer、Enum、Vector、Boolean などのスクリプト内のプロパティ

— 独自のスクリプト内で関数を呼び出すタイミング

生成されたアニメーションクリップは、アニメーターコントローラーまたは Animation Rigging で使用でき、ゲームプレイまたはシネマティクスで Timeline によって利用されます。



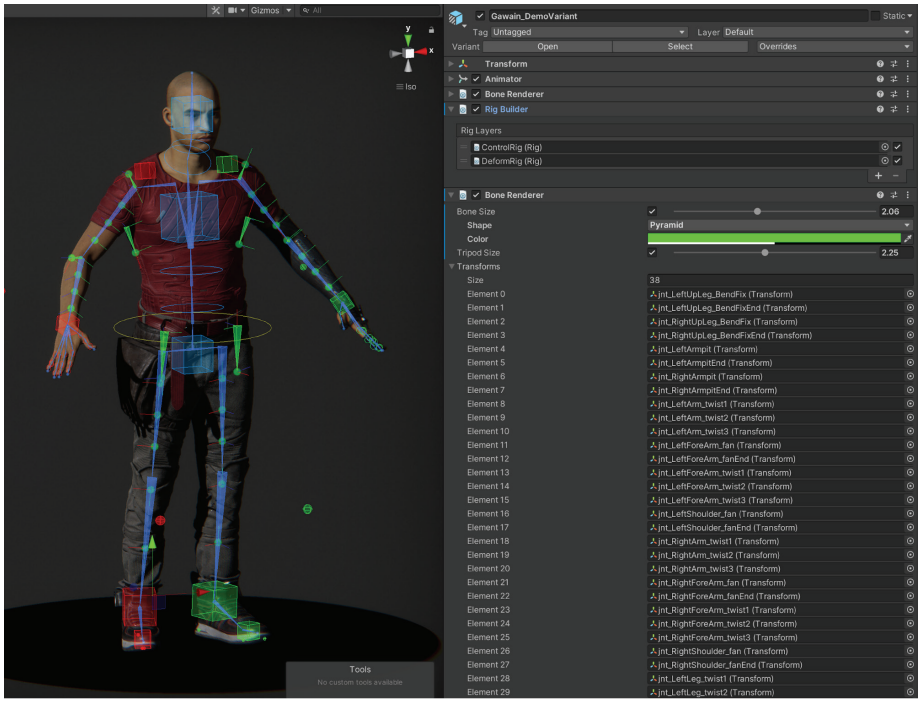
プロジェクトでキーフレームやカーブなどのアニメーションツールを使用できます。

Animation Rigging

Animation Rigging パッケージは、アニメーション化されているスケルトンに対して実行時にプロシージャルなモーションを設定するために使用します。このパッケージでは、キャラクターの制御リグ階層を手動で構築するための一連の定義済みのアニメーションのコンストレイントがあります。また、ここで C# で独自のコンストレイントを構築することもできます。これにより、ワールドとのインタラクション、スケルトンデフォーメーションリグ、物理演算ベースの二次モーションなど、ゲームプレイ中に魅力的なアクションを作成することができます。

Animation Rigging には他にも、容易にアクセスできないアニメーションを変更したり、元のアニメーションでは考慮されていなかった新たな状況に合わせてアニメーションを適応させられる、といった利点があります。

Unity の Animation Rigging パッケージを使用すれば、特定のボーンのアニメーションをオーバーライドするリグを作成したり、アニメーション化されたオブジェクトにプロシージャルなモーションを追加するためのコンストレイントを設定したりできます。動的なアニメーションを作成できるだけでなく、アニメーションツールを使用してアニメーションクリップを変更または作成したり、Timeline でシネマティックシーケンスを作成したりすることも可能です。



Animation Rigging の Bone Renderer 機能

その他のリソース

アニメーションクリップの操作
Animation Rigging でワークフローを改善する
リグ間のアニメーションの再利用およびリターゲティング

カットシーンと
シネマティクス

この HDRP テンプレートには、動画、TV 番組、その他のアニメーション化されたリニアコンテンツを作成および編集するためのプロ向けツールをショートアニメで紹介するチュートリアルが含まれています。

事前レンダリングされたインタースティシャルの時代は終わりました。Unity では、かつてはオフラインレンダリングが必要だった機能を多数使用したカットシーンと没入感のあるシネマティクスを作成できます。プロジェクトで独自の映像、トレーラー、カットシーンを作成するのに必要なすべての機能を入手するには、**Cinematic Studio 機能セット**をパッケージマネージャーからダウンロードするか、Unity 2021 LTS 以降で Unity Hub から新しいプロジェクトを開始し、「**Cinematic Studio Sample**」を選択してください。

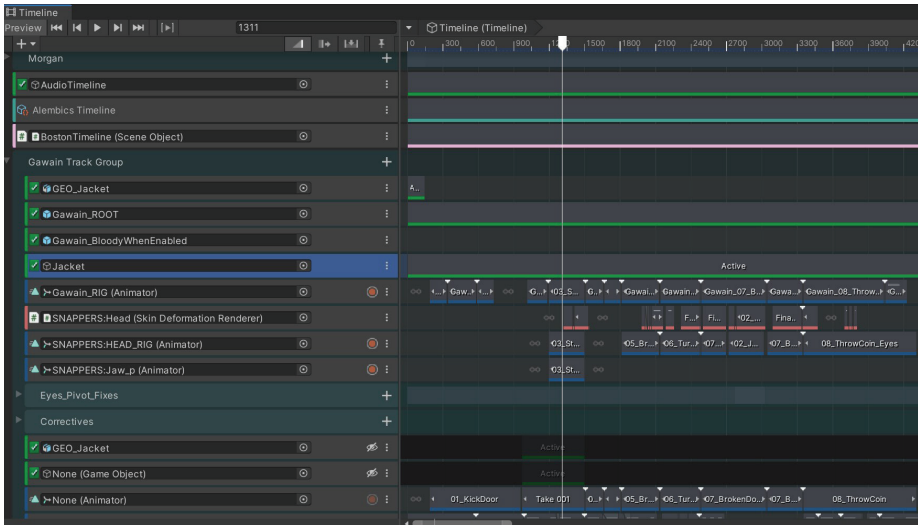
Timeline

Timeline は、Unity のバーチャル映像製作ツールの基盤です。Timeline のインターフェースでは、動作、アニメーション、オブジェクトの有効化 / 無効化など、シーンの要素を制御できます。ディレクターでもあり編集者でもある、自身の編集スキルを試すのに使用してみてください。

構成したシーンのリアルタイムビジュアライゼーションのおかげで、従来のアニメーションや VFX プロジェクトのようにアニマティクスや絵コンテに大きく頼る必要がありません。ノンリニアのビデオの編集のように、Timeline インターフェースのレイヤーはそれぞれが 1 つのトラックです。

複数のトラックを集めることで、オーディオ、ゲームプレイシーケンス、パーティクルエフェクトなどで構成されたシネマティックシーンを作成できます。昔さながらの Machinima（マシニマ）が、これまでにないほど劇場アニメ映画に近いものになってきています。

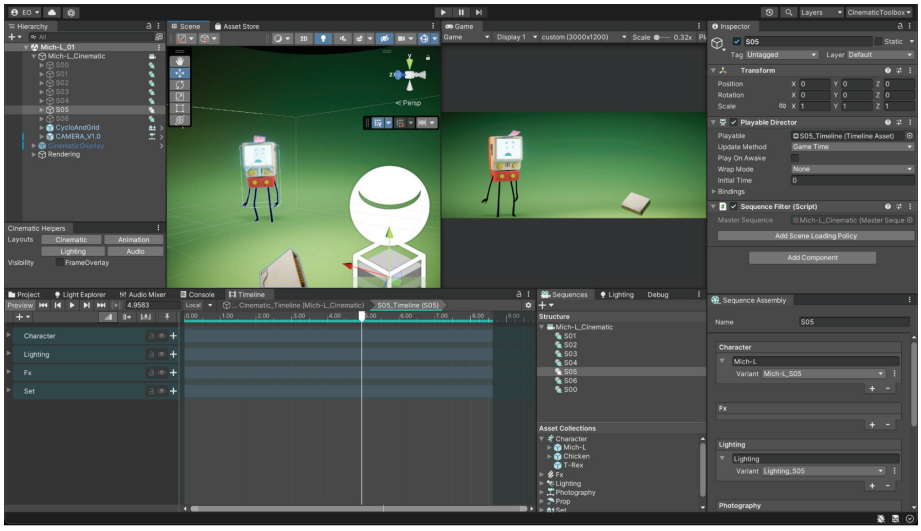
オブジェクト編集の自由度が上がるため、クリエイティブチームは、締め切り間際でさえも、非破壊的な方法で映像を大幅に変更したり再編集したりできるようになりました。ほとんどの他メディアを使用している場合では、これはまったく考えられないことです。



デモチームは API でカスタマイズした Timeline を使用し、Unity でデモフィルム『*The Heretic*』を編集しました。

Sequences

Sequences は、Unity 2021 LTS の新しいワークフローツールで、動画の編集コンテンツの整理で役立ちます。Sequences を使用してプロジェクトの編集構造とクリエイティブコンテンツを編成し、要素を整理することでオーサリングとコラボレーションが効率化します。また、非破壊的な方法でアイデアをテストできます。



「Sequences」ウィンドウに映像のアセットがまとめられており、ここで各シーケンスに必要なアセットを定義し、Timeline からクリップを編成して、プロジェクト構造全体を整理できます。

Unity で作成され、Unity Recorder で生成された短編アニメーション『*WiNDUP*』のスチール。

Unity Recorder

Unity Recorder は、レンダーパイプラインで作成されたオフラインのビデオ、トレーラー、カットシーン、映像をレンダリングします。これには、ビデオ、静止画、Arbitrary Output Variables（AOV、任意出力変数）、またはキャプチャ固有のレンダーパス（HDRP を使用）のいずれかを生成するレコーダータイプが若干含まれています。Unity 2021 LTS 以降のバージョンの Unity Recorder は、フレームの補間をサポートしており、ブロードキャスト品質のモーションブラーを維持し、パストレースされたフレームを記録できます。

Alembic のサポート

Alembic は、髪の毛やクロスなど、複雑なアニメーションの再生を最適化するのに使用されるファイル形式です。Alembic ファイル（.abc）はインポートおよびエクスポートでき、さらに Alembic 形式のアニメーションを直接 Unity で記録、ストリーミングできます。Alembic 形式は一般に、アプリケーション間でアニメーションをベイクされたデータの形で転送するために使用されます。

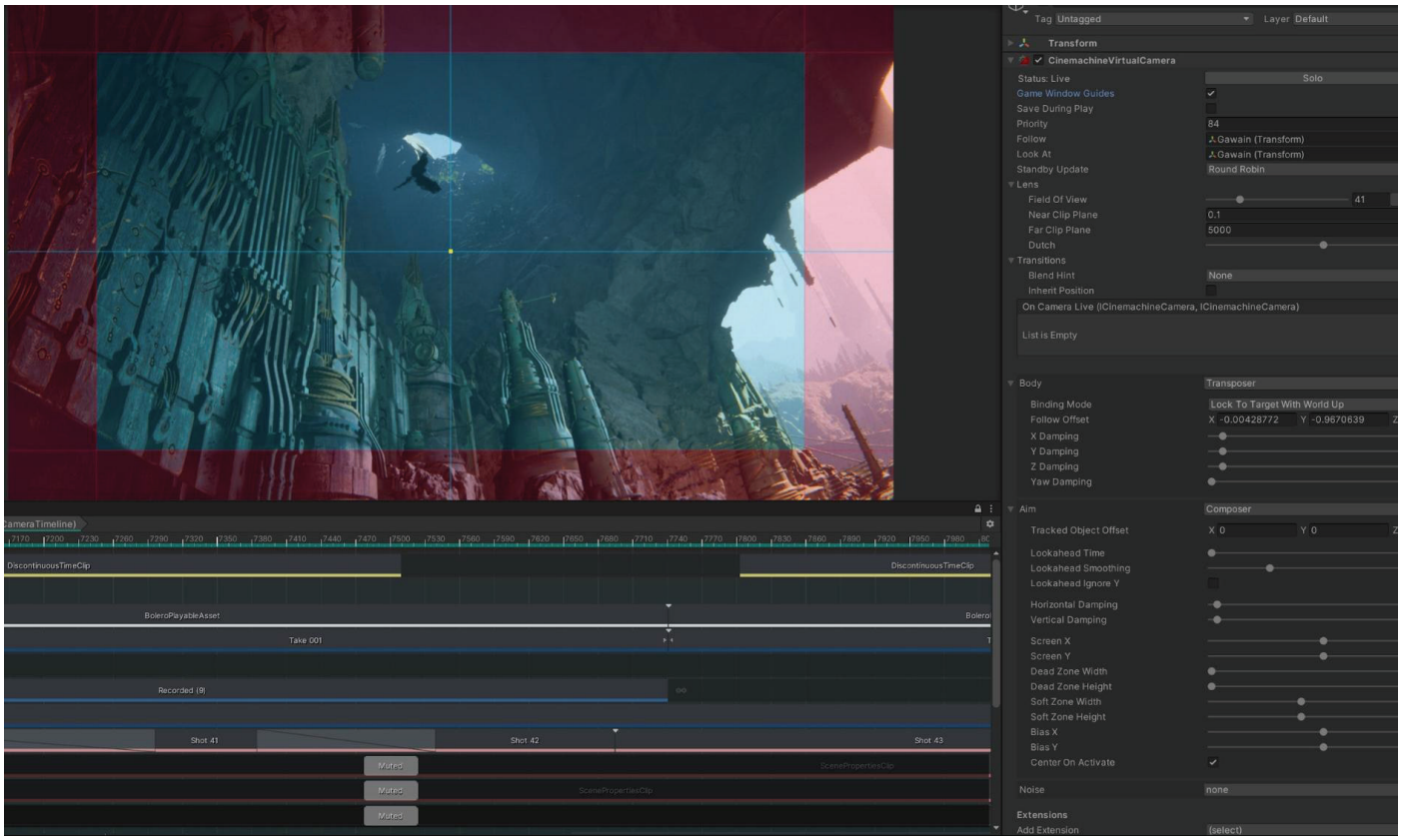
FBX Exporter

前のセクションで説明した FBX Exporter を使用すると、コンテンツを DCC にラウンドトリップできます。VFX アーティストや 3D アーティストは Unity で生成されたジオメトリやアニメーションに対して反復作業でき、最終的なアセットを簡単に Unity に戻すことができます。

Cinemachine

Cinemachine はバーチャルカメラオペレーター、つまりインテント駆動型のカメラシステムです。ハリウッドのカメラマンが撮影現場で行うのと同じような方法で、ゲームカメラをドリー、トラック、ズームできます。

Cinemachine を使用すると、コーディングを行うことなく対象物をフレーミングし、追従することができます。複雑なロジックはすべて Cinemachine が処理します。特別なカメラリグにモジュールセットを取り付け、いくつかパラメーターを入力するだけで、後は Cinemachine に任せることができます。



Timeline と Cinemachine でシネマティクスをデザイン

独自のカメラシステムをすでに作成済みの場合でも、Cinemachine はカスタムカメラソリューションと一緒に使用することができます。

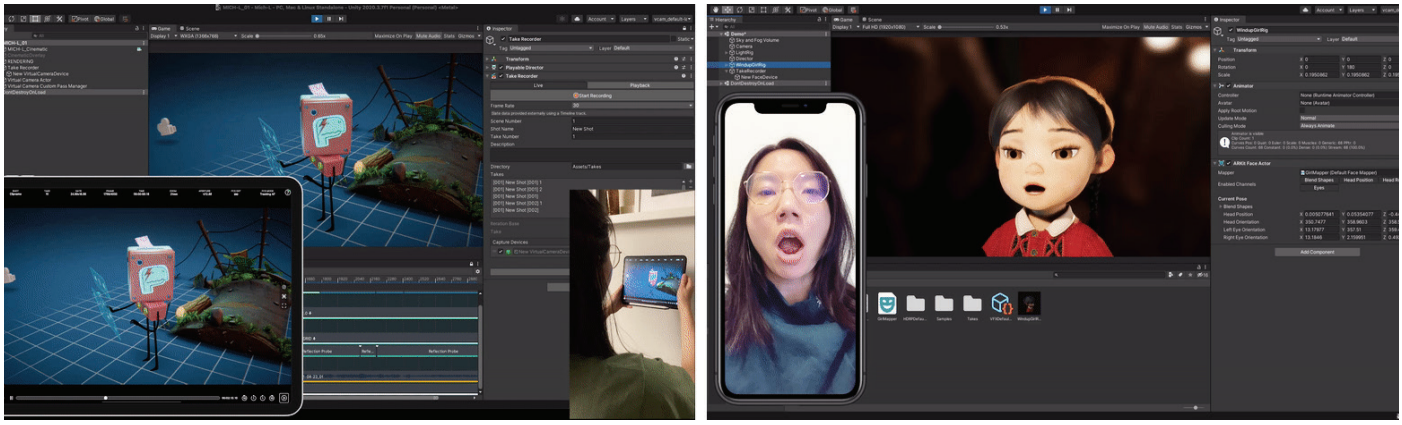
Cinemachine は、ゲーム内とカットシーンカメラアニメーションのどちらでも効率的です。一人称シューティング（FPS）、三人称視点、横スクロール、見下ろし型、リアルタイムストラテジー（RTS）など、あらゆるゲームジャンルに使用できます。

Live Capture

Live Capture パッケージは、拡張現実（AR）対応の iPad や iPhone のパワーを活用し、現実世界の身体表現を Unity へ取り込むのに便利です。このパッケージを使用して、以下の 2 つのコンパニオンアプリに接続します。

— **Unity Virtual Camera：**モバイルデバイスで Unity カメラを直感的に操作し、モーションを記録するためのアプリケーション。自然なハンドヘルドショットの撮影に最適です。

— **Unity Face Capture：**顔の表情をアニメーション化するためのアプリケーション。この情報をキャプチャしてキャラクターに適用することで、アニメーターはキャラクターの顔のアニメーション化にかかる時間を大幅に短縮できます。

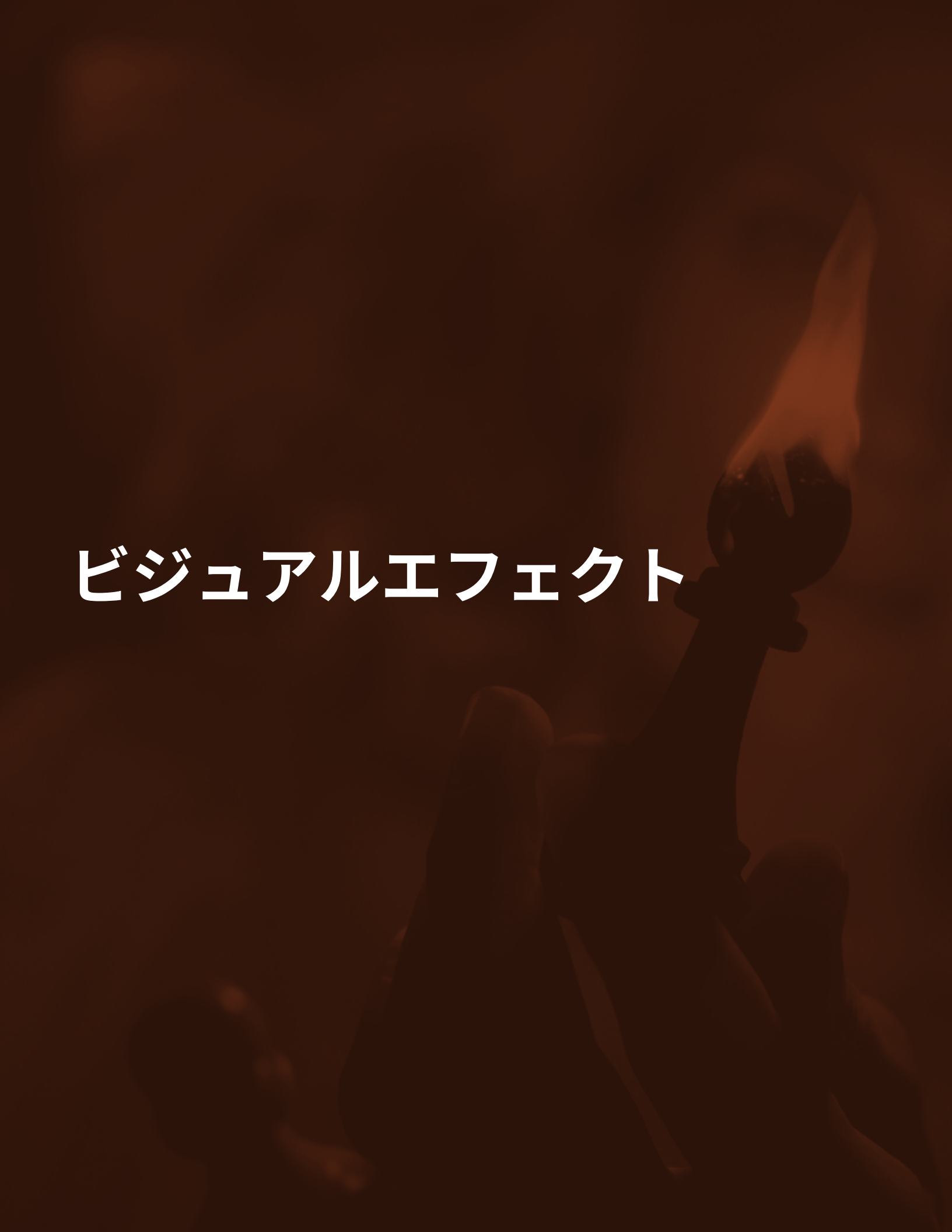


Unity Virtual Camera（左）と Unity Face Capture（右）の使用例

他の映像制作者が Unity でどのような作品を制作しているか、映像制作、アニメーション、シネマティクスページでご覧ください。

その他のリソース

イッツショータイム！映画的表現を追究するクリエイターのための新しいツールのご紹介
Unity のバーチャルプロダクションツールでゲームをレベルアップさせる方法
Timeline を利用して、ゲームプレイとストーリーテリングをブレンドする：カットシーンとゲームグラフィックス
リアルタイムテクノロジーでストーリーテラーに力を：パート 1 およびパート 2
Unity ブログのエンターテインメント記事


ビジュアルエフェクト

Unity が現在提供しているパーティクルシステム作成ツールは、**ビルトインのパーティクルシステム**と **Visual Effect Graph** の 2 つです。2021 LTS 以降では、どちらのシステムもビルトインレンダーパイプライン、URP、および HDRP と互換性があります。

ビルトインのパーティクルシステム

ビルトインのパーティクルシステムでは、システムとパーティクルに対して C# スクリプトから完全な読み取り / 書き込みアクセスが可能です。**パーティクルシステム API** を使用してパーティクルシステムのカスタム動作を作成できます。

インスペクターで変更可能な Particle System コンポーネントには強力なプロパティセットが用意されており、使いやすいようにモジュールに編成されています。これらのモジュールでは、パーティクルの放出レートと継続時間、エミッターの形状に加えて、パーティクルの生存期間中の外観、動き、動作を定義できます。

さらに、ビルトインのパーティクルシステムには、**Renderer モジュール**が含まれています。このモジュールではモジュールの設定にアクセスして、パーティクルのクアッドやメッシュの変形、シェーディング、他のパーティクルによるオーバードローの方法を決定できます。さらにパーティクルの軌跡をレンダリングすることもできます。

Lights モジュールを使用すると、パーティクルエフェクトに簡単にリアルタイムライティングを追加できます。これは、例えば火、花火、雷などで、パーティクルの周囲にライトを放つのに使用できます。このライトには、アタッチ先のパーティクルからさまざまなプロパティを継承させることができます。

各パーティクルについての情報をカスタムシェーダーまたは C# スクリプトに渡すには、頂点データストリームを有効にします。主な機能には、パーティクルシステムのパーティクルに力を適用する **Force Field** コンポーネントや、高パフォーマンスの C# 動作を記述するための **C# Job System** の統合があります。

ビルトインのパーティクルシステムには、完全なマルチプラットフォームサポート、ビルトインレンダーパイプラインのサポート、Unity の物理演算システムとの統合、パーティクルライトのサポート、C# スクリプトを介した個々のパーティクルデータへのアクセス、そしてビルトインパーティクルを含むプレハブが多数あるプロジェクトでの強力なスケーラビリティ、などの利点があります。

ビルトインのパーティクルシステムで作成されたビジュアルエフェクト

Visual Effect Graph

Visual Effect Graph を使用すると、ノードベースのビジュアルロジックを使用した効果のオーサリングが可能になります。シンプルな効果にも複雑なシミュレーションにも使用できます。Unity は Visual Effect Graph を、**Visual Effect コンポーネント**に対して使用できる **Visual Effect アセット**に格納しています。実際、Visual Effect アセットはシーンで複数回使用できます。

Visual Effect Graph は、GPU でパーティクルの動作のシミュレーションを行うため、ビルトインのパーティクルシステムより多くのパーティクルのシミュレーションを実行できます。ただし、Visual Effect Graph のシミュレーションは GPU で行われるため、計算機能を備えたプラットフォームでのみ機能します。

Visual Effect Graph のメリットには、シミュレーションがより高速になりパーティクル数を増やせる、カスタマイズ可能な動作、拡張性（サブグラフ、テンプレートの作成など）、カメラバッファアクセス（HDRP の場合）、およびネイティブシェーダーグラフの統合などが挙げられます。Visual Effect Graph をターゲットにするには、シェーダーグラフで作成したカスタムシェーダーを使用します。これらのシェーダーは、HDRP の髪の毛と布地のような新しいライティングモデルを使用できます。また、頂点レベルでパーティクルを変更して、鳥の羽ばたきや、石けんの泡のように不安定に揺れるパーティクルなどの効果を作成できます。

Visual Effect Graph で作成されたビジュアルエフェクト

Visual Effect Graph のコンポーネント

Visual Effect Graph にはシェーダーグラフとの類似点が多数あります。Visual Effect Graph の基本コンポーネントであるコンテキスト、ブロック、ノード、およびプロパティについて以下に簡単に説明します。

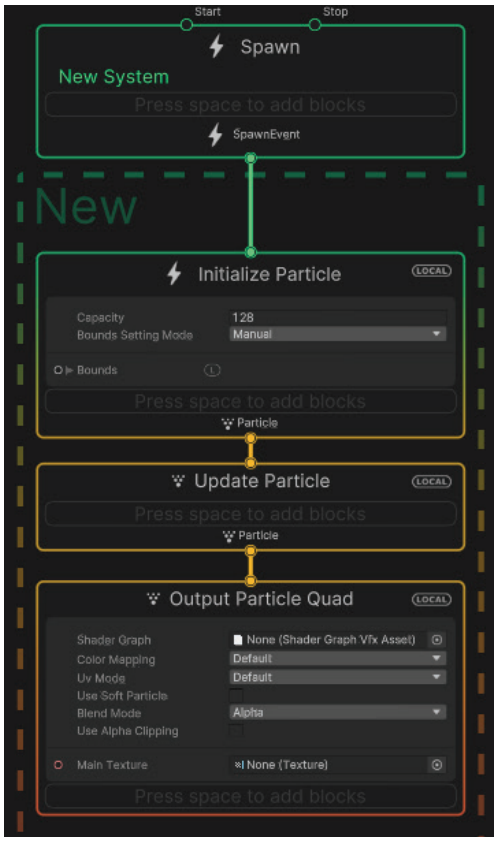
コンテキストは、処理されるパーティクルの演算の順序を表します。

— **スポーン：**パーティクルのスポーンを制御

— **初期化：**初期パーティクル値を設定

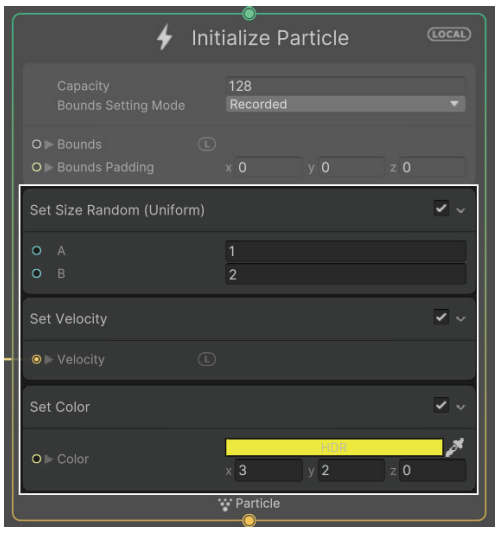
— **更新：**時間の経過に伴うパーティクルの動作を制御

— **出力：**結果のパーティクルを画面にどのようにレンダリングするかを定義



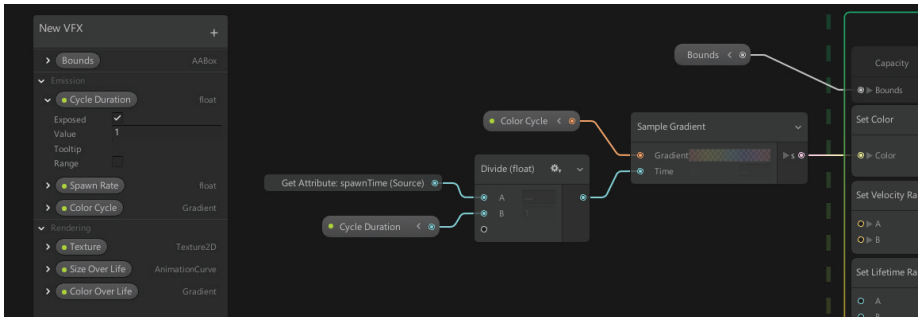
VFX の初期化から出力までのノードの流れ

ブロックは、各コンテキストに追加される操作です（C# の関数呼び出しに似ています）。ブロックを追加するには、コンテキスト内でスペースバーを押すか、右クリックして「**Create Block**」を選択します。



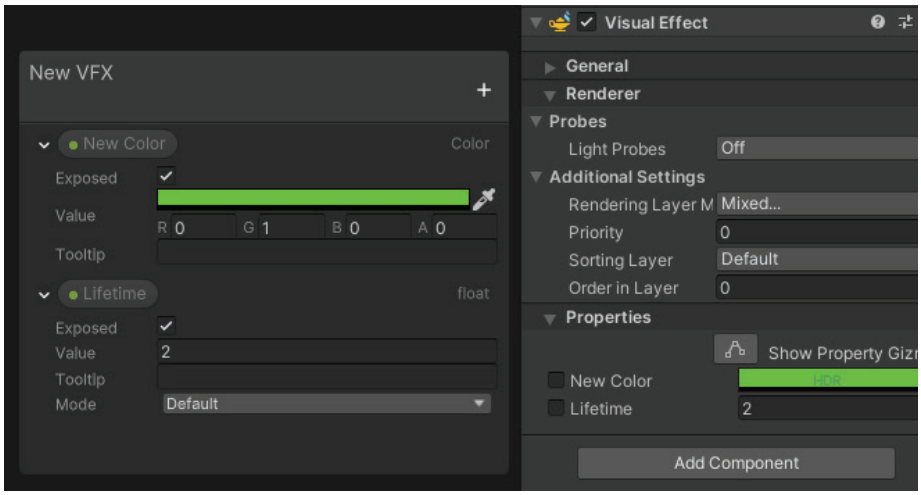
パーティクルグループの開始プロパティ

ノードは、リンクされた個々のオペレーションを行って大きな計算を実行します。シェーダーグラフでのノードの使用に似ています。以下に示すように、これらを変数の代わりに使用して、ブロックのプロパティを駆動できます。



ブラックボードにはグローバル変数が表示されており、インスペクターに表示するパラメーターを設定できます。

プロパティは、ブラックボードに追加し、グラフ内で使用できるデータを表します。プロパティを公開するよう設定すると、インスペクターまたは C# スクリプトを使用して Visual Effect Graph エディター外で変更できます。



インスペクターに公開されたブラックボードのプロパティ

詳細については、Visual Effect Graph マニュアルを参照してください。

ビルトインレンダーパイプラインで作成した Unity のデモ『*Neon Challenge*』のポストプロセスエフェクト

ポストプロセス

Unity は、わずかな設定時間でアプリケーションの外観を大幅に向上させることができる、さまざまなポストプロセスエフェクトを提供しています。これらの効果を使用して物理カメラとフィルムのプロパティのシミュレーションを行ったり、スタイリッシュな外観を作成したりできます。

よく使用される効果には、**被写界深度**、**ビネット**、**トーンマッピング**（カスタム LUT を含む）、**シャドウ / 中間調 / ハイライト**、**スプリットトーン処理**、および**色収差**があり、加えてコントラストと彩度の一般的な色調整もよく使用されます。

HDRP 機能の中でも特にこれらのポストプロセスエフェクトについて学習したい場合は、こちらの Unite Now セッションをご覧ください。

ポストプロセス向けのレンダーパイプラインソリューション

ビルトインレンダーパイプラインには、デフォルトでポストプロセスソリューションは含まれておりません。ビルトインレンダーパイプラインでポストプロセスエフェクトを使用するには、**ポストプロセスパッケージ**をダウンロードします。

一方、URP と HDRP にはそれぞれ独自のポストプロセスソリューションが含まれています。追加のパッケージは不要です。スクリプタブルレンダーパイプラインの実装は **Volume** コンポーネントを使用します。他の Volume オーバーライドを追加するのと同様の手法で、カメラにもポストプロセスエフェクトを追加できます。Unity プログラミングの経験がある方は、このセクションは読まずとも問題ありません。

Unity で高度なビジュアルエフェクトを制作するための最も信頼のおけるガイド

この e-book は、アーティストとテクニカルアーティストが同じように Visual Effect Graph を使用して、リアルタイムのビジュアルエフェクトを作成できるようにします。Visual Effect Graph の詳細なチュートリアルを通じて、よく使われる効果を作る方法を説明するだけでなく、パッケージに Unity およびサードパーティの他のツールを統合する方法をご案内し、主なワークフローの例となるサンプルを紹介しています。

e ブックを入手する

その他のリソース

Visual Effect Graph のデモ『*Spaceship*』
Visual Effect Graph で美しく複雑な効果を作成する
『*The Heretic*』のメイキング：VFX で出来たキャラクター「Morgan」
パーティクルシステムを開始する Post-Processing Stack の概要

UNITY での スクリプティング

Unity の Visual Scripting ソリューション

ノート：Unity プログラミングの経験がある方は、このセクションは読まずとも問題ありません。

Unity は、ゲーム開発で最も一般的なユースケースに対応する多数のツールを提供しています。しかし、テクニカルアーティストは、チームの作業効率を上げるのに役立つカスタムツールが必要な状況に遭遇することがあります。Unity でのコーディングのしくみと**スクリプティング API** の使用方法を理解することで、手続き型システムやデザイナー向けの Visual Scripting ツールの作成から、高度にカスタマイズされたアセットインポートワークフローまで、制作のどんな局面でも効率化していける柔軟性を持つことができます。

Unity でスクリプトを作成したいけれどもコーディングの経験がない方は、この簡単な概要を読み、下に挙げた初心者向けリソースで学習することをお勧めします。

Unity は業界標準の言語である C# をサポートしています。C# は Java や C++ と似ているところがあります。Unity で開発されたすべてのゲームプレイとインタラクティビティは、ゲームオブジェクト、コンポーネント、変数の 3 つの基礎となるビルディングブロックに基づいています。

キャラクター、ライト、特殊効果、小道具など、Unity ゲームのあらゆるオブジェクトはゲームオブジェクトです。ゲームオブジェクトはそれ単独では何もできません。アニメーションにするには、ゲームオブジェクトにコンポーネントを追加してプロパティを指定する必要があります。

コンポーネントは、それがアタッチされているゲームオブジェクトの動作を定義して制御します。簡単な例を挙げるなら、ライトの作成でしょう。ライトを作成する際には、ゲームオブジェクトに Light コンポーネントをアタッチします。また、Rigidbody コンポーネントを追加すると、オブジェクトが落下するようになります。

コンポーネントには、Unity エディターの「Inspector」ウィンドウまたは C# スクリプトを介して調整できる、数多くの編集可能なプロパティ（変数）があります。上記の例では、ライトのプロパティとして範囲、色、輝度があります。

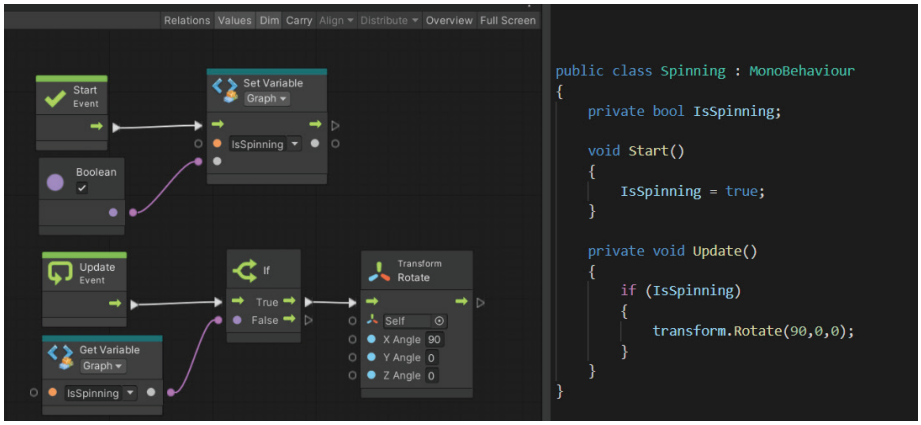
Unity Learn では、Unity で C# スクリプトを作成する方法を学習するための優れた無料のチュートリアルとコースを提供しています。以下はお勧めする 3 つの学習パスです。

— **短いスクリプトチュートリアル：**初心者および中級者向けのガイドプロジェクトから開始します。

— **コーディングのクリエイターキット：**このキットを数時間かけて完成させ、アクションロールプレイングゲーム（RPG）のコンテキストで C# コードの基本を学習します。

— **Create with Code：**37 時間を超える指導が受けられる包括的なコースを受講します。

Unity の Visual Scripting



Unity の Visual Scripting ソリューション

Unity の Visual Scripting システムを使用すると、従来のコードを記述することなく、Unity プロジェクトのゲームロジックを作成することができます。Unity 2021 LTS 以降では、Visual Scripting がエディターからパッケージとしてデフォルトでインストールされます。2019 LTS や 2020 LTS など、これより前のバージョンでは Unity Asset Store からインストールする必要があります。

Visual Scripting には、視覚的に理解できるノードベースのグラフが備わっており、プログラマーでもそれ以外でも最終的なロジックを設計したり、簡単なプロトタイプを作成したりできます。スクリプトに関するコラボレーションが容易になり、プロジェクトで既存の C# スクリプトと並行して使用することもできます。さらに、グラフベースのロジックがどのように実行されるかをリアルタイムで監視できます。

Visual Scripting には、ゲームを構築する際に使用される以下の 4 つの基本概念があります。

- **ノード：**Visual Scripting との相互作用を作成する基本のビルディングブロックです。イベントのリッスン、変数の値の決定、ゲームオブジェクトのコンポーネントの変更など、さまざまな処理を行います。ノード同士をつないで、ビジュアルスクリプトのデータまたはロジックを定義できます。
- **グラフ：**グラフはアプリケーションのロジックを作成するノードのセットを視覚的に表します。グラフには、**スクリプトグラフ**と**ステートグラフ**の 2 種類があります。スクリプトグラフは、特定の時間に特定の順序で実行する動作を定義するのに使用します。これに対し、ステートグラフには、ステート（ユーザー定義の動作のセット）と特定のユーザー定義パラメーターが満たされた場合のステート間の遷移が含まれます。
- **スクリプトマシンとステートマシン：**これらはアプリケーションでスクリプトグラフまたはステートグラフを使用できるようにするコンポーネントです。スクリプトグラフまたはステートグラフは、これらのコンポーネントのいずれかを使用する、シーン内のゲームオブジェクトにアタッチしない限り使用できません。
- **変数：**値とデータを格納するコンテナです。これにより、グラフ内のノードは値やデータを参照できるようになります。変数内の値はランタイム時に変更できます。

Visual Scripting は、プログラマーと TA が、アーティストやデザイナーとより効率的にコラボレーションするための拡張機能、テンプレート、ツールを作成するためのソリューションになります。これにより、C# の知識があるかどうかにかかわらず、プロジェクトのすべてのメンバーが効率化されたプロセスで連携できます。

Unity ゲームデザイナープレイブック

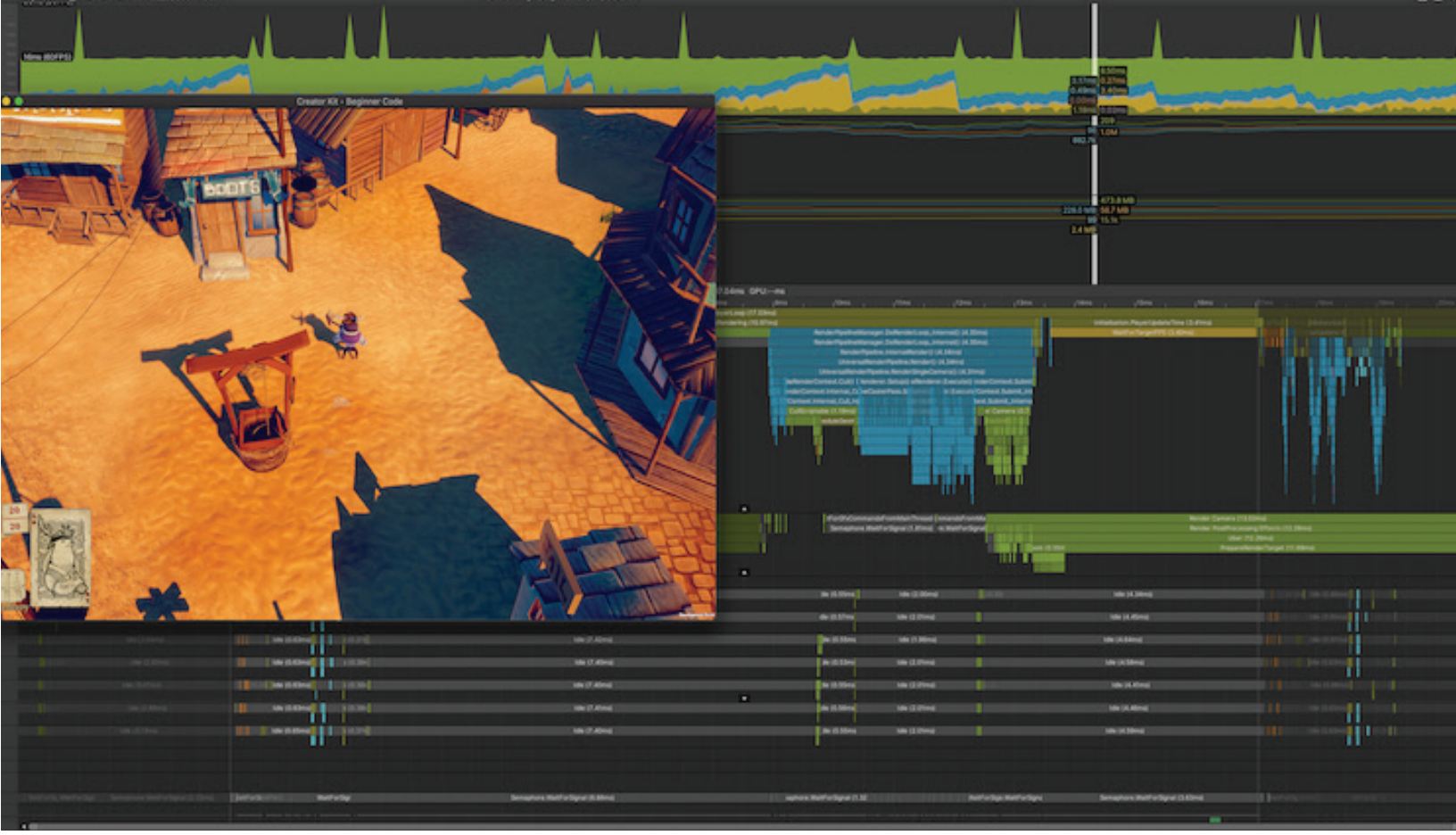
このガイドは、デザイナーが Unity でゲームプレイのプロトタイプを作成し、制作、改良するのに不可欠な基本ツールについて説明します。C# でのスクリプティング、Unity Visual Scripting の詳しい概要を把握し、入力コントロール、キャラクターコントローラー、グレーボックス化などを使用して最小限のコーディングでゲームデザインタスクを達成する方法を学習できます。

e ブックをダウンロード

その他のリソース

Unity Visual Scripting の詳細
Visual Scripting 入門チュートリアル

プロファイリング とデバッグ



エディター内で実行中のゲームの統計情報を表示するプロファイラーの概要

プロファイラー

Unity Profiler（プロファイラー）はアプリケーションのパフォーマンス情報を取得するのに使用できるツールです。このツールをネットワーク上のデバイスに接続することで、目的のリリースプラットフォーム上でアプリケーションがどのように実行されるかを実際に見てみることができます。これをエディターで実行してアプリケーションの開発中にリソースの割り当てについての概要を確認することもできます。

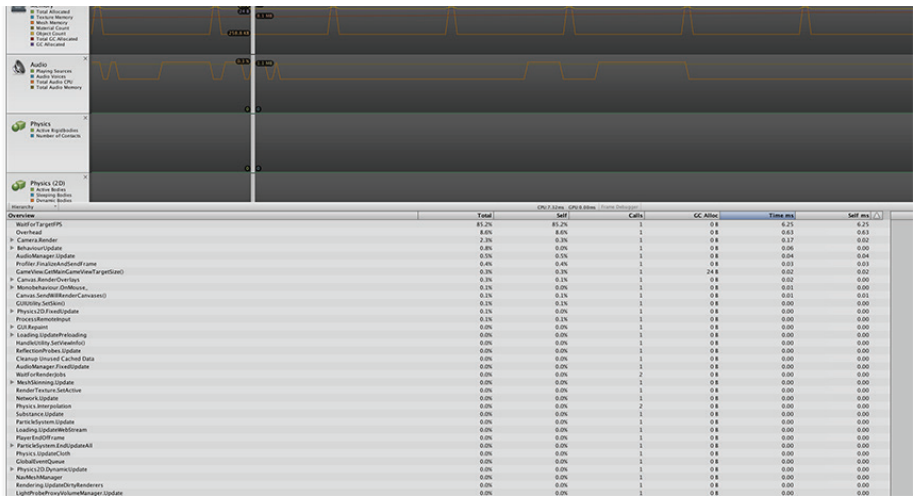
プロファイラーは、対象となるアプリケーションのパフォーマンス情報を収集して表示します。これには、CPU、メモリ、レンダリング、オーディオなどに関する情報が含まれています。プロファイラーはアプリケーションのパフォーマンス改善の余地を特定するのに有用なツールです。コード、アセット、シーン、設定、カメラ、レンダリング、およびビルドがアプリケーションの全体的なパフォーマンスにどのように影響しているかを突き止めることができます。プロファイラーには、一連のグラフとして結果が表示されるため、アプリケーションのパフォーマンスにスパイクが生じた場所が明確にわかります。

ビルトインプロファイラーを使用するだけでなく、低レベルネイティブプラグインの **Profiler API** を使用してプロファイリングデータをサードパーティ製のプロファイリングツールにエクスポートすることや、**Profiling Core パッケージ**を使用してプロファイリング分析をカスタマイズすることもできます。また、**Memory Profiler** や **Profile Analyzer** などの強力なプロファイリングツールをプロジェクトに追加すると、パフォーマンスデータをより詳細に分析できます。

「Profiler」ウィンドウにアクセスするには、**「Window」>「Analysis」>「Profiler」**の順に移動します。「Profiler」ウィンドウの左側には、**Profiler モジュール**の列が表示されます。各モジュールには、コンテンツの特定の側面に関する情報が表示されます。CPU 使用率、GPU 使用率、レンダリング、メモリ使用率、オーディオ、物理演算、ネットワーキングの個別のモジュールがあります。

各種システム間を移動して最適化すべき領域を特定

「Profiler」ウィンドウの下半分には、選択されているモジュールから得られる、選択されているデータフレームの詳細情報が表示されます。



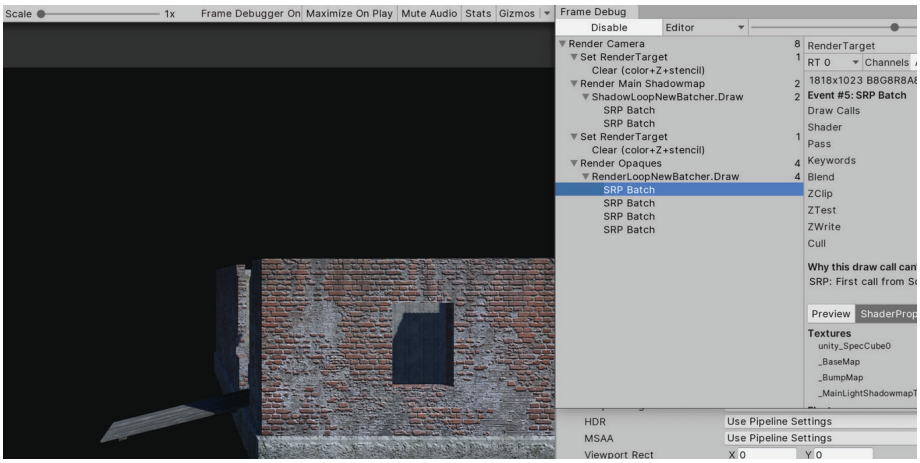
タスクの一覧とその処理に必要なフレーム単位の時間

上の画像に示されているデータのタイプは、現在選択されているプロファイラーによって異なります。例えば、**メモリ使用率のプロファイラー**を選択した場合は、最もメモリを消費しているアセットとメモリの合計使用量に関する情報が表示されます。一方で**レンダリングのプロファイラー**を選択した場合は、レンダリング中のオブジェクトの数または実行中のレンダリング処理の数に関する統計情報が表示されます。

フレームデバッガー

フレームデバッガーは、実行中のゲームの再生を特定のフレームで停止できる便利なツールです。停止することで、そのフレームのレンダリングに使用されている個々のドローコールを確認できます。またこのデバッガーでは、フレームを 1 つずつ移動して、シーンがどのように構築されているのかを詳しく調べることができます。そのため、特定のシーンが原因でフレームレートの問題が発生している場合に、プロジェクトを効果的にデバッグできます。

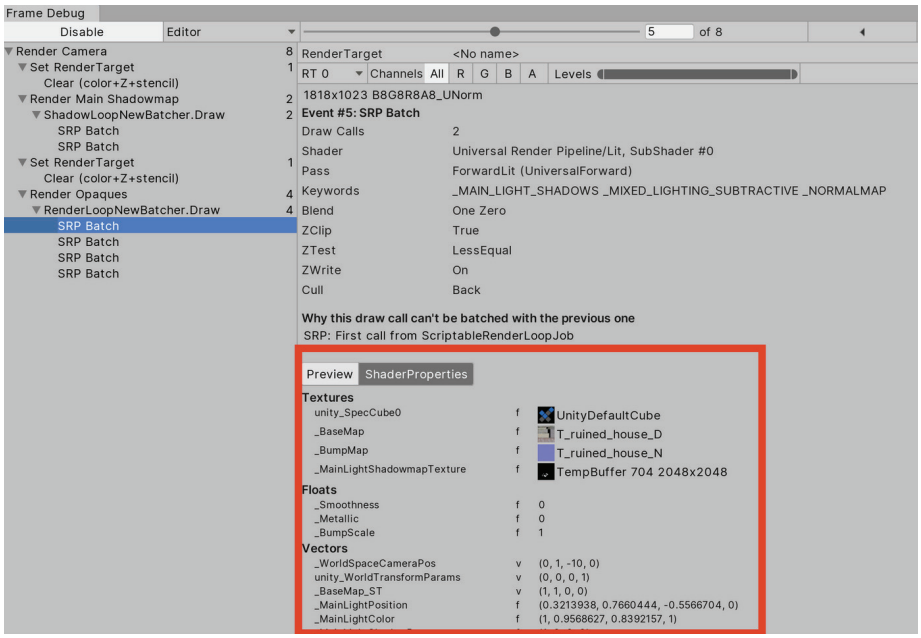
このデバッガーは、**「Window」>「Analysis」**ツール内にあります。デバッガーを有効にするとゲームビューが動かなくなり、その停止したフレームで画面上に発生している各種のドローコールを観察できます。例えば、ウィンドウの左側にあるいずれかの Draw Mesh 呼び出しをクリックすると、その Draw Mesh によって実際にレンダリングされたオブジェクトで「Game」ウィンドウが更新されます。



フレームデバッガー内のレンダーパス

「**Frame Debug**」ウィンドウは、ユーザーが探している情報を効率よく見つけられるよう、複数のセクションに分けられています。左側には、一連のドローコールやその他のイベント（ポストプロセスエフェクトなど）が表示されます。ウィンドウの右側には、選択されているドローコールに関するさらに詳しい情報（ジオメトリの詳細や、レンダリングに使用されるシェーダーなど）が表示されます。

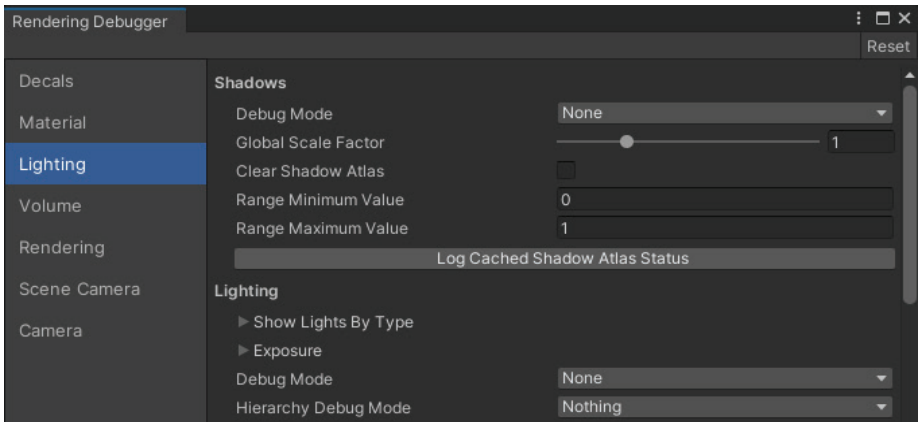
「**ShaderProperties**」には、使用されたシェーダーステージも表示されます。これは、選択されたシェーダーの現在の状態と使用されているプロパティを把握するのに便利です。これで描画プロセス中にシェーダーが適切に機能しているかどうかを確認できます。



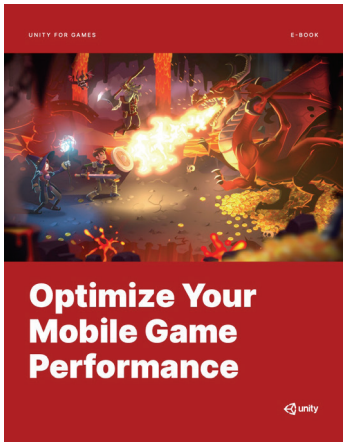
フレームデバッガーの「ShaderProperties」ビュー

Rendering Debugger

Unity 2021 LTS 以降では、URP プロジェクトと HDRP プロジェクトの両方に Rendering Debugger を使用できます。Rendering Debugger では、ライティング、レンダリング、マテリアルの各プロパティをより詳細に可視化できます。そのように高いレベルで可視化することで、レンダリングの問題を特定でき、シーンやレンダリング設定を最適化できます。各種デバッグビューの詳細については、URP と HDRP のドキュメントをご覧ください。



開発ビルドでは、Rendering Debugger を再生モード中にエディターで利用できるほか、任意のデバイスの Unity Player で実行時に利用することもできます。

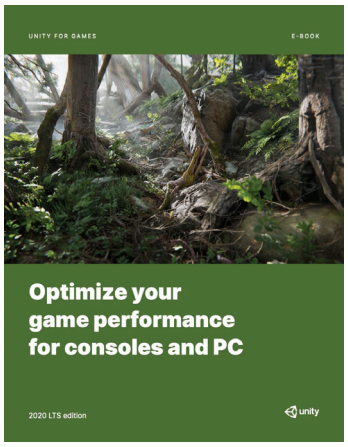


ゲームパフォーマンスの最適化

サポートエンジニアで構成された Unity のエキスパートチームから 75 個の実用的なヒントとベストプラクティスを集めた最も包括的なガイド。洗練された体験をプレイヤーへ提供しつつ、可能な限り多くのデバイスでスムーズに動作するようにモバイルゲームを最適化する方法を学べます。

大手ゲームスタジオが参加した実際のプロジェクトから得られた詳細なアドバイスが、ここに集約されています。Unity を最大限に活用してモバイルゲームのパフォーマンスを向上させるのに役立ちます。

e ブックをダウンロード



コンソールや PC を対象とする開発者向けにも、最適化に関する優れたヒントが数多く提供されています。Unity のエキスパートエンジニアチームから集めた、PC とコンソール向けの最適化に重点を置いた 80 個の実用的なベストプラクティスをこちらの e ブックをご確認ください。

e ブックをダウンロード

その他のリソース

ゲームをプロファイルして最適化する方法
Unity におけるプロファイリングの概要

2D ゲーム開発

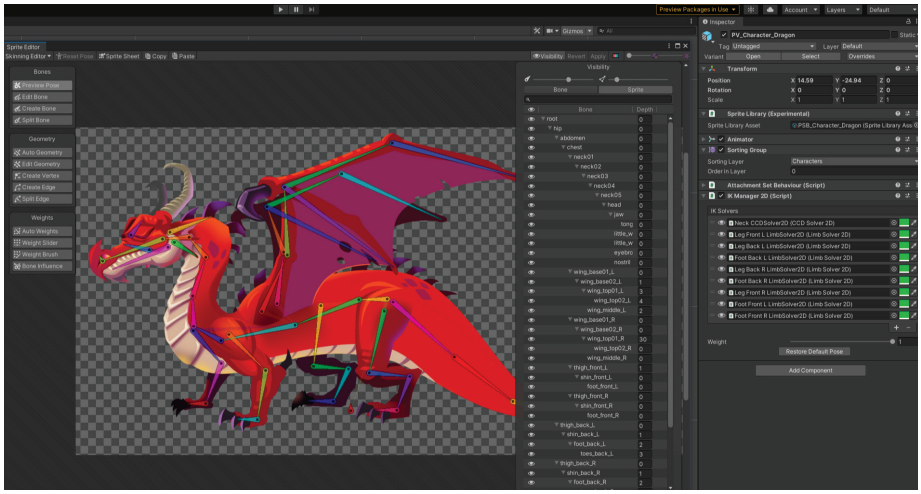
Unity のデモ『*Dragon Crashers*』は、Asset Store から入手できます。

RPG からマッチ 3 ゲームまで、近年最も成功を収めた売上上位のゲームには 2D ゲームが入っています。Unity での 2D と 3D の開発ワークフローは似ているので、2D 制作と 3D 制作を交互に行うことの多いモバイルスタジオにとって Unity は特に有利な環境と言えます。一通り揃っている Unity の 2D ツールスイートを活用することで、開発チームに柔軟性とスケーラビリティが備わり、複数のプラットフォームにわたってあらゆる種類の 2D ゲームや 2D 体験を制作できるようになります。

Unity のネイティブ 2D ツール

Unity には、優れたグラフィックスだけでなく、スプライトのサポート、インバースキネマティクス（IK）による 2D スケルタルアニメーション、タイルマップベースのグリッドや有機的形状によるワールド構築、2D 物理演算、複数のスプライトをまとめてテクスチャにするスプライトアトラスツールなど、2D ゲーム開発に必要な機能が揃っています。

これらの 2D ツールは、**ビルトインレンダラー**と URP に含まれている **2D Renderer** の両方と互換性があります。2D レンダラーとは、2D Lights、ポストプロセス、シェーダーグラフによるビジュアルシェーダー作成などのビジュアルエフェクトを可能にするレンダラーです。



『*Dragon Crashers*』の敵キャラクター、Dragon Boss のスケルトンとパーツ

2D Animation

2D Animation ツールでスプライトのリグを作成し、ボーンを設定することにより、滑らかなスケルタルアニメーションを制作します。このアニメーション機能と PSD Importer を併用すると、キャラクターのアートワークを Photoshop から Unity へ簡単にインポートできます。そうすることでアニメーションを有効化し、レイヤードアートワークをプロジェクトに直接組み込むことができます。さらに、これらのツールにはスワップ機能もあり、同じリグとアニメーションを再利用してキャラクターを作成できます。

ライトの情報を補助的なテクスチャ（法線マップとマスクマップ）に移動することで、より動的な臨場感のあるライティングエフェクトを 2D で作成できます。

2D グラフィックス：ライトとシェーダー

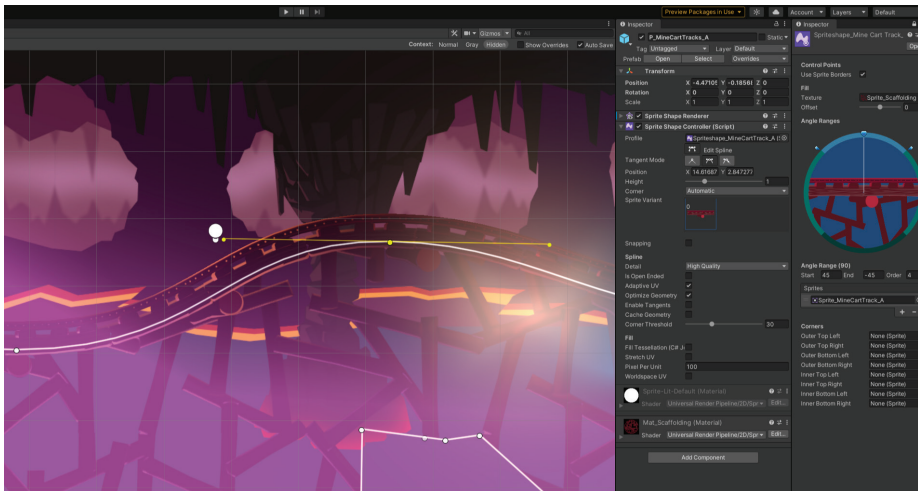
動的な **2D ライティングシステム**で、より臨場感のある 2D の外観や 2D ゲームプレイを実現できます。URP に含まれている 2D ライティングシステムは、アーティストにとって使いやすいツールとランタイムコンポーネントで構成されており、ライティングされた 2D シーンをすぐに作成できます。**Sprite Renderer** や、お馴染みの **3D Light** コンポーネントの 2D 版として機能する **2D Light** コンポーネントなどの、Unity のコアコンポーネントを介して動作します。

インスペクターで、ライトの色、輝度、減衰、ブレンディングエフェクトなどのパラメーターを簡単に適用できます。また、スプライトに法線マップを含めることで、2D Light のさらなる可能性を引き出すことができるほか、シェーダーグラフで視覚的に構築することにより、シェーダーを簡単に作成できます。

ワールド構築

タイルマップ

タイルマップシステムにより、サイズとパフォーマンスが最適化された巨大なグリッドベースのワールドを、**六角形**や**等角**で作成します。タイルマップのオーバーヘッドは個々のスプライトよりも少なく、**Tilemap API** と追加機能を使用することで、多岐にわたるクリエイティブな選択肢が得られます。必要に応じてカスタムブラシの作成、**タイルパレット**へのゲームオブジェクトの追加、さまざまなスプライト**ソートレイヤー**ロジックの適用を行えます。



Sprite Shape によるトラックのベジエカーブの変更

Sprite Shape

視覚的で直観的なワークフローを使用して、有機的な 2D 環境を作成します。ベクター描画ソフトウェアに似た 2D Sprite Shape は、形状のアウトラインに沿って**スプライトをタイリング**し、アウトラインの角度に基づいてスプライトのデフォームと切り替えを自動的に行います。**Collider 2D** コンポーネントを Sprite Shape にアタッチして Collider プロパティを有効化し、**Sprite Shape API** でスプラインのアンカーポイントを変更すれば、実行時に動く形状を作成し、ゲーム内でプレイヤーと相互作用させることができます。例えば、ゲームプレイ中に地形を作り出すことや、変形させることが可能です。

Sprite テクノロジー

Sprite エディターでは、2D プロジェクト用のアートアセットを簡単に設定できます。Sprite エディターを使用することで、スプライトの**ユニットあたりのピクセル数**（PPU）を設定してラウンドトリップの精度を高められるほか、スプライトをタイル状にする、スライスする、衝突を定義する、ピボットポイントを定義するなどの操作を行えます。

2D と 3D の外観の融合

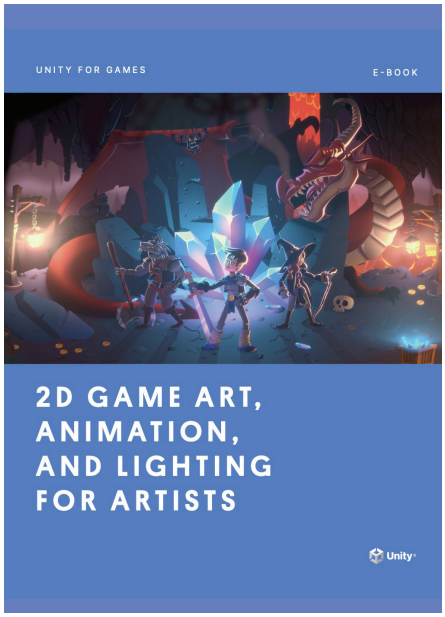
プロジェクトでビルトインレンダーパイプラインや URP を使用している場合は、比較的簡単に同じシーンで 2D と 3D の要素を融合させることができます。2D レンダリングでは、**ソートレイヤー**と**ソーティンググループ**という概念を使用してゲーム要素のレンダリングの順序を定義します。

Sorting Group コンポーネントを 3D ゲームオブジェクトに追加すると、同じゲーム内で 3D オブジェクトと 2D オブジェクトを統合できます。共通の **Physics system**（ゲームの制作手法に応じた 2D または 3D 物理演算）を使用してそれらのオブジェクトを相互作用させることができるほか、URP の **Camera Stacking** 機能を使用して 2D と 3D のライティングシステムを融合させることもできます。

ピクセルアートグラフィックス

ピクセルアートゲームは決して時代遅れになることはありません。**Pixel Perfect Camera** コンポーネントでは、望むままの低解像度かつ低忠実度の設定であったとしても、理想的な美観を実現することができます。それが古き良き様式美やドット絵の見た目などであっても可能です。

Pixel Perfect コンポーネントプロパティで一貫したピクセル解像度を設定できます。スプライトの回転や拡大縮小を行っても、補間なしでピクセルアートの鮮明さを保つ、アップスケーリングのような機能を含めることができます。これにより、設定内で指定されているピクセルグリッドに従ってスケーリングとカメラ移動を行えます。このツールでレトロな外観を再現することも、2D Light、シェーダー、ポストプロセスなどの最新グラフィックスと組み合わせて、現代的なピクセルアートグラフィックスを実現することもできます。



アーティストのための 2D ゲームアート、アニメーション、ライティング

2D ゲームは今、時代の追い風が来ています。ハードウェア、グラフィックス、ゲーム開発ソフトウェアの進化によって、2D ゲームでリアルタイムライトや高解像度のテクスチャを使えるようになり、スプライトの数の制限もほとんどなくなりました。

商用 2D ゲームを制作したい開発者とアーティスト向けに作成された、Unity の最も幅広くトピックを扱っている 2D 開発ガイドをぜひご覧ください。

e ブックをダウンロード

その他のリソース

2D アートのワークフローを加速させる方法
2D ゲームの制作に役立つヒント
レトロな 8 ビットおよび 16 ビットゲームの作り方
2D ライトで雰囲気を作り上げる方法

付録1：UNITY のデジタルヒューマン

この『*Enemies*』のデジタルキャラクターには Unity の新しいストランドベースのヘアソリューションとリアルタイムヘアレンダリングが使用されています。

ゲームにおける映画的ストーリーテリングの進化により、リアルな人間キャラクターを作成できるツールへの需要が高まっています。昨今の AAA タイトルにプレイヤーが期待しているのは、どの角度や距離からでもリアルに見える多面体のキャラクターです。そうした理由から、受賞歴のある Unity のデモチームは『*Enemies*』や『*The Heretic*』などの作品でデジタルヒューマンの作成ツールを活用し、それまで不可能だったことを実現してきました。

最新デモの『*Enemies*』は、Unity の SRP レンダリングアーキテクチャをベースに、高度なライティングと VFX を駆使して構築されています。HDRP とそれに組み込まれた Post-Processing Stack の最新の実装を使用し、同チームは、カメラの挙動を本物のカメラに極めて近い形で再現して、映画のような様相をすべてリアルタイムレンダリングで作り出すことに成功しました。『*Enemies*』と『*The Heretic*』に登場する人間のキャラクターは、クローズアップにも耐えられるよう高いレベルで細部を表現する必要があり、それがデモチームにとって克服すべき大きな課題でした。

『*The Heretic*』に登場するデジタルヒューマンの Gawain は、教育プロジェクトや非商用プロジェクトに利用できるよう Unity Asset Store で公開されています。

準備

Unity のデモチームは、微細な表情の動きも含め、本物の人間のあらゆる細部を再現する必要がありました。それを実現するためにデモチームが採用したのは、パフォーマンスキャプチャとキャプチャ後の細部の追加というハイブリッド手法でした。

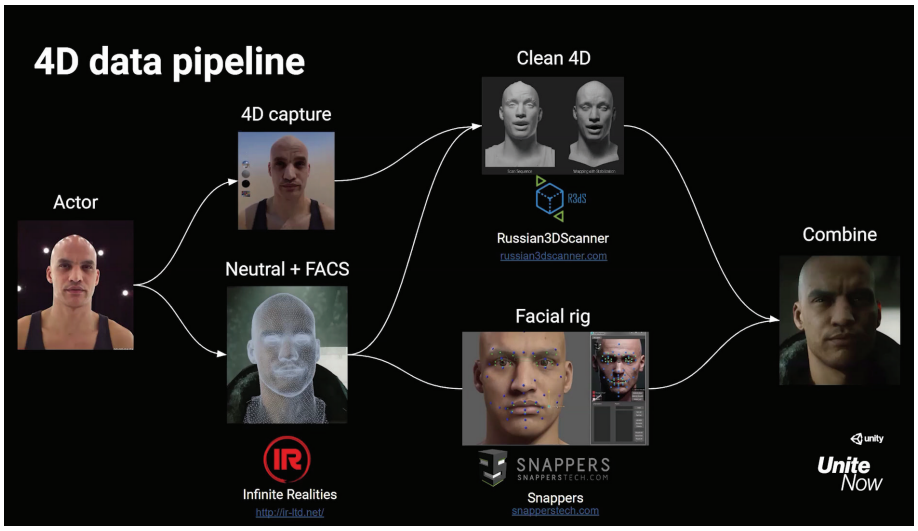
データ取得

『*Enemies*』と『*The Heretic*』での基本となるフェイシャルモーションキャプチャについては、Infinite-Realities スタジオでフォトグラメトリを使用し、役者の顔を毎秒何回も再構築しました。その結果、アニメーションの各フレームに対応した高解像度の 3D スキャンが「4D データ」として得られ、メッシュ、参照、テクスチャを介して役者の顔の動きの微妙な差異を再現するのに役立ちました。

『*The Heretic*』用に作成された役者の 4D スキャン

データ処理

4D スキャンでは役者の演技の微妙な差異を捉えられますが、ノイズが多かったり特定のデータポイントが失われたりする可能性もあります。それを軽減するには、データのクリーニング、メッシュの安定化、Wrap3D などのソリューションやサービスプロバイダーによる調整を適切に行う必要があります。



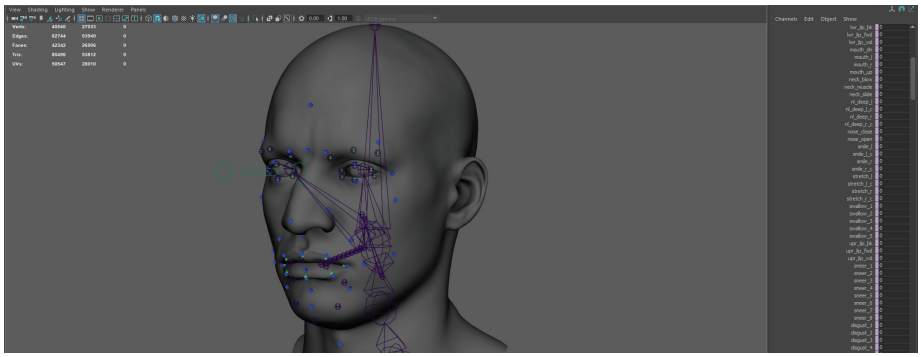
『*The Heretic*』の制作に使用された 4D データフェイシャルキャプチャパイプラインの概要

『*The Heretic*』の場合は、SnappersTech スタジオが 300 個を超えるブレンドシェイプで従来型のフェイシャルアニメーションリグを作成しました。ポーズについてはクリーンアップした 4D データに関連付けることができました。それらを組み合わせることができたら、今度は皮膚に細部のレイヤーを追加しました。そうすることで、最終的にリアルなパフォーマンスを得ることができたのです。

Unity の設定

『*The Heretic*』と『*Enemies*』はリアルタイムデモなので、映画制作よりもゲームプロジェクトに近い方法でキャラクターリグを構築しなければなりません。ジョイントの数については、最適なものにしなければなりませんでした。

『*Enemies*』においては、具体的に言えば血流のシミュレーション、しわマップ、新しいヘアテクノロジーを、HDRP のライティング機能と併用することで、今までにないレベルのリアルさを実現することができました。なお、この新しいヘアソリューションは、GitHub リポジトリで近日中に一般公開される予定です。



『*The Heretic*』の Gawain を作成するためフェイシャルアニメーションリグに 300 個を超えるブレンドシェイプを使用しました。

目

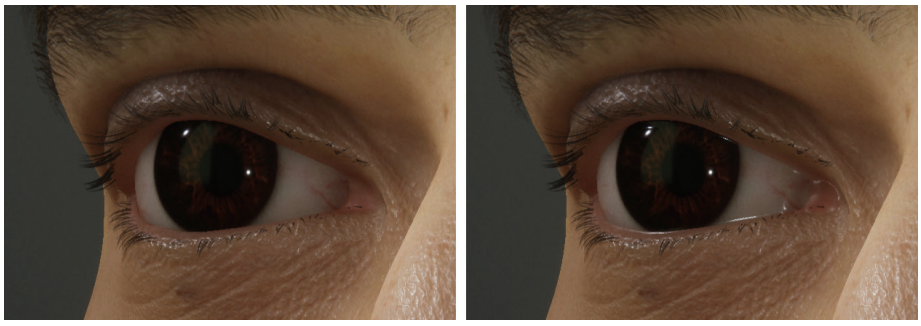
『*The Heretic*』でリアルな目を作成するために、同チームは HDRP の Eye シェーダーを使用し、光を反射する能力などの特性と、目の各コンポーネントに対する制御を追加しました。

『*Enemies*』では、このシェーダーを改良し、角膜曲率によって生じるコースティクスとリアルに輝く虹彩を加えました。このソリューションは、すでにこちらの GitHub リポジトリで公開されています。

『*Enemies*』には HDRP のカスタムバージョンの Eye シェーダーが必要でした。

目には特別な処置を行い、眼球（虹彩と強膜）のディフューズ色とは別に角膜の輝きをレンダリングできるようにしました。また、このプロセスによって虹彩により正確に光を反射させることができました。

『*The Heretic*』では、個別のティアラインメッシュによってキャラクターの目の、眼球とまぶたが接する部分の濡れ具合を制御しています。



ティアライン（右）にまぶたと目の間の濡れ具合が表現されています。

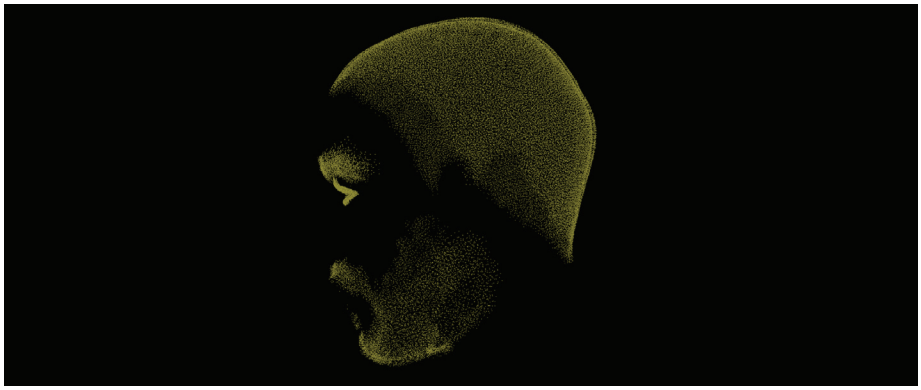
髪の毛

同チームが直面した最も大きな課題の 1 つが、リアルタイムに動作できる詳細な髪の毛を生成することでした。『*Enemies*』では、メインキャラクターが長い髪をなびかせます。ビルトインの物理演算のおかげで、リアルなライティングによる輝きも含めて、本物の髪の毛と同じように表現できています。

当然ながら髪型にはさまざまなものがあるので、同チームはストレート、巻き毛、ショートの各ヘアスタイルを再現する形状をデザインしました。Maya の XGen でさまざまな髪型を作成してから、それらのデザインを Unity にインポートしました。同様のテクノロジーが、まつ毛や眉毛などの他の顔の毛や、皮膚にも応用されました。このテクノロジーは全ユーザーを対象に近日中にリリースされる予定で、業界最高水準のヘアソリューションになることを目指しています。

このヘアソリューションはリアルな髪の毛のシミュレーションに必要な機能セットを完備しています。このスクリーンショットは『*Enemies*』のメインキャラクターの髪の毛が重力に反応する様子を示しています。

一方で Gawain のキャラクターデザインでは、意図的にロングヘアが除外されていますが、よく見ると数多くの短毛が存在していることに驚くかもしれません。デモチームは、眉毛、無精ひげ、まつ毛を適切な位置に保つため、スキンアタッチメントシステムを作りました。顔をどのようにアニメーション化しても、数万本の短い毛は 4D キャプチャに張り付いたままです。



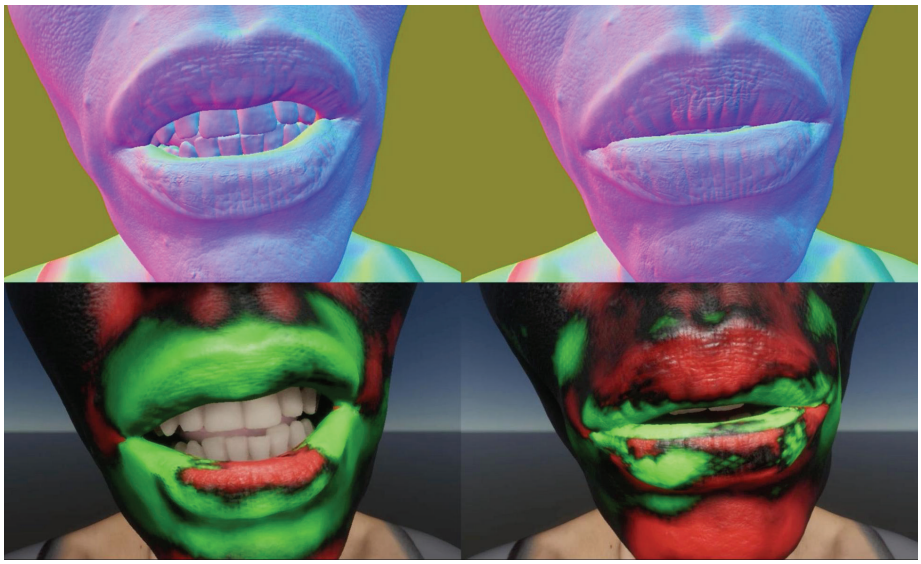
『*The Heretic*』では、スキンアタッチメントシステムによって Gawain に対して数万本の短い毛が適切な位置に保たれています。

皮膚

HDRP の既存の Lit および Layered Lit シェーダーは、デジタルヒューマンのシェーディング機能の強固な基盤となります。このデモでは、HDRP のサブサーフェススキャタリングに対する既存のサポートを活用し、皮膚と歯、両方の有機的サーフェスの下にある領域内で、光が透過したり移動したりする様子のシミュレーションを行いました。

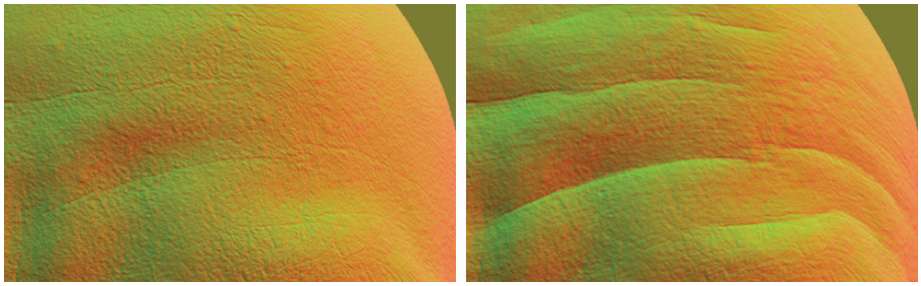
さらに『*Enemies*』デモには、Skin Tension ソルバーが使用されています。これは皮膚が伸び縮みする場所に基づいて複数のテクスチャマップ間で簡単にブレンドを行える、新しいテクノロジーです。このソリューションで使用されるマテリアルには、肌のハリに応じて、しわの多いテクスチャか、しわの少ないテクスチャが採用されます。リグが設定されていても、ボリュメトリックキャプチャでアニメーション化されていても、頭部の皮膚全体や任意のモデルに使用できます。皮膚の伸び縮みする部分がポーズの変化にどれくらい反応するかは、Tension Scale 値によって決まります。

Tension シェーダーツールがシェーダーグラフとともに作成され、最小限のコーディング経験しかないクリエイターでも利用できるようになっています。『*Enemies*』の髪の毛と皮膚のテクノロジーは、どちらも全ユーザーを対象に近日中に公開される予定です。最新情報については、『*Enemies*』のページをご覧ください。



『*Enemies*』からの抜粋：上の 2 つの画像は唇部分に使用された個別の法線マップテクスチャを表しています。デバッグモードで取得した下の 2 つの画像は、皮膚の伸ばされた部分が緑色で示されています。

『*The Heretic*』では、Snappers のリグからポーズ主導の属性を使用し、皮膚の細部をより細かく表現しました。例えば、目を細める表情やしかめ面など、Gawain の一部の表情については、顔に追加される溝を定義したしわマップを生成し、4D キャプチャ上にある程度の表情を追加しました。



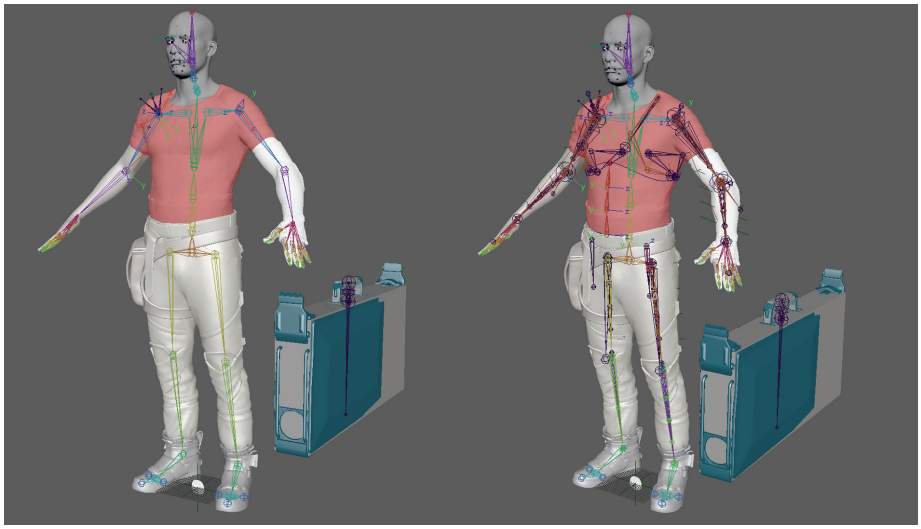
『*The Heretic*』の Gawain に対して基になる 4D キャプチャした皮膚の上にポーズ主導のしわを追加

ベースレイヤーは、Autodesk MotionBuilder® と Maya の間でボディのモーションを伝える役割を担うメインスケルトンで構成されています。MotionBuilder 専用に構築された低忠実度バージョンの Gawain もあり、すべてのモーションキャプチャのクリーンアップ用に使用されました。

2 番目のレイヤーは、デフォーメーションレイヤーとして機能します。デフォーメーションを向上させるため、実際のジオメトリとそれに含まれるジョイントの駆動には、この詳細なバージョンのキャラクターが使用されています。具体的には、腕を長さ方向に沿ってねじるためのジョイントと、肩部分を中心としたデフォーメーションを緩やかにするファンジョイント、ひざ周りの縮みによるアーティファクトを解消するダブルニー設定が、このレイヤーに使用されています。Snappers のフェイシャルリグは、シーン内で参照されてから Unity にエクスポートされました。

ボディのアニメーション

Gawain のアニメーションの大部分は、モーションキャプチャ施設で制作されました。プロセスはかなり標準化されていますが、大きな課題が 1 つありました。それがキャラクターのボディモーションを事前に記録されている 4D フェイシャルパフォーマンスと同期させることです。



MotionBuilder バージョンのリグ（左）を Maya のデフォーメーションリグ（右）と比較。

この同期でボディモーションをフェイシャルパフォーマンスに合わせるために多数のテイクを要しました。アニメーションデータを処理し、細かい調整を適用し、MotionBuilder で動きのタイミングを調整して完成させたアニメーションでは、ボディと顔のパフォーマンスがほぼシームレスに同期するようになりました。

人の顔のシェーディングは容易ではありません。

4D とモーションキャプチャのプロセスの詳細については、こちらの舞台裏を解説したブログ記事を、パイプライン関連の詳細については、Unite Now セッションの「Meet the Devs：ショートフィルム『*The Heretic*』」をご覧ください。

以下に挙げる数値は、『*The Heretic*』のメインキャラクター制作に費やされた膨大な労力を示しています。

— 頂点：頭部に 28,000 個、ボディに 57,000 個、ジャケットに 43,000 個

— Maya スケルトンリグ：約 360 個のジョイント

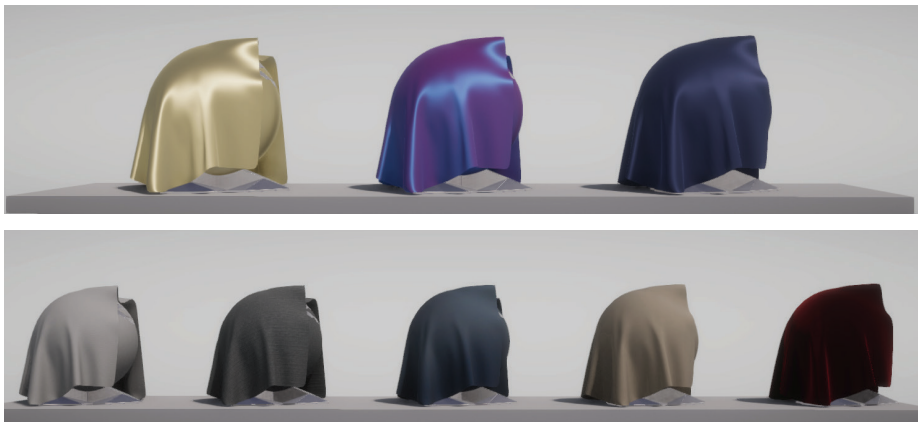
— 基本テクスチャデータ：4,000 個のマップ（アルベド、法線、キャビティ、厚みなど）

— ポーズ主導のテクスチャデータ：48 個のアクティベーションマスクと 16 × 3 の追加の 4,000 個のマップ（アルベド、法線、キャビティ）

— フェイシャルブレンドシェイプ：318 個

小道具と衣服

Unity でキャラクターの小道具や衣服に物理ベースレンダリングを適用すると、フォトリアリスティックなレンダリング結果が得られます。HDRP には、マテリアルやサーフェシングのニーズの大部分に対応できるシェーダーが用意されています。石、金属、布などのマテリアルには、現実世界の同等の素材に合わせた値が使用されています。

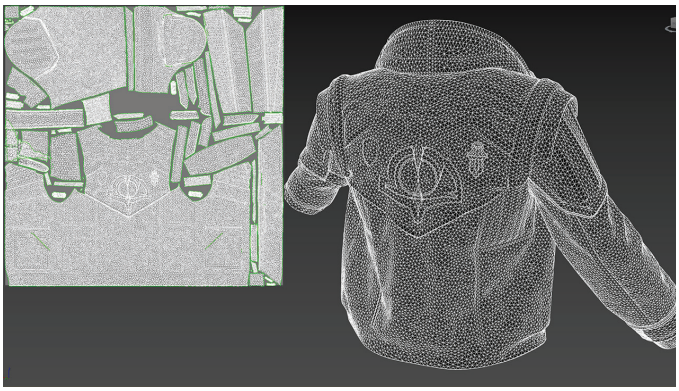


衣服や布のニーズには、綿と絹のシェーダーで対応できます。

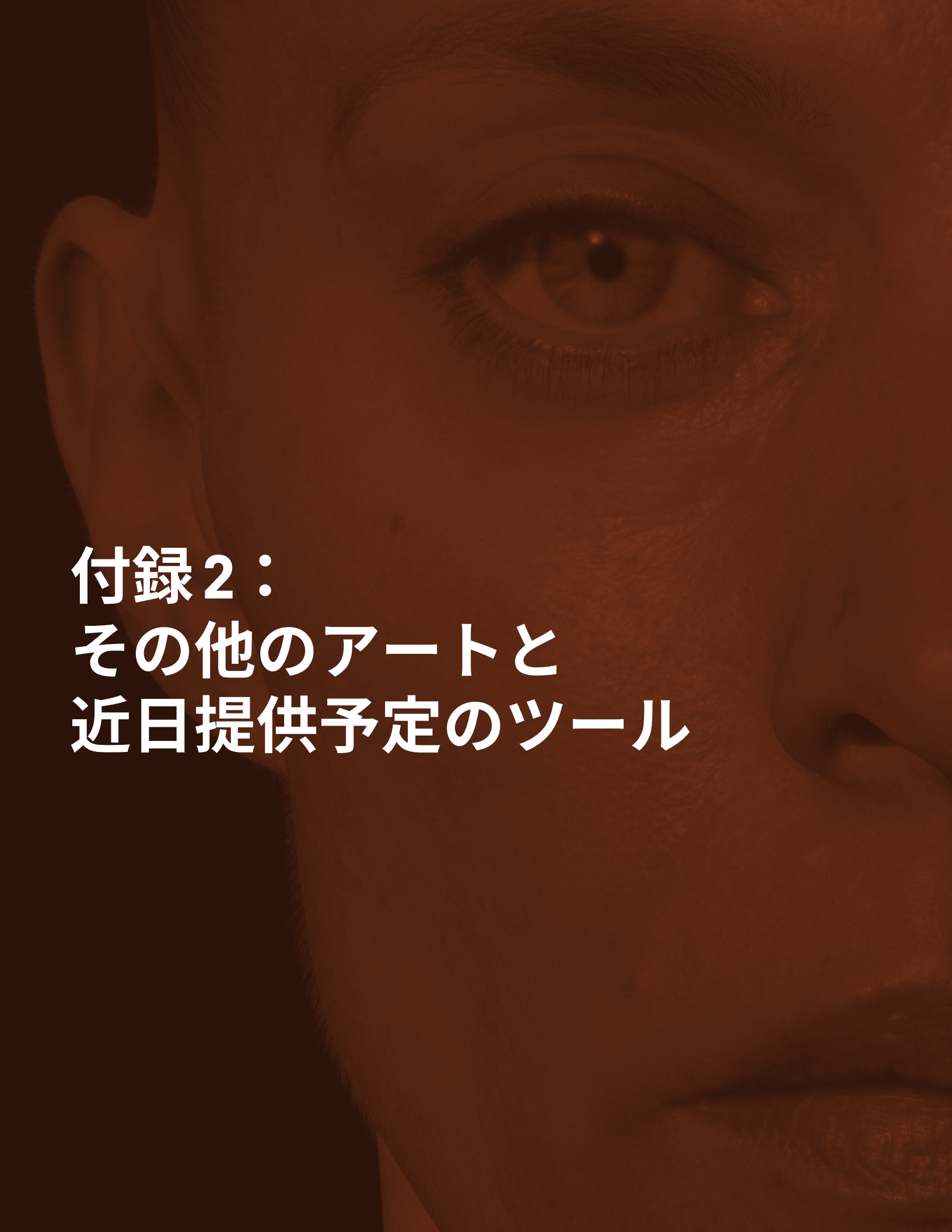
着ている衣服でキャラクターの特徴が決まるわけではありませんが、きれいな見た目にしても損はありません。『*The Heretic*』は、Unity 以外で制作された部分が、エンジン内で制作された部分と同じくらいあります。例えば、アーティストは Gawain のジャケットのシミュレーションを Marvelous Designer で行いました。

それから Alembic ファイルをインポートして、後でリアルタイム再生ができるようにしました。Boston が地下通路で両開きのドアを引き裂いたシーンのように、アニメーション化したジオメトリは、シミュレーションのベイク処理によって得られます。

Gawain のジャケットに使用された布地のマテリアルのレンダリングテスト



Gawain のシャツとジャケットは、どちらも Marvelous Designer で組み立てとシミュレーションを行ってから、Substance Designer でテクスチャを設定しました。

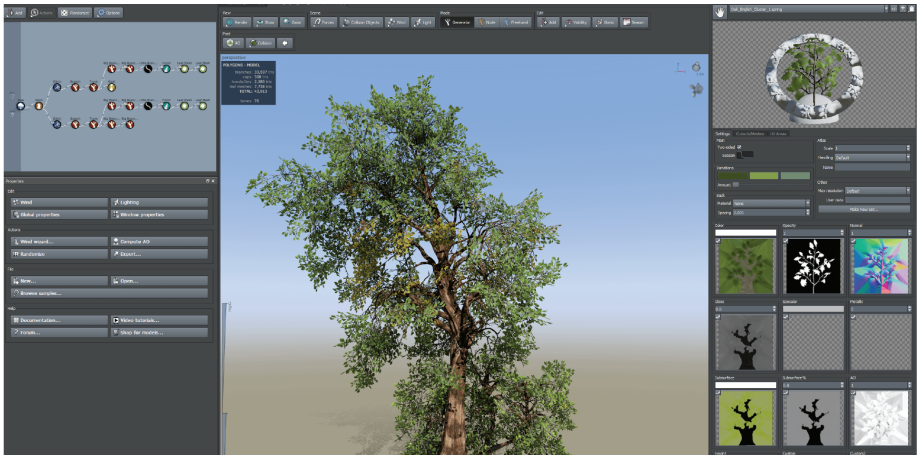


付録2：
その他のアートと
近日提供予定のツール

Unity は、世界中のアーティストがクリエイティブなビジョンを実現できるよう支援することを目指しています。以下のテクノロジーは、多様で要求の厳しい制作現場において、プロのアーティスト向けツールと効率的なチームワークフローを同時に実現します。これらのツールはエンジンに依存しませんが、独自のライセンス契約やスタンドアロンアプリケーションのインストールが必要になる場合があります。

SpeedTree

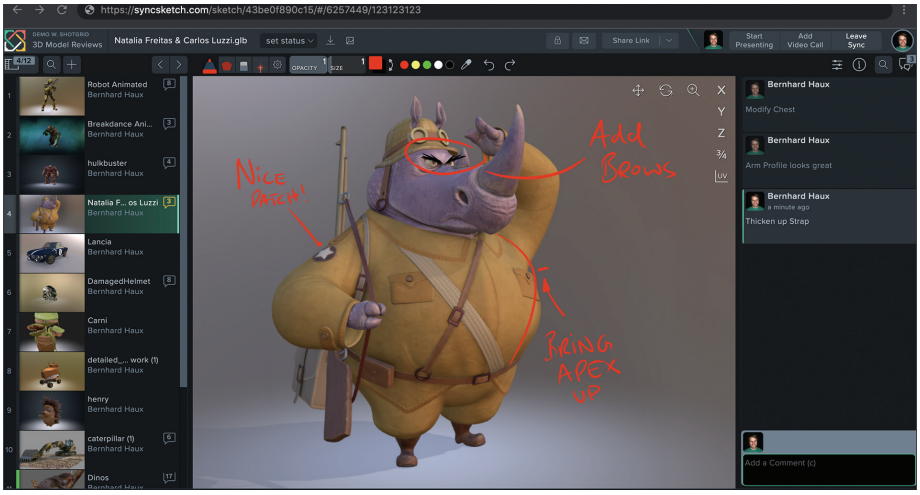
現実に近い大規模で複雑なランドスケープを作るには、多くの場合、樹木や植生が不可欠です。SpeedTree は、SpeedTree ライブラリの包括的な樹木と植生の数々に加え、皆さんのようなビジュアルクリエイター向けの強力な手続き型ハンドモデリングツールを備えています。それらを活用すると、制作中の革新的な体験に、自然な見た目のアセットを追加していくことができます。



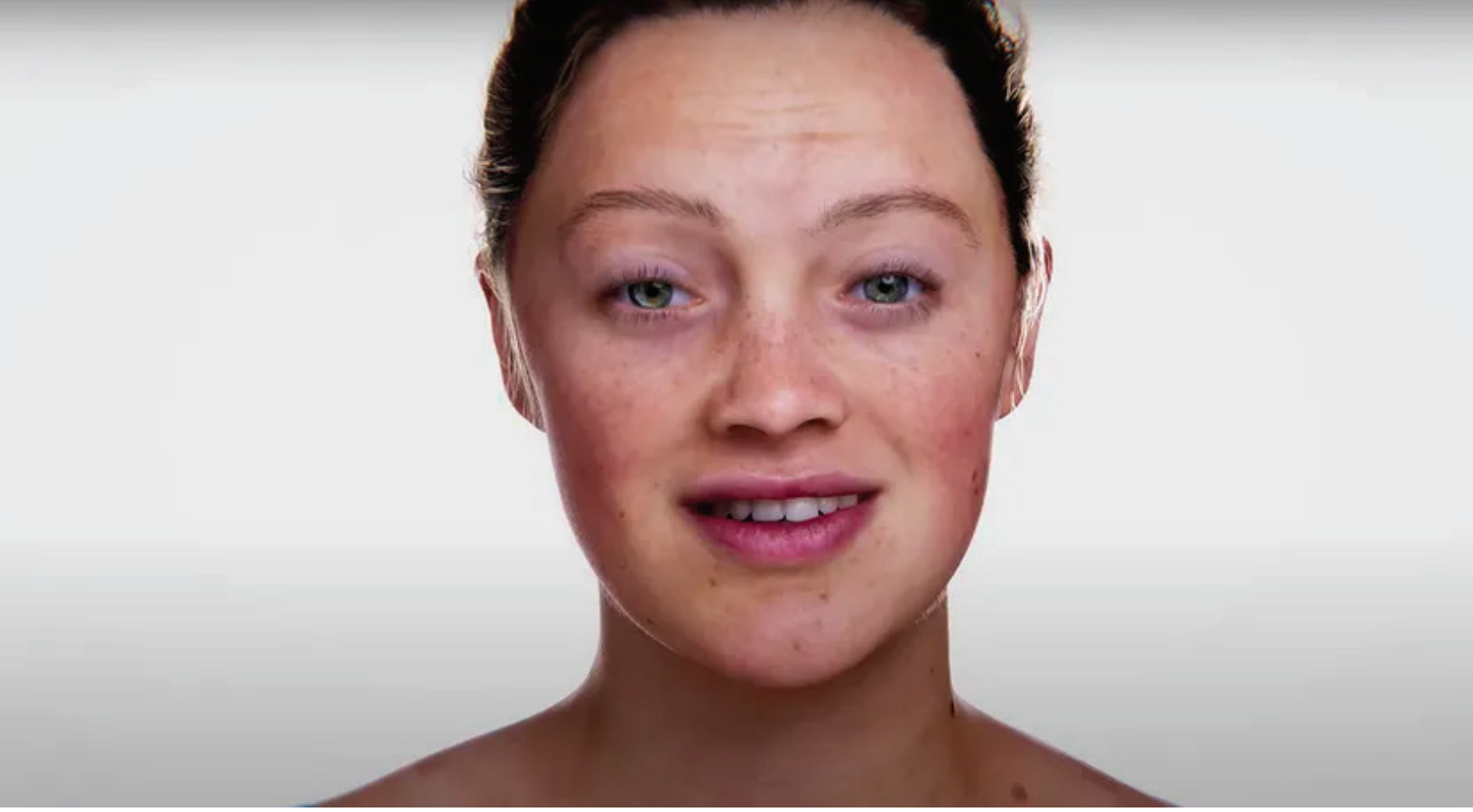
SpeedTree は、多くの商用ゲームや映画に使われているソリューションです。Unity では、Terrain（地形）または SRP ワークフローの一部として利用します。

SyncSketch

アートチーム同士やアートチーム内でのスムーズなコラボレーション向けに設計された SyncSketch は、アーティスト、アニメーター、VFX プロフェッショナル、学生などのクリエイターが、リアルタイムにコミュニケーションを取るのに役立ちます。



SyncSketch は、リンクのみを使用した高速ビジュアルコミュニケーションを実現します。



Ziva のツールで実現できることの例

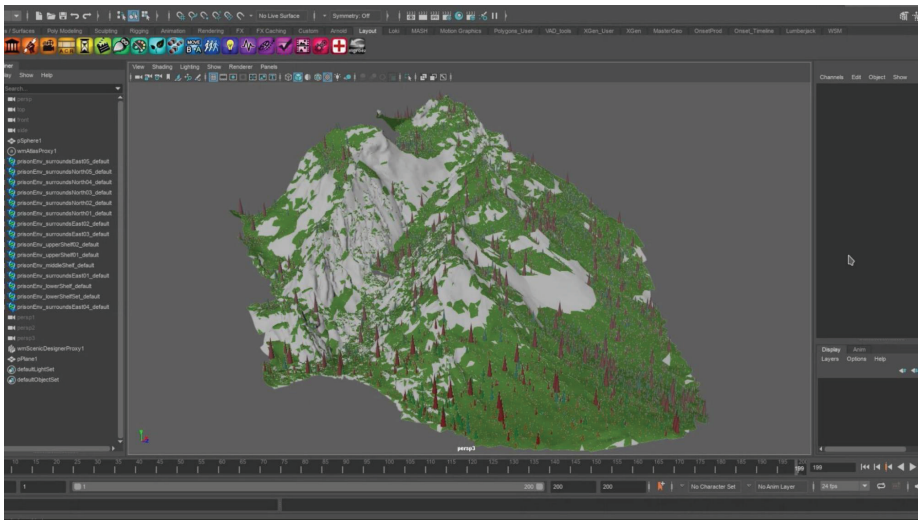
SyncSketch は、2D イメージ、ビデオ、3D モデルをサポートしています。ユーザーは静的またはアニメーション化した 3D アセットの回転、ライティングの変更、メモの追加、フィードバックの共有を、リンクのみを使用した手軽な方法で、リアルタイムまたは非同期に行えます。スマートフォンやタブレットからプロジェクトを確認し、PDF 形式で要約を作成し、それを簡単に共有することができます。ホワイトボード機能もあるので、その場でブレインストーミングを行えます。詳細については、こちらをご覧ください。

近日提供予定のソリューション

近日中にリリースされる Ziva Dynamics のツールは、ゲーム向けか映画向けかにかかわらず、リアルなものから定型化されたものまで、人間や生き物の高品質なキャラクターを作成できるツールとして、新たなスタンダードになるでしょう。Ziva テクノロジーを利用すれば、筋肉、脂肪、皮膚などの軟部組織の物理的特性と物質性をデジタルに複製、連結できるので、アーティストが今までにない最もリアルな CGI キャラクターを作成できます。

2021 年に、Unity は Weta Digital を買収し、同社のアーティストツール、コアパイプライン、知的財産、エンジニアリング人材を獲得しました。アカデミー賞受賞で知られる Weta の VFX サービスチームは、Weta FX の名称で独立会社として存続し、間もなくメディアおよびエンターテインメント業界における Unity 最大の顧客となる予定です。

Weta Digital の業界をリードする VFX ツールと技術力を Unity のリアルタイム開発プラットフォームと組み合わせることで、メタバースの可能性を存分に解き放つソリューションをお届けしたいと考えています。今後の予定については、こちらのブログをご覧ください。

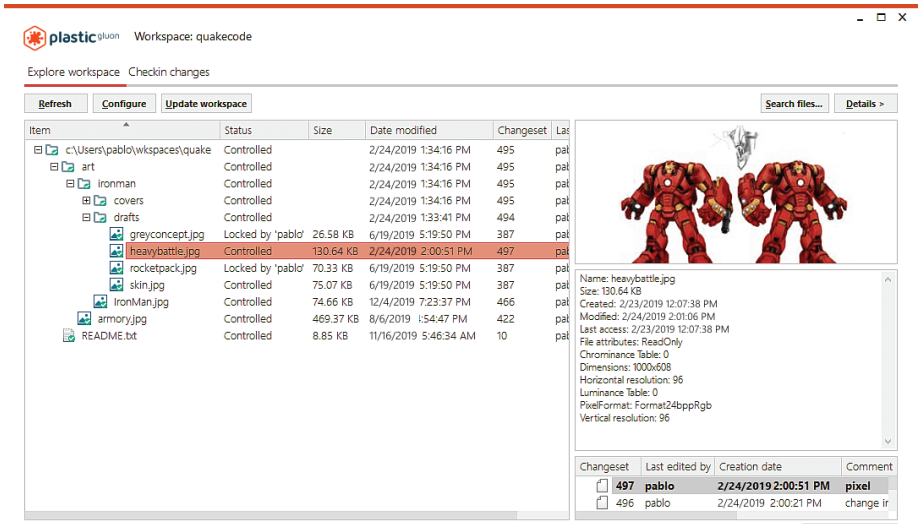


Weta の Totara：植生とバイオーム用の手続き型の生育およびシミュレーションシステム

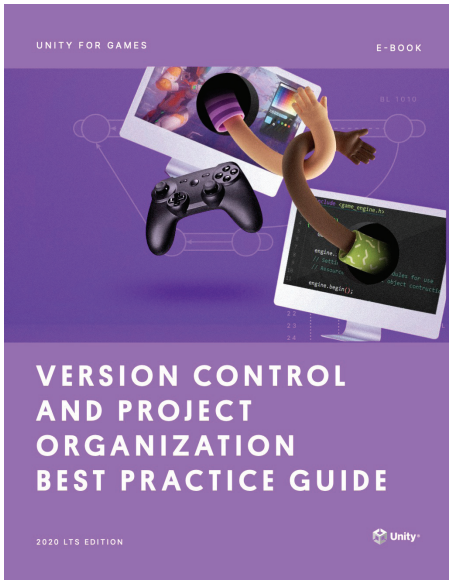
ソース管理

Unity では、Perforce、Plastic SCM、Git LFS（Large File Support）を含む Git など、さまざまなバージョン管理ソリューションを提供しています。Plastic SCM が 2020 年に Unity に初めて仲間入りしたときに、そのバージョン管理ツールがエディターに組み込まれました。

Plastic は、ユースケース主導のゲーム開発向けに設計されています。大きなリポジトリとバイナリファイルの扱いに優れ、プロジェクトビルド全体ではなく、自分の作業するファイルのみをダウンロードできます。Plastic SCM では、プログラマーとアーティスト、両方のニーズに合わせて作られたバージョン管理ソリューションである Gluon も、その他のバージョン管理ツール、ホスティング、GUI クライアントとともに提供しています。



Plastic のワークフローは、技術的知識の有無にかかわらず、すべてのクリエイターがファイルのプレビュー、変更のチェックイン、履歴の確認を行えるよう設計されています。



バージョン管理とプロジェクト整理のベストプラクティス

技術的知識の有無にかかわらず、すべてのクリエイターを対象としたこの e ブックでは、バージョン管理および計画を設定して生産性の高いチームワークを実現する方法について説明します。このガイドでは、集中型システムと分散型システムの比較や、Git、Perforce、Plastic SCM などのソリューションを取り上げるほか、Unity プロジェクトを効果的に整理する方法についても解説します。

e ブックをダウンロード

その他のリソース

SpeedTree の YouTube チャンネル
Ziva Dynamics のウェブサイト
Weta Digital のウェブサイト
Plastic SCM

アーティスト向けの Unity

アートや技術に関するその他のヒントについては、**Unity ブログ**を確認するか、ハッシュタグ #unitytips を **Unity Learn** および Unity の**コミュニティフォーラム**で検索してください。**Unity の開発者向けツール**のマイクロサイトでも、ドキュメントから Unity の最新ロードマップやリリースの詳細まで、テクニカルアーティスト向けの主要なリソースがいくつか提供されています。

その他のサポートをお探しですか？ Unity プロフェッショナルトレーニングでは、テクニカルアーティストやクリエイティブチームがプロジェクト目標の達成に必要なスキルを身に付けることができます。Unity を使用して制作を行うあらゆる業界、あらゆるスキルレベルのプロフェッショナル向けに設計された、広範なトレーニングのカタログを用意しています。すべての教材は、Unity の経験豊富なインストラクショナルデザイナーがエンジニアや製品チームと協力して制作しています。これは、常に Unity の最新テクノロジーに関する最新のトレーニングを受けることができることを意味します。

皆さんや皆さんのチームが受講できるトレーニングの詳細は、**こちら**で確認してください。

このガイドの制作に貢献していただいたエキスパートの皆様に感謝の意を表します。読者の皆様もありがとうございました。今後のクリエイティブなプロジェクトのご成功をお祈りしています。


Unity®