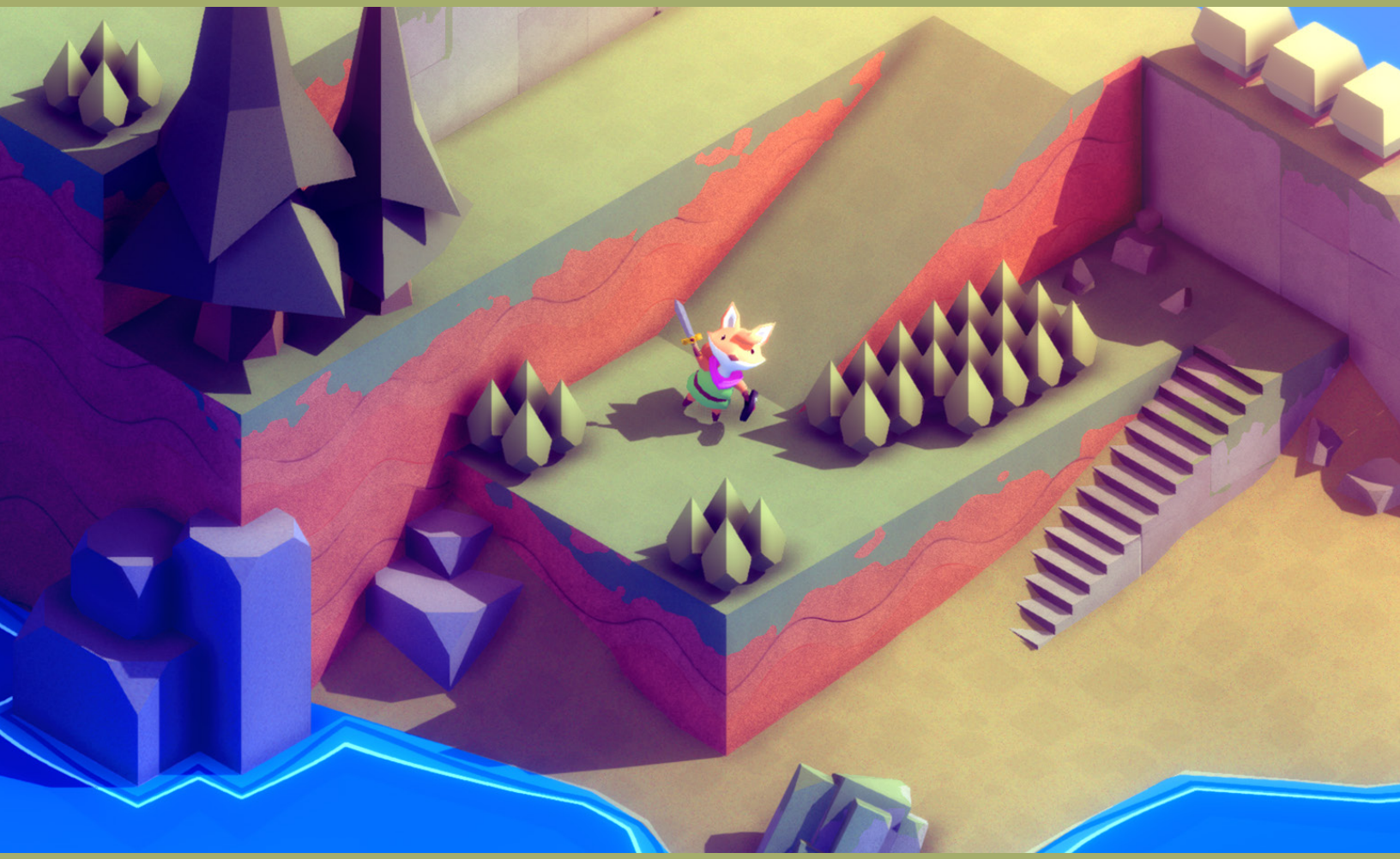
Unity®

E-BOOK



ゲームレベルデザイン入門

UNITY 2022 LTS エディション

目次

入門

仮想世界の創造者であり、空間認識の魔術師であるレベルデザイナーは、ゲーム開発のマスタービルダーとして、完全に実現されたプレイアブル空間の一貫したレイアウトと構成を担っています。

この 3 つのパートからなるガイドは、レベルデザイナー志望者と経験者の両者を対象としています。Unity 社内およびプロのゲーム開発チームのレベルデザイナーとゲームデザイナーによって執筆されました。

Unity は最も広く使用されているゲーム開発プラットフォームです。Unity の開発スキルを習得することは、他のエンジンを使用するチームに所属していても、新たなチャンスを掴むきっかけとなるはずです。

このガイドの内容

パート I では、ゲーム開発制作サイクルにおけるレベルデザイナーの役割や位置付けについて紹介します。ここで取り上げるトピックは、アイデアのリサーチと開発、ホワイトボクシング（グレーボクシング）によるデザインのプロトタイピング、そしてプレイヤーパス、ライティング、手がかり、ポイント獲得といったレベルデザインにおける基本要素の取り扱いなどです。

パート II では、Unity エディター、Unity でシーンに要素やインタラクティブ性を追加するための基本的な構成要素、そして、アセット、スクリプティング、物理演算、アニメーションの基本事項について紹介します。

パート III では、ProBuilder と Polybrush ツールセットの詳しい使い方、Unity Terrain システムの概要、Unity Asset Store で入手可能なおすすめツールを紹介します。

Ustwo Games の『Monument Valley 2』は、3D 空間の独創的な使用が特徴のパズルゲームです。

この eBook は、Unity ツールに関するゲームデザイナーおよびレベルデザイナー向けのより全般的な概要や、ゲームプレイデザインのエキスパートによるヒントを記した「The Unity Game Designer Playbook」を補完するものです。

寄稿者

Christo Nobbs 氏は、システムゲームデザインと Unity（C#）を専門とするシニアテクニカルゲームデザイナーです。Unity 4 時代からの Unity ユーザーであり、PUBG/PlayerUnknown Productions でテクニカルゲームデザイナーを務めた経験があります。

Stefan Horvath 氏は、ゲーム開発業界で 10 年以上、レベルおよびゲームデザイン、そして品質保証に携わってきました。これまでに手掛けたタイトルには、Cloud Imperium Games の『Star Citizen』、Behaviour Interactive と 505 Games の『Dead by Daylight』、そして Norsfell Games Inc. の『Tribes of Midgard』などがあります。

Eduardo Oriz 氏は、Unity でシニアコンテンツマーケティングマネージャーを務め、2D ツールグループなどの Unity 開発チームと何年にもわたり連携してきた経験があり、ゲーム開発者やスタジオに上級者向けのチュートリアルコンテンツを提供しています。

パートI：ゲーム開発におけるレベルデザイン入門

頻出のレベルデザイン用語

レベルデザイン用語	説明
3C	キャラクター、カメラ、コントロールの頭文字を取った略語です。3C は、プレイヤー入力と画面上のキャラクターの反応、およびそのキャラクターがゲームのカメラフレーム内でどのように写るかという、複雑な相互作用関係を素早く説明する際に使用されます（ソース：Pluralsight）。
アフォーダンス	オブジェクトの性質またはプロパティを指し、考えられるユースケースを定義したり、オブジェクトの使用方法を明確にしたりします。（ソース：Merriam-Webster）。
AI	ゲームの AI（人工知能）とは、人間による入力ではなく、コンピューターコードに依存して動作するゲーム内のエンティティのことです。一般的な AI エンティティとして、NPC（プレイヤーが操作しない Non Player Character）が挙げられます。
ブロックアウト	プリミティブジオメトリ形状とゲームアートを用いて、非常にシンプルなレベル環境デザインを作成するプロセスです（モックアップと似ています）。
カラーコーディング	識別とカテゴリ化の手段として、デザインモックアップの要素を異なる色でマークすることを指します。

Clever Endeavour Games の『Ultimate Chicken Horse』は、友達とレベルを作成して、クリアを目指すパーティ型プラットフォームゲームです。

クリティカルパス	プレイヤーが辿りうるゲームクリアまでの最長パスです。
ファーストプレイアブル	ファーストプレイアブルとは、主要なゲームプレイ要素（とアセット）を提供する、プレイ可能なゲームバージョンのことです。多くの場合、プリプロダクションで制作されたプロトタイプを基にしています（ソース：Wikipedia）。
FOV	一人称視点ゲームにおいて、有効視野（FOV）とは、任意の瞬間に画面上で確認できるゲームワールド範囲のことです（ソース：Wikipedia）。
ゴール	ゴールとは、プレイヤーがゲームをクリア / 完了するためにこなす必要があるチャレンジやアクティビティのことです。ゲーム全体にゴールを設定することで、プレイヤーに達成感を与え、プレイし続けるモチベーションを高められます。
ゴールデンパス	通常、この用語は最高のゲームプレイ要素、ストーリー、ゲーム内報酬や隠し要素を提供するパスのことを指します（ソース：tv tropes）。
インタラクタブル	プレイヤーが相互作用できるオブジェクトのことです。

レベルデザイン用語	説明
見通し	見通し（英語の「Line of sight」から LoS と略されることもある）とは、戦争ゲームやロールプレイングゲーム（RPJ）における、フィールドの可視性（誰が何を見ることができるか）を指します。
メッシュ	ビデオゲームでモデルの基盤となる、頂点、辺、面の集合体のことです。
ミニマップ	ミニマップは、通常画面の端に配置される小さなリファレンスマップで、プレイヤーがゲームレベル内を移動する際に使用されます（ソース：Sharp Coder blog）。
改造	ビデオゲームにおける改造とは、ビジュアル要素、キャラクター、動作など、ゲームの 1 つ以上の側面をプレイヤー / ファンが改造する過程のことを指します。
経路検索	経路検索は、AI による環境内移動に使用される手法で、これには通常ナビゲーションメッシュ（ナビメッシュ）が使用されます。
プレイテスト	プレイテストとは、バグを発見し、ゲームプレイの流れを確認し、改善の余地を探るために、ゲームのすべての新しいビルドをプレイすることです。
プロトタイプ	ゲーム開発におけるプロトタイプとは、基盤となるゲームプレイのアイデアをテストし、投資家やその他の関係者にコンセプト実証として提示する目的で制作される、ゲームシーンの簡素化されたサンプルのことです。
可読性	読みやすい、または解読しやすい性質を指します。
トップダウン	ビデオゲームにおけるトップダウン視点（俯瞰視点またはヘリコプター視点）とは、プレイヤーが上からレベルを見下ろす視点のことです。また、上から見たレベルの 2D ビジュアライゼーションを表すために使用されます。
ビューポイント	対象を見たり、考慮したりするための位置または視点（ソース：Merriam-Webster）。
ワイヤーフレーム	ワイヤーフレームモデルは、3D コンピューターグラフィックスで使用される 3D 物理オブジェクトを視覚的に示したものです（ソース：Wikipedia）。
ホワイトボクシング（またはグレーボクシング）	ホワイトボックスは、単純な 3D 形状で作成されるレベルで、レイアウトや動線をテストするために使用されます。

Unity Documentation の用語集では、ゲーム用語についてさらに詳しく確認できます。

プリプロダクション

このセクションでは、デザイン哲学、指標、ペーパーデザイン、ゲームメカニクスの設計など、ゲーム開発のプリプロダクション段階において重要ないくつかのトピックやタスクについて説明します。十分に思索した上で取り組むことで、これらのプロセスはゲームのビジョンをより鮮明にし、効率的な制作への移行を可能にします。

一貫したデザイン

一貫したレベルデザインは、テーマ、スタイル、時代など、作成するゲームワールドを総合的に理解することで生まれます。これらの要素のリサーチ（同僚と一緒に、ゲームのテーマやバックストーリー、主要な指標やゲームメカニクスの肉付けを行うなど）は、没入感があり細部まで作り込まれたワールドをプレイヤーに提供する上で非常に役立ちます。

執筆においては、文章や段落同士を論理的かつスムーズにつなげることで、一貫性が生まれます。同様に、ゲーム開発における一貫したデザインでは、ゲームデザインの各要素が論理的に連結し、全体的な体験に貢献する、体系的で一貫性のあるデザインの実現を目標とします。

Acid Nerve の『Death’s Door』は相互に関連するレベルデザインを上手く実現しています。同チームへの Unity Creator Spotlight のインタビューはこちら。

強固で一貫したゲームデザインを巧みに実現したゲームの例として、『Death's Door』や『Cuphead』が挙げられます。開発者曰く、『Death's Door』のレベルデザインは『DARK SOULS』シリーズから着想を得たそうです。

「初めて『Dark Souls』をプレイしたときに驚いたのは、レベルに 3 次元空間が使われていたことでした」と、Mark Foster 氏（リードプログラマー、コードデザイナー、アニメーター、ライター）は述べます。「すべての要素が重なりあっていて、近道を見つけるのがすごく楽しいのです。」

レベルの結合性は、プレイヤーを実際の空間のように感じられるゲームワールドに没入させることができ、プレイヤーは、レベルが直線的でなく、自然な形で設計されていることなどに気づくとき、喜びを感じることができます。

『Cuphead』では、ビジュアルからゲームメカニクス、サウンドまで、すべてのコンポーネントが連動し、一貫した体験を生み出しています。

リサーチおよびリファレンスガイド

リサーチ結果とリファレンスをまとめてドキュメント化またはガイド化し、構築しているゲームワールドやレベルを理解するのに役立てましょう。よくまとまった、定期的に更新されるリファレンスガイドは、ゲームのルックアンドフィールの実現に関して、開発チーム全体の認識を揃えるのに役立ちます。

Studio MDHR の『Cuphead』は、1930 年代の漫画に大きなインスピレーションを受けています。

リファレンス画像のまとめや共有に使えるアプリの例として、Pureref、Figma、Pinterest などが挙げられます。以下に、レベルデザインアイデアの参考となるアプリおよびウェブサイトをいくつか記載します。

— **VGmap**：古典的な（主に 2D の）ゲームマップのアーカイブ

— **Noclip**：古典的な 3D ゲームマップのアーカイブ

— **地理情報システム（GIS）**：地理データを格納する一種のデータベースで、これらのデータを管理、分析、視覚化するソフトウェアツールも統合されています。現実世界とその自然分布を参照して、GIS データを地形およびハイトマップのベースに使用し、それを基に構築しましょう。Unity には、Unity Asset Store から入手可能な Terraworld Automated Level Designer など、GIS データを活用するためのツールが数多く用意されています。

GISTech による Terraworld Automated Level Designer は、現実世界のリファレンスを使用して地形を生成する Unity プラグインです。

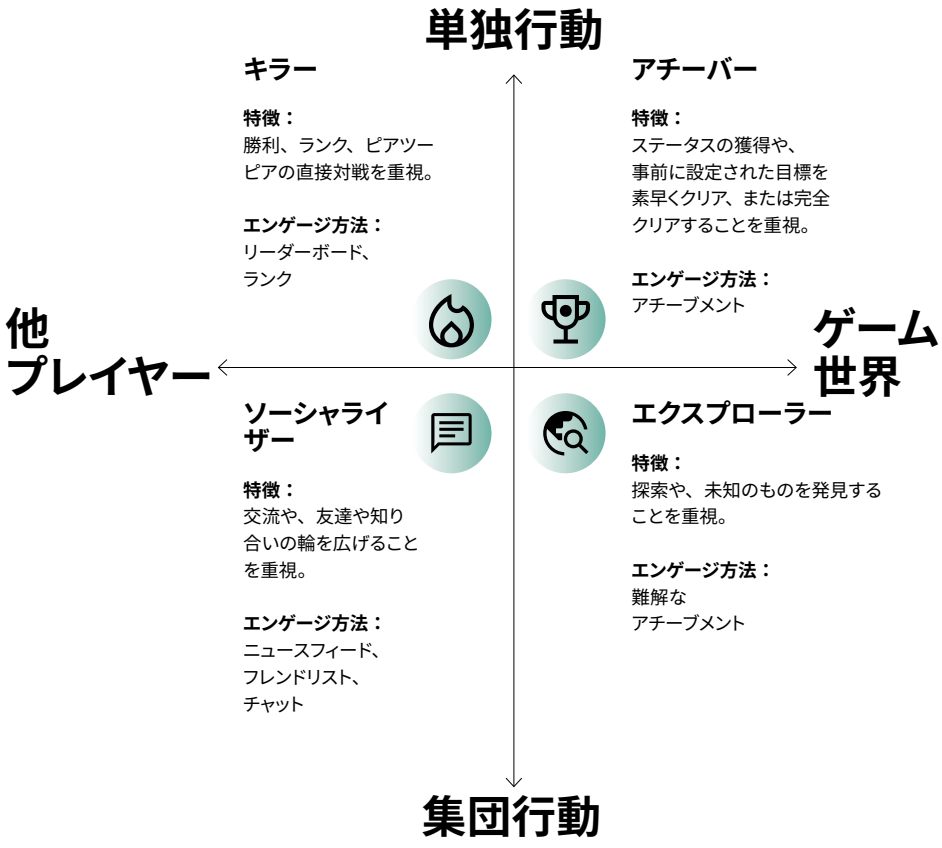
また、他のゲームからインスピレーションを得たり、参考にしたりしましょう。時には、他の人がプレイしたり視聴したりできる機能的なリファレンスを使用することで、アイデアの魅力が伝わりやすくなったり、理解しやすくなることがあります。

ターゲットオーディエンスを知る

オーディエンスを知ることは、レベル計画において重要です。ターゲット層は、ハードコアなゲームオーディエンスですか？想定している年齢層やプレイスタイル傾向はありますか？

通常、ターゲットオーディエンスはクリエイティブディレクターやプロダクション、マーケティング部門によって設定されます。担当チームの同僚に確認を行い、ゲームのターゲットオーディエンスに関する認識を合わせておきましょう。

ゲーム内の行動傾向に基づきプレイヤーを分類するバートルテストについて知っておくことも役に立つでしょう。これは、パーソナリティ理論に基づき、4 種類のパーソナリティを定義しています。アチーバー（達成者）、エクスプローラー（探検家）、ソーシャライザー（社交家）、キラー（殺人者）です。



バートルテストは、Richard Bartle 氏により 1996 年に発表された論文に基づく、ビデオゲームプレイヤーの分類法です。画像ソースはこちら。

レベルデザイナーは、これらのパーソナリティアーキタイプを意識しながら、ワールドを構築するとよいでしょう。

制作中のゲームは、チーム対戦型マルチプレイヤーゲームでしょうか？その場合、このタイプのゲームはピアツーピアの対戦やチームワークに焦点が置かれることが多いため、パーソナリティアーキタイプはキラーとソーシャライザーになるかもしれません。

シングルプレイヤーのオープンワールドアドベンチャーゲームであれば、プレイヤーはゴールを達成しながらオープンワールドを探索できるため、パーソナリティアーキタイプはエクスプローラーとアチーバーになるかもしれません。

ゲームは 1 つのパーソナリティアーキタイプのみに焦点を絞る必要はありません。ほとんどのゲームでは、すべてのアーキタイプをターゲットに含めることができます。例えば、シングルプレイヤーゲームの場合、ギフトの交換やプレイヤーベースの訪問などを通して、他のプレイヤーとのゲーム外交流を可能にすることで、ソーシャライザーアーキタイプを含めることができます。マルチプレイヤーシューティングゲームであれば、キラーアーキタイプを主なターゲットとしつつ、アチーバーにも訴求できます。

ただし究極的には、ゲームのビジョンを主軸に、主要または最も重要なパーソナリティアーキタイプのニーズを満たすことを目指すべきであり、多くのプレイヤータイプに合わせるためにゲームのプレイ性を弱めるべきではありません。

3C：カメラ、キャラクター、コントロール

レベルデザイナーは通常、ゲームカメラ、キャラクター、コントロールコンポーネントの微調整を行い、ゲームデザイナーの作業を容易にするためのテスト用の「ジム」を作成します。このプロセスは、レベルデザイナーがゲーム内で 3C がどのように設定されているかを理解する助けとなり、最終的には、重要な指標を確立するための土台となります。

実際にプレイヤーに動きを教えて見せる役割を果たすことになるので、体験を向上させるためにゲームデザイナーにフィードバックを提供しましょう。

カメラ

カメラは指標を操作するのに役立ち、プレイヤーがゲームをプレイする際の視点を提供します。

レベルデザイナーが制作過程でどのようにカメラを考慮するのかについて、いくつか例を挙げます。

- カメラの高さを使用して、プレイヤーがドアを通過する際のクリッピングを回避する
- 有効視野（FOV）とカメラを使用して、プレイヤーの視点とゲーム内のシーンがどのように表示されるかを示す
- 異なる視点の長所と短所を比較検討する：一人称視点の場合、プレイヤーは一方向を向いているため、背後から忍び寄る敵を発見する能力が制限されます。対照的に、アイソメトリック視点のゲームの場合、プレイヤーは全方向を見渡せます。

キャラクター

キャラクター要素は、キャラクターの武器、能力、その他のユニークな特徴に言及するものであるため、レベルデザイナーにとっては重要です。究極的には、これがプレイヤーのレベル内での移動やエンゲージ方法を決めるツールキットです。

カメラの位置や FOV は、キャラクターや環境の大きさについての感覚を与えます。VR では、環境に没入するため、この点がさらに重要になります。VR Beginner:The Escape Room のチュートリアルで、VR についてさらに学びましょう。

レベルのプレイ方法に影響を与える可能性があるため、開発過程でキャラクターの能力やその他の特性に変更があった場合は注意が必要です。

レベルデザイナーがキャラクターを考慮する方法の例をいくつか記載します。

— キャラクターの能力および、それをどのように段階的にプレイヤーに教えて見せることができるか、いくつか例を挙げます。
— ゲーム内の任意の時点でのキャラクターの力や、様々な段階でのチャレンジがキャラクターの強さ / パワーレベルに合っているか
— プレイヤー向けのまたはプレイ方法に関するキャラクターカスタマイゼーションオプション（例：ステルス、ラン、ガン、ハッキングなど）
— プレイヤーのダブルジャンプ能力：ジャンプ可能な高さと距離はどれくらいか、またその能力を上手くデザインに活用できているか

コントロール

コントロールは、任意の周辺機器を使ってプレイヤーがどのようにキャラクターを操作するかを決定します。ステルスでレベルを進む、木々の間を駆け抜ける、車で突っ走る、『Fall Guys』のキャラクターのようにぶつかり合うなど、様々な場面で、優れたコントロールはゲームプレイに大きな影響を与えます。

異なる対象周辺機器を使ってレベルを最後までプレイしてみてください。ターゲットオーディエンスによっては、コントロールの差異を補うために、短い連続したアクションの繰り返しを減らしたり、通路を広げたりした方が良い場合があります。

Monomi Park の『Slime Rancher 2』は、一人称視点のハイペースなアドベンチャーゲームです。他の FPS と同様に、旋回、照準、移動の応答性が高いコントロールは、ゲームプレイ体験の重要な部分を担います。

例えば、値がクランプされない限り、マウスとキーボードによる操作の方が、ジョイスティックよりも動きの精度が高く、可動域も広くなります。ジョイスティックは通常、0 〜 100 の値の間を一定の速度で移動できますが、その分マウスやキーボードに比べて回転速度や反応速度が遅くなります。

カメラ、キャラクター、そしてコントロールの詳細は、Unity Learn および Unity Documentation で確認できます。

指標

ゲーム指標は、プレイアブル空間における関係性を規定します。レベルと空間の構成を決めるものであるため、制作開始前に特定および設定しておくことが重要です。

例えば、高速で移動するキャラクターは、動きが遅いキャラクターよりも広いプレイアブル空間を必要とします。理想は、レベルデザイナーがレベルを作成する前にほとんど指標を手元に準備し、その後で「トレーニング環境」内で指標をテストすることです。「トレーニング環境」は、基本的にゲームのコンポーネントを探索するためのプレイグラウンドです。

別の例は、プレイヤーが屈んだり、オブジェクトの後ろに潜んで敵から隠れたりできるゲームの場合です。このシンプルなメカニズムだけでも、以下の指標を知る必要があります。

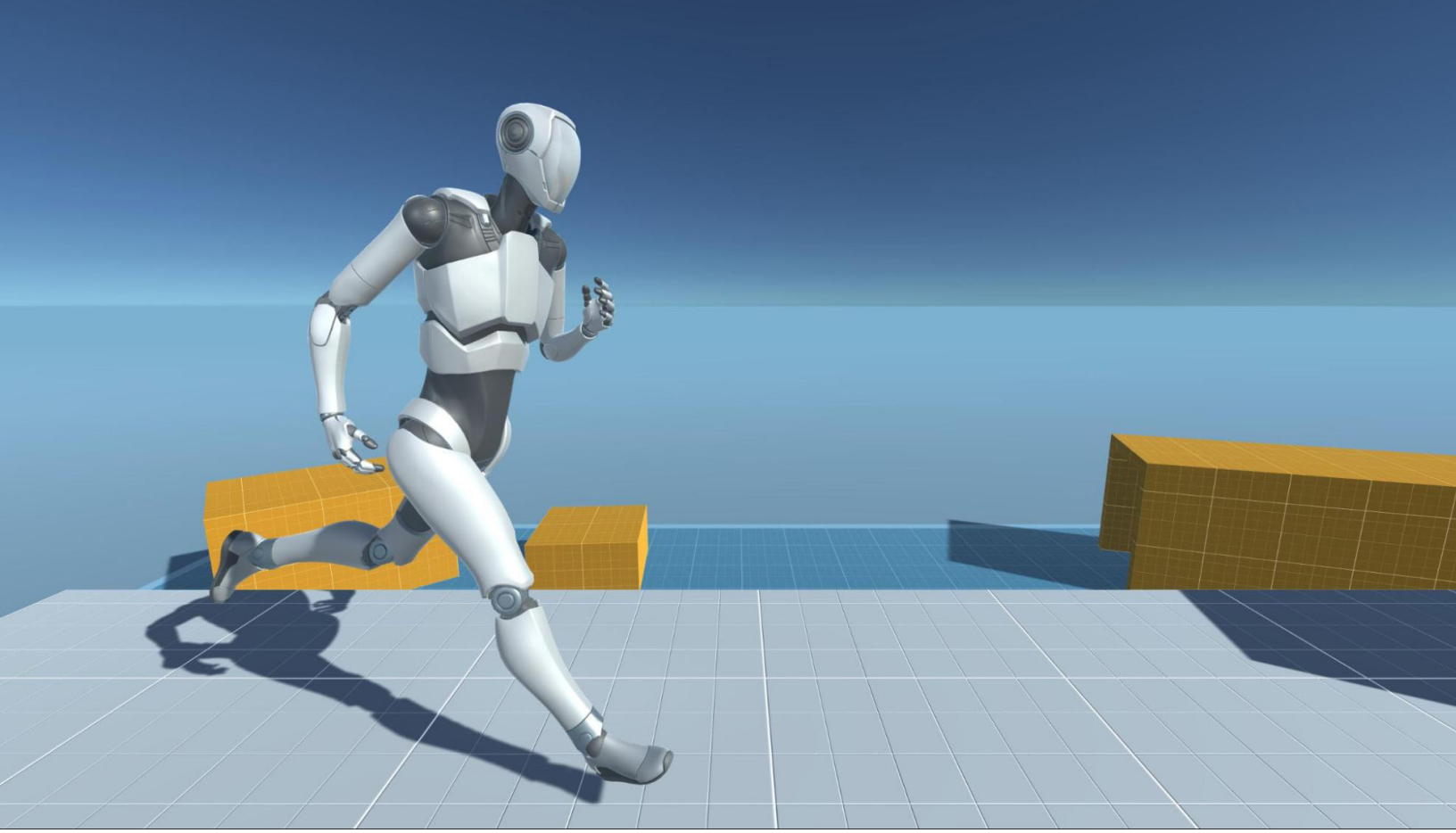
— 屈んだ際のプレイヤーの高さ（キャラクターモデルの垂直の高さ）

— 屈んだ際のプレイヤーカプセル / ヒットボックスの高さ（プレイヤーへの「ヒット」を決める高さで、通常キャラクターモデルと同等以上の高さ）

この 2 つの指標を使用して、障害物の最小の高さを決定し、プレイヤーが障害物の後ろに潜んで敵から身を隠したり、障害物の反対側から敵の攻撃を防いだりできるようにします。

ゲーム開発サイクルを通して、指標を変更する際は常に注意が必要です。上記の例では、レベル作成後に指標を変更すると、元々のカバーに関するルールが考慮されなくなるため、すべてのカバー障害物を変更する必要が出てくるでしょう。変更を加えたことで、隠れているはずのプレイヤーが見つかったり、攻撃を受けたりするなど、意図しない結果に繋がる可能性があります。

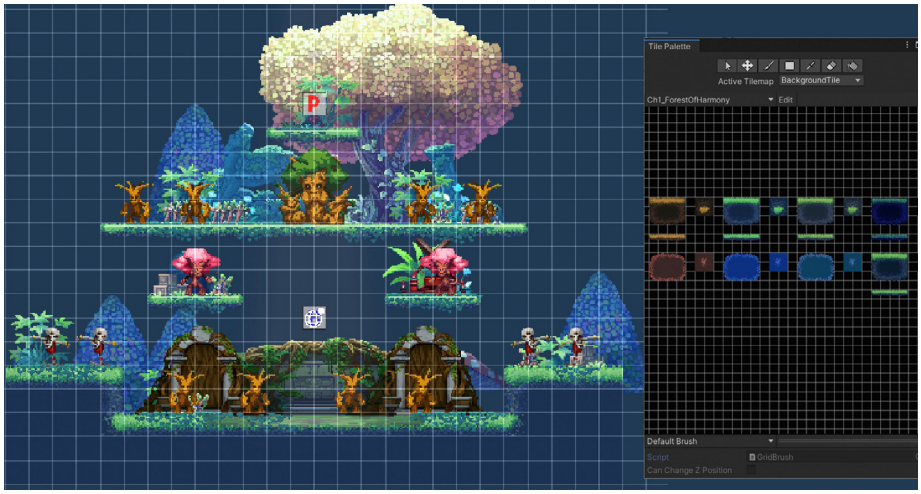
通常、Unity のような 3D ソフトウェアやゲームエンジンは、現実をシミュレートするために、物理演算、カメラ、レンダリングなど、他のシステムの基準として機能するデフォルトのユニットサイズで設計されています。Unity ユニットは 1 メートルに相当し、これが Scene ビューで見えるグリッドサイズになります。キューブなどの新しいプリミティブ 3D オブジェクトを作成すると、1 立方メートルの体積を占めます。環境とキャラクターを作成する際は、このスケールを考慮することが推奨されます。



Starter Assets は、Unity Asset Store で入手可能な、キャラクターコントローラーやカメラセットアップが含まれた Unity テンプレートです。テストレベルには、Unity のユニットサイズリファレンスに従ったグリッドテクスチャが含まれています。

Unity で 2D 環境を作成する際は、現実世界のスケールを保ちながらも、柔軟に対応することが推奨されます。例えば、2D ゲームでタイルシステムを使用している場合、ゲームの他の部分がそのシステムに基づいた設計になっていれば、代わりにそのリファレンスを使用することができます。

Unity ゲームグリッドの調整方法に関する詳細は、後述の「Unity を始める」のセクションで確認できます。



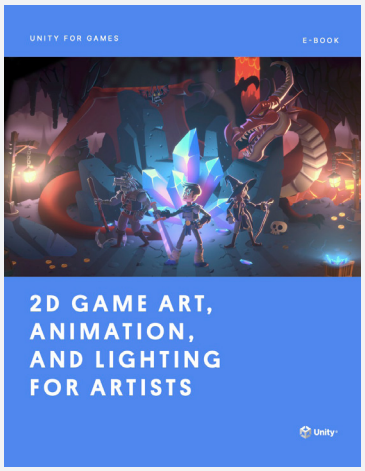
『Skul: The Hero Slayer』は、Unity を用いて制作されたピクセルアートスタイルのローグライトゲームで、ワールドの構築に 2D Tilemap Editor を使用しています。

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

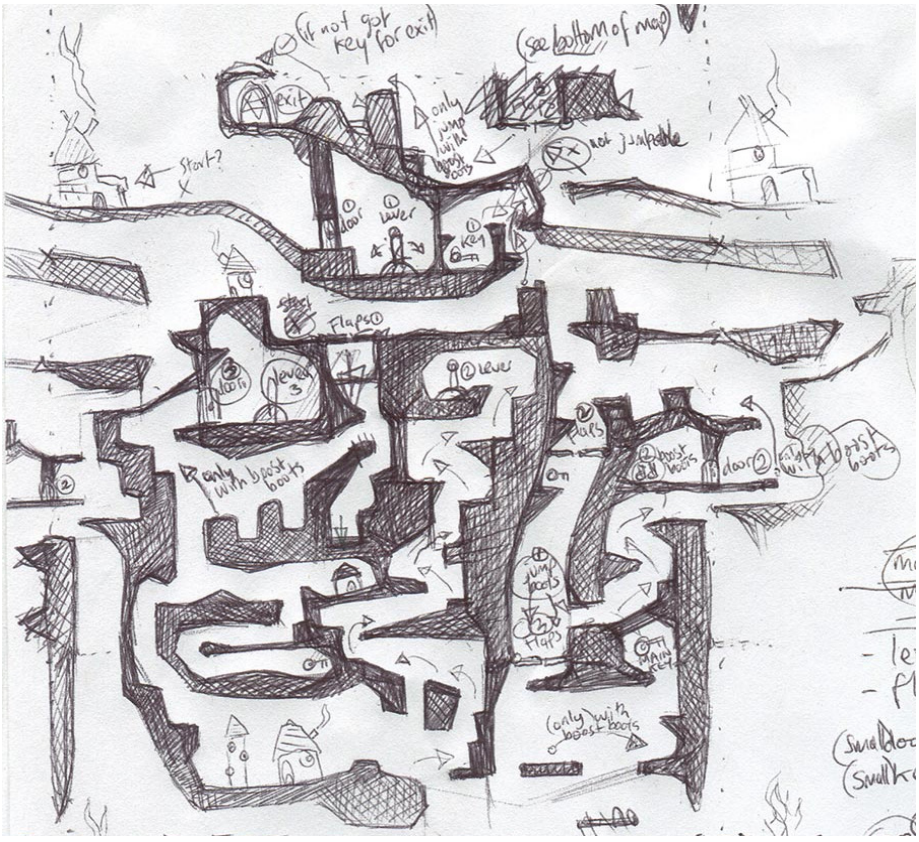
2D ゲームが注目を集めています。ハードウェア、グラフィックス、そしてゲーム開発ソフトウェアの進化により、リアルタイムライト、高解像度のテクスチャ、ほぼ無制限のスプライトを持つ 2D ゲームを作成することが可能になりました。

商用 2D ゲームを制作したい開発者やアーティスト向けに作成された、Unity の最も包括的な 2D 開発ガイドをお読みください。

eBook をダウンロード



ペーパーデザイン



紙に描かれた 2D プラットフォームゲームのレベル
ソース：Peter Mcclory

Unity での作業に入る前にアナログでレベルのザインを行うことは、レベル制作のリサーチ段階で役立ちます。

レベルのラフスケッチやレイアウトには、パズル、敵との遭遇、プレイヤーが遭遇するその他のゲームプレイ要素を含めることができます。

アナログでのデザインを通して、ゲームプレイアイデアや要素配置の潜在的な問題と利点の両方を想定することができます。特に、ゲームエンジン内にブロックアウトおよびテスト用のキャラクターコントローラーやレベルエディターがない場合、この手法が有効です。

一方で、すべてのレベルデザイナーがアナログの工程を取り入れているわけではありません。中には、視覚化したり空間内を動き回ったりすることができると言う理由で、始めから任意のゲームエンジン内で作業するやり方を好むデザイナーもいます。

スケッチ用アプリケーションは、Diagrams.net、Microsoft Visio、yED、Gravit Designer など、様々なものが公開されています。

Hugues Barlet 氏により LEGO でブロックアウトされたレベル
ソース：https://www.gamedeveloper.com/design/block-design-in-level-design

様々なゲームメカニクスを特定、理解する

プレイヤーが使用するゲームデザインメカニクスやゲームプレイは、デザイナーのツールキットの中でも重要な部分です。それらを学習し、理解が深まるほど、実際に使用する際により大きな効果を期待できます。

例えば、ジャンプメカニクスは、実装方法に応じて複数のパーツに分解できます。

最もシンプルな方法は、他のボタンは使わずにジャンプボタンだけを押して上にジャンプする静止ジャンプです。より複雑な動きには、全力で走る、ジャンプボタンを長押しして最大の高さまでジャンプする、壁ジャンプを行うなどがあります。

ジャンプメカニズムが持つ複雑さを理解できたら、最終的な結果を分解し、各メカニズムを

徐々にプレイヤーに紹介します。操作が最も簡単なアクションから始め、ゲームの進行に合わせてより複雑なインタラクションを導入していきましょう。

壁ジャンプを例にとり、以下のセグメントに分解してみます。

— ボタンを押してジャンプ（シンプルな上方向へのジャンプ、Y 軸の高さを得る）

— ボタンを長押ししてジャンプ（より高い上方向へのジャンプ、Y 軸の高さ + 追加の高さを得る）

— 走ってジャンプ（Y 軸の高さと X 軸の距離を得る）

— 走ってジャンプ長押し（Y 軸の高さ + 追加の高さと X 軸の距離を得る）

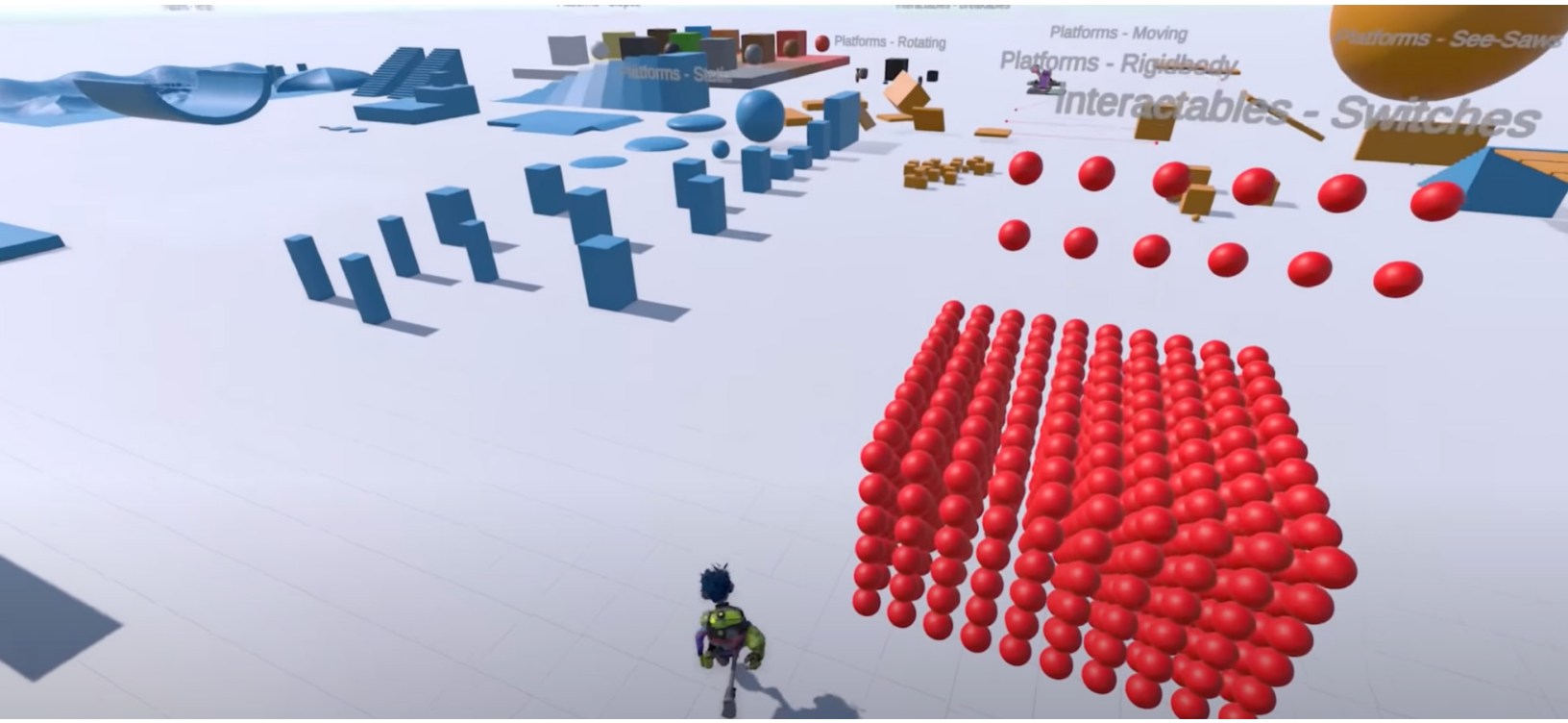
— 壁にジャンプして滑り落ちる

— 壁にジャンプして滑り落ち、別の壁にまたジャンプするなど

ゲームの進行に合わせて各セグメントを導入することで、学習の負担が少なくなり、プレイヤーが各能力を覚えやすくなることを期待できます。

『Indivisible』は、ゲームのレベル移動のメカニクスを徐々にプレイヤーに紹介する方法を上手く取り入れたゲームです。シンプルなジャンプから始まり、最終的には複数のメカニクスの組み合わせにまで至ります。プレイヤーが圧倒されることなく、スキルを覚えてマスターできるだけの十分な余裕が設けられています。

ゲームメカニクスをトレーニング環境 / ジムでテストする



扱いにくいプロトタイプの要素と障害物を使用して作成されたジムレベル

トレーニング環境 / ジムは、最終バージョンで使用されるメカニクスをテストおよび改善する目的で作成されたシンプルなゲーム空間です。通常、ジムは制作チームによって制作過程でのみ使用されます。

トレーニング環境でテストされるメカニクスには、以下のようなものがあります。

— 出入り口を作成して、カメラの高さとプレイヤーの左右に必要な空間をテストする
— プレイヤーの動きとコントロールをテストするために、方向転換や曲がり角、曲がりくねった道を追加する
— 傾斜をテストするために斜面を追加し、その際、複数の傾斜を含めて粒度をテストする
— 登れない斜面をテストして、通行可能な地形と通行不可能な地形のバランスを調整する
— トリガーやジャンプ距離、その他様々な要素をテストする

最終的には、効果的なメカニズムやゲームプレイ要素と、逆に省くべき要素を特定することが目標となります。

必要であれば、複数のトレーニング環境を作成し、地形タイプ、シューティングギャラリー、バトルメカニクスなどのメカニクスに区分けしてテストします。

開発チームとトレーニング環境を共有しましょう。例えば、QA チームはメカニクスのテストと区分け、ゲームデザイナーはデザインの調整、改善、テスト、テクニカルアーティストはアートアセット（LOD など）のテストのためにトレーニング環境を使用できます。

Unity は、メカニクスのテスト用に代用品として使用できるキャラクターコントローラーを提供しています。Unity のキャラクターコントローラーを新しく置き換える場合は、新しいコントローラーをトレーニング環境で再テストして、以前のコントローラーのゲーム指標や機能性に適合するか確認し、必要に応じて調整する必要があります。詳しくは、キャラクターコントローラーのセクションをご確認ください。

ペースとゲームプレイの展開

優れたペースはプレイヤーを飽きさせません。これはゲームプレイの「展開」、つまりプレイヤーにゲームを進めさせるゲーム内の主要なポイントやアクティビティによって支えられています。

レベルデザイナーは通常、ゲーム全体のテンポをコントロールします。視覚的なタイムラインを作成することは、どのようなテンポでプレイヤーにゲームをプレイして欲しいか、そしてどのタイミングで新たなゲームプレイの展開が発生するかを決定するのに役立ちます。

あるレベルやゲームのセクション内の時間制限をなくすことで、その特定のパートを自由に探索する時間をプレイヤーに与えましょう。逆に、プレイヤーに緊迫感を与える必要がある場合は、時間制限を設けてプレイヤーを躍起にさせましょう。

激しめのゲームでは、プレイヤーが本格的にプレイを再開する前にクールダウンできるよう小休止時間を与えてみると良いかもしれません。『ストリートファイター II』の「車破壊」ステージが良い例です。AI や他のプレイヤーとの激しい戦闘の後、反撃してこない車を破壊することで、プレイヤーは技を練習し、自信を取り戻すことができます。

Endnight Games の『Sons of the Forest』（発売元：Newnight）は、戦闘の合間に提供される息抜きのタスクとして、クラフティングをメインのゲームメカニクスの 1 つに持つ人気の高いサバイバルゲームです。

ペースについての詳細は、Brackeys によるこちらの動画をご覧ください。

生きた世界：環境ストーリーテリング

Slow Bros の『Harold Halibut』は、Unity Cinemachine と Timeline をうまく活用した、友情をめぐる手作り風のストーリーゲームです。

薄暗い空間にいる 1 人の男が壁のグラフィティを見つめています。赤い文字で「WHERE' S HOME?（家はどこだ?）」と書かれています。一見したところ、血で書かれているかのように見えますが、赤いペンキの缶は、キャラクターの（そしてプレイヤーの）目を引きます。このシーンはプレイヤーに、その意味を想像しながら、目の前の光景を解釈するよう促します。このグラフィティやペイントのようなシンプルな環境要素は、プレイヤーがゲームのストーリー世界に没入することを促します。

環境ストーリーテリングは、レベルデザインや環境アセットを通してストーリーを語るというコンセプトです。明示的ではなく、暗示的なマーカーや手がかりなどを用いて、ゲーム内で繰り広げられる出来事を説明します。

強力なワールドビルディングは、ゲームのストーリーをサポートするために環境ストーリーテリングを使用し、ゲームの世界を本物のように感じさせ、プレイヤーの没入を促します。レベルデザイナーは、ゲームの進行に合わせてストーリーが明かされるよう、ライティングを工夫し、テーマに沿ったアセットを使用し、プロップの配置に注意を払うとよいでしょう。

以下のような基本的な質問に答えることで、ゲームのストーリーテリングを明確にしましょう。

— **物語の背景：**プレイヤーがある場面に至った理由を示唆するために含められそうな要素はありますか？プレイヤーを特定のポイントやマイルストーンに導くクエスト進行 / ゲームプレイを考え、シーンおよびレベルのトーンや雰囲気の設定に使用しましょう。

— **出来事：**シーン内に、物語の出来事をプレイヤーに理解させるためのトーンや要素の描写はありますか？

— **舞台：**舞台とそのバックストーリー、また、舞台設定をサポートするために含めるべき要素や焦点を当てるべき要素を考えましょう。

— **時間軸：**ゲームワールドのタイムラインと時代を考えましょう。時代を設定することは、デザイナーが時間軸の中に身を置き、ストーリーをより良く伝えるために含めるべき要素を見出す助けとなります。

— **経緯：**地震が起きて、街中がパニックになったとします。倒れた写真、ひびの入った建物などの背景要素を使って、どのようにその状況をプレイヤーに伝えられるか考えましょう。

これらの質問に答えることで、レベルを構築する際の不足情報を補い、筋の通ったバックストーリーに基づいた、一貫性のある多角的なワールドを構築しやすくなります。

制作

プリプロダクションが終わると、レベルのブロックアウトとイテレーションを開始するのに必要なツールと情報が揃うはずです。このセクションでは、レベル構築のヒントとベストプラクティスをご紹介します。

ホワイトボクシング：レベルのブロックアウト

ホワイトボクシング（グレーボクシングとも呼ばれる）とは、レベルデザインとスタイルのビジョンに最も適したレイアウトを特定するために、単純な 3D 形状を作成して配置することです。

ホワイトボクシングをシンプルに保つことで、アート、ライティング、その他のディテールを調整することなく、レベルを簡単に操作できるようになり、迅速なイテレーションプロセスを実現できます。

レベルデザイナーの中には、3D 空間で作業してゲームエンジンのワークフローに慣れるために、最初に紙のデザインを作成せず、直接ホワイトボックス作業に入ることを好む人もいます。デザイナーは、自分に合った最適なアプローチを自由に決められます。

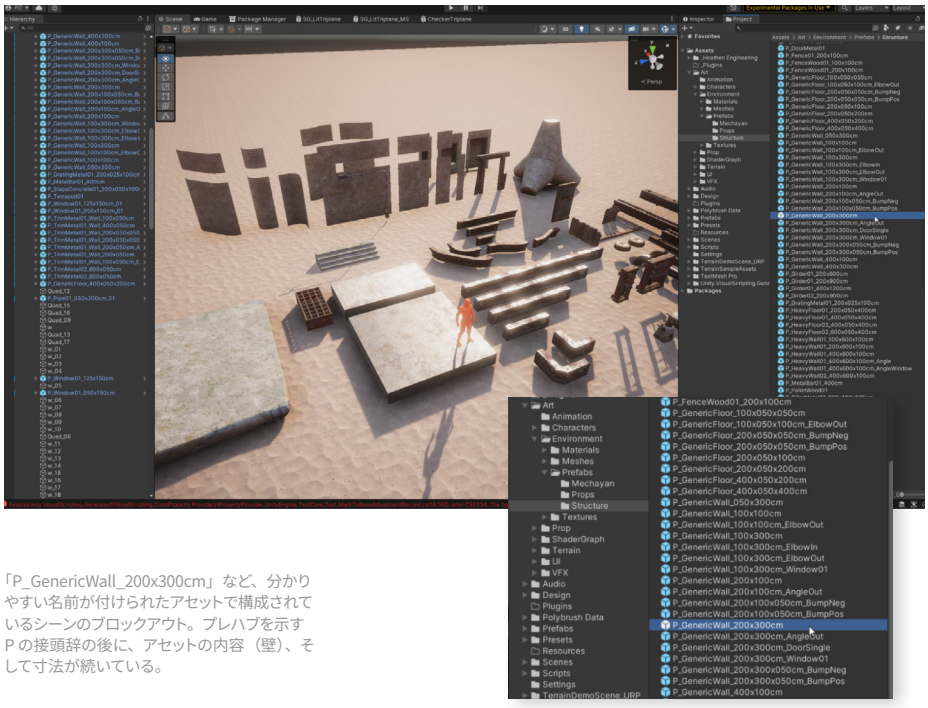
このガイドのパート III では、Unity の ProBuilder を使用して効率的に形状と地形を生成する方法をステップバイステップで説明します。ProBuilder は、ゲームデザインでよく使われる既製のモデリングツールと定義済みの要素を提供します。また、パート II で扱っているように、プリミティブ 3D オブジェクトを Scene ビューに直接配置することもできます。

Multiple weapons tdm AI VR shooter d18 – reddit（投稿者：niv-vada）より、ProBuilder を使用したブロックアウトレベルの例
Reddit のスレッド

ブロック状のアセットに説明的な名前を付ける

各ブロック状のアセットには、その用途を特定できるよう、説明的な名前を付けましょう。これにより、環境アーティストが、実際のゲームアセットへの置き換えを行う際に、それぞれのオブジェクトにどのような意図があるのかを理解しやすくなります。

例えば、ブロックは壁を表しているのか、その場合、プレイヤーの視界を遮るために最低限の高さが必要になるか、といったことです。このアセットは、「wall_interior_w2_h4_l6」と名付けるのが良いかもしれません。 このラベルからは、アーティストに伝えるべき重要な詳細であるオブジェクト、その位置、そして寸法を識別できます。また、名前の書式を統一することで、同僚がその意味を理解しやすくなります。



「P_GenericWall_200x300cm」など、分かりやすい名前が付けられたアセットで構成されているシーンのブロックアウト。プレハブを示す P の接頭辞の後に、アセットの内容（壁）、そして寸法が続いている。

利便性のため、3D 空間内にフローティングテキストを使用して意図を示すこともできます。これにより、同僚がエディターにアクセスしなくても、ビルドを実行してプランを見ることができます。

最後に、アセットをラベル付けすることで、作業中のタスク管理が容易になるため、シーンを見る他の人だけでなく、自分自身の役にも立ちます。

意図を視覚化するため、ブロック状のアセットにマテリアルを追加する

アセットに説明的な名前を付けるだけでなく、ブロックにマテリアルを適用して意図を明確にすることもできます。これは、インタラクティブなオブジェクトと静的なオブジェクト、プレイできる空間とできない空間、破壊できるものとできないものなどを区別するための便利な方法です。

詳細なアートアセットの作成は避ける

> 「3D ステージでは、コンセプトやモデルに対して支払いを行う必要が出てきます。我々にとっては、レベルを構築し、細部まで作り上げるという最も時間がかかる箇所でした。どんなレベルにしたいかが明確になるまで、この段階に進むのは避けるべきです。」
>
> David Fenn 氏、『Death’s Door』のコンポーザー、サウンドデザイナー、プロデューサー、レベルデザイナー

ホワイトボクシングの意義は、アイデアを試してイテレーションを回すことにあります。最終決定されていないアセットの変更に時間を費やすのを避けるため、デザインの承認が下りるまで、チームは 3D アセットを作成すべきではありません。

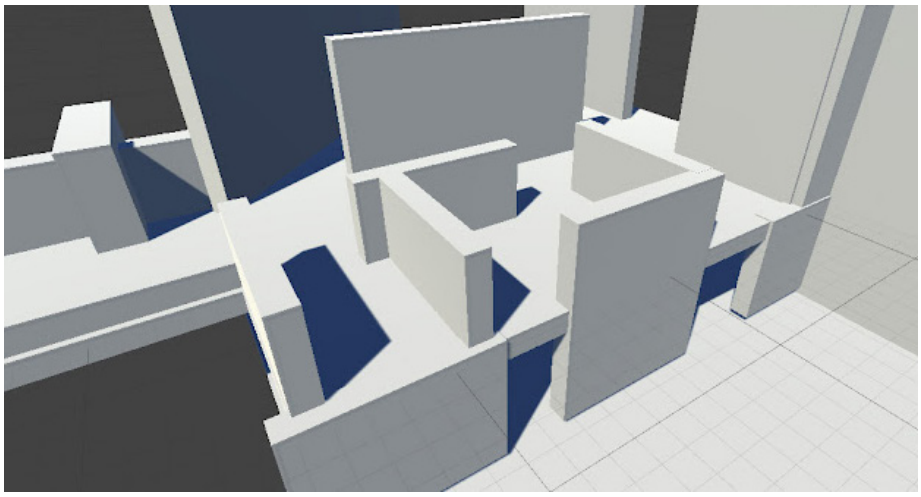
他の関連チームのメンバーからの賛同を得て、満足のいくレベルデザインができた後、アーティストはレベル全体で、3D アセットを追加してホワイトボックスの置き換え作業を開始することができます。

コンセプトアーティストに助けを求める

チームメンバーは、ホワイトボックスシーンから最終的なシーンを想像するのに苦戦するかもしれません。コンセプトアーティストは、ホワイトボックスにペイントオーバーを行うことで、ビジョンの具体化を手助けしてくれます。これはレベルの 2D 表現であり、3D アセットの作成は必要ありません。一般的に、ペイントオーバーは、レベルのスクリーンショットを撮り、それをペイントして希望のスタイルやムードを表現します。

実際のシーンをコンセプトアーティストと共有しても良いでしょう。それぞれのブロックが何を表しているかをドキュメント化しておけば、コンセプトアーティストが全体的なビジョンに沿った表現をしやすくなります。

コンセプトアーティストが制約とブロックアウトを考慮して作業すれば、ペイントオーバーはチームからの賛同を得る良い方法となるでしょう。



画像ソース：How to graybox, blockout 3D video game、FMPONE と Volcano の『de_crown』- ペイントオーバー前のブロックアウト

画像ソース：How to graybox, blockout 3D video game、FMPONE と Volcano の『de_crown』- ブロックアウトのペイントオーバー

既製アセットを活用する

ホワイトボックスシーンを豊かにするもう 1 つの効率的な方法は、既製のアセットを使用することです。

Unity Asset Store には、アイデアを視覚化するために、すぐに使えるアセットが多数用意されています。例えば、Synty Studios の POLYGON Prototype Pack は、「組み込まれたメモとマーカー機能を使って、デザインの決定をチームに伝える」のに役立ちます。レベルデザインで特定の Unity Asset Store 製品を使用する方法の詳細は、このガイドのパート III を参照してください。

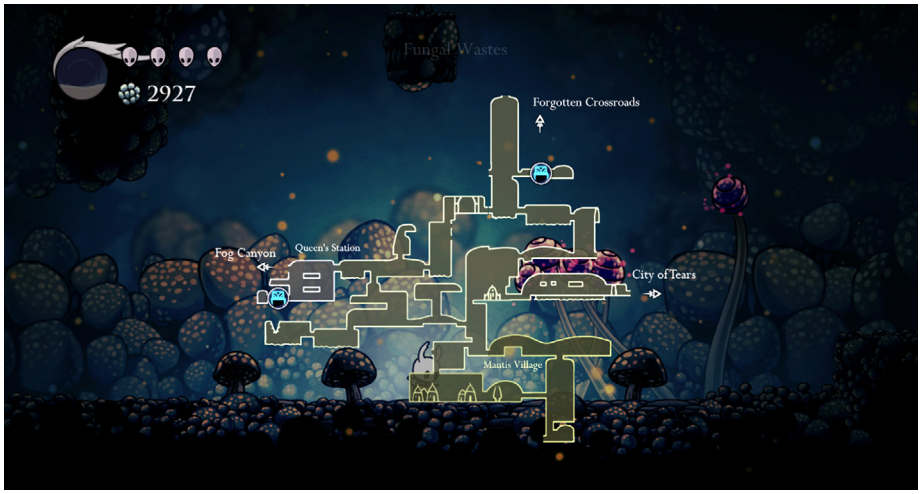
Synty Studios の POLYGON Prototype Pack は Unity Asset Store で入手可能

プレイヤーの動線

プレイヤーの動線とは、プレイヤーがゴールを達成するために通る経路（パス）を指します。ゲーム開発におけるパスには、クリティカル、ゴールデン、二次経路、および三次経路の 4 つのカテゴリがあります。

クリティカルパス

クリティカルパスは、ゲーム完了までの最長経路です。リリースするすべてのコンテンツを効率的にテストするために通る必要がある経路だと考えてください。



Team Cherry の『Hollow Knight』は、Unity で制作された人気の 2D ゲームです。本ゲームのクリティカルパスは、ゲームの 112% をクリアするのに必要なパスで、これはマップ全体を指します。

ゴールデンパス

ゴールデンパスには、最高のゲームプレイ要素、報酬や隠し要素が含まれます。中には、TUNIC Team の『TUNIC』のように、複数のエンディングが用意されているゲームもあり、最高のエンディングにつながる経路がゴールデンパスとみなされます。

TUNIC Team の『TUNIC』は、複数のエンディングが用意されている、アイソメトリックアクションアドベンチャーゲームです。

ゴールデンパスは、プレイヤーにとって最も魅力的で楽しい経路となるべきです。必ず同僚とテストを実施し、フラストレーションや混乱を招くポイント、またコンテンツの魅力を感じられる部分を特定するようにしてください。

通常ゴールデンパスは、プレイシステムのすべてではないにせよ、その大部分を網羅しているため、プレイヤーはゲームが提供する主な魅力をすべて体験できます。ただし、ゲームのすべてのコンポーネントを含むわけではありません。例えば、複数のエンディングがあるゲームの場合、クリティカルパスに沿って各エンディングをプレイする必要がありますが、ゴールデンパスは 1 つのエンディングに到達するまでの最も魅力的で、面白く、楽しい唯一の経路となります。

二次経路と三次経路：サイドクエスト、隠し要素、ショートカット

何種類のパスが必要でしょうか？ゲームのプレイスタイルごとに経路を提供することが理想です。例えば、Eidos Montreal 氏の『Deus Ex: Human Revolution』では、ラン&ガン、ステルス、ハッキングなど、複数のプレイスタイルがあります。最適なプレイスタイルを特定し、デザイナーがそれを念頭に置いてレベルを構築できるようにしましょう。さらに、各スタイル、またはその組み合わせに基づく問題を解決するための経路を複数用意します。

少ない報酬または何の報酬もなく、プレイヤーをすでに探検したエリアに引き返させることは、フラストレーションの原因となります。ショートカットやサイドクエストの終点に到達したら、すぐにメインストーリーに戻れるようにすることがより優れたプレイヤー体験に繋がります。

さらに、たとえ大したものでなくても、道の端やオープンエンドの空間にコンテンツを追加することは、探検したプレイヤーに報酬を与え、ゲーム全体を通してプレイヤーの探検を促すことができます。

Obsidian Entertainment の『Pillars of Eternity』は、進む道などの選択により、プレイヤーの運命が左右される魔法の世界を舞台としています。上の画像は、メインクエストの他にサイドクエストの選択を示しています。

プレイヤーにメカニクスを教えるための 3 の法則

プレイセッション中に何度も繰り返すことで、プレイヤーに新しいメカニズムやシステムを学ぶ時間を与えましょう。一般的なルールとして、プレイヤーが新しいアクションや一連のアクションに慣れるまで、少なくとも 3 回実行する必要があります。プレイヤーに過度の負担をかけないように、新しいシステムや仕組みは間隔を空けて導入しましょう。

典型的なプラットフォームゲームで、新しい仕組みを導入する際に「3 の法則」を使用する例を見てみましょう。

1. プレイヤーが初めて 1 体の敵に遭遇し、上に飛び乗ることで敵を倒せることを学びます。
2. 最初の遭遇に続き、複数の敵が登場し、ここでも敵の上に 1 回ずつ飛び乗ることで倒せます。これにより、敵の頭の上に飛び乗って倒すというアクションがさらに定着します。
3. 3 回目では、プレイヤーは間隔をあけて敵の集団に遭遇します。それぞれの敵の上に飛び乗ることで倒せますが、今回は器用な操作が求められます。

この 3 回の遭遇を経て、プレイヤーは 1 度飛び乗れば複数の敵も 1 体の敵も倒せることを学ぶはずです。

「様々なゲームメカニクスを特定、理解する」で説明したように、新しいメカニクスやシステムを学習しやすいチャンクに分割しましょう。

Team17 の『Overcooked 2』は、ゲームプレイを段階的に導入していて、簡単なアクションから、徐々により手の込んだレシピへと進んでいきます。

期待を裏切る

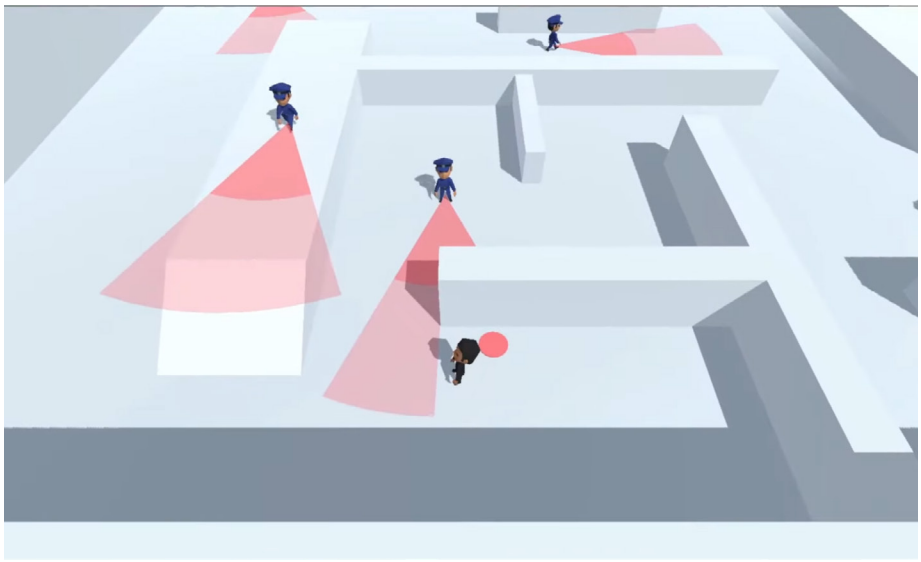
プレイヤーの心にパターンを植え付けたら、次はそれを覆しましょう！新しいメカニクスや既存のメカニクスの拡張に基づく予期せぬチャレンジは、ゲームをさらに面白くし、プレイヤーを惹きつけることができます。

先ほどの例では、プレイヤーは上に飛び乗ることで敵を倒せることを学びました。

今度は、プレイヤーはヘルメットを被った敵に遭遇します。反射的に敵の上に飛び乗り、これで倒せるだろうと期待しますが、ヘルメットだけが破壊され、敵は生き残ります。プレイヤーは、今度はヘルメットを被っていない状態の敵に飛びかかり、2 回目のジャンプで敵を倒せることを学びます。プレイヤーの期待は裏切られましたが、敵の上に飛び乗るというおなじみのパターンが使えるため、フラストレーションを感じさせることはありません。

Second Dinner の『Marvel SNAP!』では、ゲーム全体でスリリングな展開をもたらす能力を持ったカードが登場します。同チームの GDC23 セッションでも紹介されましたが、Unity でどのようにこのモバイルヒットが制作されたかをご覧ください。

見通しと死角



画像：Indie Marc の Enemy Vision 2 – Noise detection, two levels cones, code restructuring

見通しは、視覚軸または視線とも呼ばれ、観察者の目線と注目の対象、またはその相対的な方向を結ぶ架空の線です。

見通しは、ビデオゲームで敵やプレイヤーを検出するなど、プレイヤーの視線の外側にあるオブジェクトをカリング処理するためによく使用されます。原点を始点とする線の配列からなる円錐状の形で表されます。

例えば、FPS ゲームでは、レベルデザイナーは見通しと死角に注意を払います。レベル全体でプレイヤーが見える範囲や見られる範囲を把握することは、銃撃戦が発生する場所や、状況に対処するのに十分なカバーがあるかどうかなど、プレイヤーのための課題を作成する上で重要です。また、見通しの良い場所を使用することで、プレイヤーに安全なルートを探させることができます。

視線とは対照的に、プレイヤーや AI からはすぐに見えない場所が死角です。死角を利用して、プレイヤーの視界の外に敵を出現させたり、隠しオブジェクトを配置したり、強烈なステルスの瞬間を作り出しましょう。

Unity ではレイキャストや物理演算で視線や同様のシステムを作成することができます。これについては、パート II で説明しています。

プレイヤーの注意を引く

多くのゲームでは、プレイヤーを誘導して特定の空間を探索するよう促しつつ、一部の空間は避けるよう促すことが望ましくあります。プレイヤーの注意と動きを誘導するために使えるテクニックを確認しましょう。

Playdead によるインディーアドベンチャーゲーム『INSIDE』では、画面上の道や注目ポイントにプレイヤーの注意を引くため、至る所でライトが使われています。

ライティングと空間

プレイアブル空間のライティングは、プレイヤーに動きを指示し、シーンの雰囲気を作り出すのに非常に役立ちます。

トンネルの出口の光など、特定のパスを強調するライティングを用いて、プレイヤーを誘導しましょう。光を消すことも効果的です。薄暗い空間や暗闇の空間はプレイヤーの恐怖心を煽り、潜んでいる敵から遠ざけることができます。

物理ブロッカー

物理ブロッカーは、プレイヤーの動きをブロックする衝突のことです。物理ブロッカーを使用して、プレイヤーをプレイアブル空間に留めましょう。例えば、プレイアブル空間の境界を決める目に見えない衝突に視覚的なキューを加えるなどして、ブロッカーを明確にマークまたは示唆します。

プレイヤーをプレイアブル空間に留めるために使える方法には、以下のようなものがあります。

— プレイヤーが接触すると死ぬ「即死」フロアで囲まれているまたは密閉されている宙吊りのプレイアブル空間

— プレイヤーが利用できるリソースを制限し、アンカーポイントから離れ過ぎた空間を探索できないようにする

— 見えない境界線を越えると、間もなく死亡するか元の場所に戻るという内容の通知やメッセージが表示される（通常バトルロイヤルゲームで使用される）

もちろん、プレイアブル空間内にプレイヤーを留める方法は沢山あります。ゲームに合ったシステムを探して活用しましょう。

No Matter Studios の『Praey for the Gods』は、ボスの打倒を目指すオープンワールドアドベンチャーゲームで、プレイヤーは滅びゆく凍てつく大地の果てに送り込まれた孤独なヒーローとなり、終わらない冬の秘密を探り解決します。ゲーム全体を通して、プレイヤーが世界にスポーンした際、物理的なブロッカーや道標がプレイヤーの進行方向を示します。

案内表示

案内表示とは、プレイヤーが各レベルを理解、移動、進行するのに役立つ手がかりや標識、その他のコンテンツのことです。案内表示は、どこに向かえば良いのかを知らせ、道に迷ったときに頼りになる手がかりや方向を教えてくれます。

地図や信号、標識、ドアの「押す」「引く」のプラカードなど、現実世界が案内表示で溢れているように、ゲームでもプレイヤーが自分のいる世界を理解するための手がかりやメッセージを十分に提供する必要があります。さりげない案内表示と、遠くからでも見やすく認識しやすい目印のようなわかりやすいものの両方を使用しましょう。

グローバルオーディエンスが容易にその意味を理解できる場合は、色を案内表示に使用することもできます。例えば、赤で「止まれ / 危険」、緑で「進め」を表したりする場合です。多くのプレイヤーは、真っ赤な樽が爆発などの危険を示していることを知っているでしょう。

音

音も、プレイヤーの注意を引く優れた方法です。遠くの銃声、プレイヤーに前方に進むよう促す声、暗闇から聞こえる恐怖心を煽るような何かが軋む音など、音はプレイヤーの体験を豊かにし、ゲームのメカニクスや行動を強化するのに役立ちます。

『Subnautica』では、音を使ってプレイヤーのステータスを理解しやすくしています。例えば、プレイヤーの酸素が少なくなると、音楽がより激しく不明瞭になり、プレイヤーの口から泡が出るのが見えたり聞こえたりして、最後にはすべてが暗転します。

聴覚障がいのあるプレイヤーにも楽しんでもらえるよう、音声と視覚の両方を取り入れましょう。『Subnautica』では、UI のテキストメッセージといった視覚的な手がかりが、プレイヤーのステータスを示す音声の手がかりと一緒に表示されます。

Unknown Worlds Entertainment の『Subnautica』では、プレイヤーは住居を作って、道具をクラフトし、水中世界に深く潜って生存を試みます。この画像では、体力のステータス、UI テキスト、サウンドエフェクトによって、プレイヤーが酸素不足になりかけていることが示されています。

スポーンポイント

プレイヤーをワールドにスポーンさせることで、ゲーム体験のトーンを設定し、プレイヤーを意図した方向に誘導することができます。進行方向を向いた状態でプレイヤーをスポーンさせるのは良い方法です。これにより、プレイヤーはすでに通った道に戻る必要はありません。

スポーンの瞬間を利用して、重要な要素にカメラをパンさせます。これは、プレイヤーにコントロールを返す前に、要素の重要性を知らせるためです。

セーブポイントとチェックポイント

通常、レベルデザイナーによって配置されるセーブポイントとチェックポイントは、プレイヤーをゲームに没頭させ続ける助けとなります。進行状況をセーブすることで、最初からやり直すことを恐れずに、色々なことを試すようプレイヤーを促すことができます。

Team Cherry の『Hollow Knight』では、セーブポイントとベンチが世界とストーリーテリングの一部になっています。

セーブポイント

一部のゲームでは、セーブポイントが固定されていることがあり、プレイヤーは手動でセーブポイントを探す必要があります。静的なスポーンポイントは、デザイナーがいつ、どこで何をセーブするかをコントロールでき、任意の場所でゲームをセーブすることがシステムにとって難しい場合に便利です。

他のゲームでは、プレイヤーはゲームプレイ中にほぼどの場所でも、または一定の制限付きでセーブを行えます。通常、アクションを完了した後に任意の場所でセーブをリロードすることは複雑になる可能性があるため、これはより多くの作業と堅牢なゲームコードを要します。

多くのゲームは両方のタイプのセーブポイントを使用しています。例えば、任意の場所でセーブできるものの、すべての敵がリセットされ、セーブした場所ではなく、デザイナーが設定した静的な場所にスポーンされるゲームもあるでしょう。この複合型のアプローチは、複雑なゲームとの相性が良く、任意のタイミングでのセーブが可能な一方、セーブシステムをシンプルに保ち、潜在的なバグや問題を最小限に抑えることができます。

チェックポイント

チェックポイントは、レベル全体に配置されます。一時的にメモリに保存され、プレイヤーがミッションやレベルを進めている間のみアクセスできます。

このテクニックは、デザイナーが長いミッションの中で一時的なセーブを設定し、それを小さなセクションに分割することで、プレイヤーのゲーム進行を容易にします。チェックポイントは、一連の流れの中に複数のチャレンジがある場合や、ミッションが複雑すぎて任意のポイントでリロードできない場合にも便利です。

ソフトロックの回避

プレイヤーのソフトロックを避けるため、ゲーム内のすべてのチェックポイントをテストしましょう。ソフトロックとは、プレイヤーがゲームを進行したり後戻りしたりできなくなることを指し、多くの場合、以前のセーブデータをロードするか、最初からプレイをやり直すことになります。

ソフトロックの例をいくつか挙げます。

— プレイヤーが衝突に巻き込まれ、脱出できなくなった場合

— プレイヤーが装備を失い、進むことも後戻りもできない地点でチェックポイントがトリガーされる

— 死を回避できない地点でセーブがリロードされる

複数のオートセーブスロットがあるゲームでは、プレイヤーはゲームプレイの様々な場所で、手動でセーブすることができ、進行をロールバックしてソフトロックを回避できます。

Free Lives の『Broforce』は、横スクロールのラン&ガンプラットフォームゲームで、セーブポイントとチェックポイントをスマートに使い、プレイヤーがゲームに熱中した状態を保ちながらゲームプレイをセグメント化しています。

セーブポイントやチェックポイントを慎重に配置

プレイヤーのフラストレーションにならないよう、ゲーム中のセーブポイントやチェックポイントの配置には注意が必要です。例えば、難しいミッションのセーブポイントを、スキップできない長いカットシーンの前に配置することは避けましょう。レベルの攻略に失敗するたびにカットシーンを見せられると、プレイヤーはすぐにフラストレーションを感じてしまいます。長いカットシーンの後にセーブポイントやチェックポイントを置いて、プレイヤーがすぐにアクションに戻れるようにしましょう。あるいは、プレイヤーがカットシーンをスキップできるようにしましょう。

これは、難易度の高い連続チャレンジがあるゲームプレイセクションにも該当します。例えば、プレイヤーが難しいプラットフォームセクションをクリアした後、大きなボス戦に挑む場合、セーブポイントやチェックポイントを挟んでチャレンジを分けた方が、より良いゲームプレイ体験につながるでしょう。

プロシージャルなレベルデザイン

プロシージャル生成は、ゲームにおいて大量のコンテンツの作成を容易にする強力なツールです。プロシージャル生成を用いることで、開発時間を節約したり、プロジェクト規模を小さくしたり、ランダムなレイアウトを生成して予測しにくいゲームプレイを実現したりすることができます。

Mechanistry　の『Timberborn』では、プロシージャル生成された地形がパズルや、この「始まりの村」のような、街の土台となるプレイグランドをプレイヤーに提供します。

ゲーム開発にプロシージャル生成を取り入れる方法はいくつかあり、その技術は急速な成長を遂げながら変化しています。どのプロシージャル生成のタイプが、現在取り組んでいるプロジェクトのタイプに最も適しているか調べてみましょう。Unity での手法の例については、タイルマップを活用したプロシージャルパターンに関するこちらのブログ記事をご覧ください。

プロシージャル生成は、アルゴリズムと与えられたコンテンツに基づいて結果を生成します。コンテンツは一般的に開発チームによって作成されますが、プレイヤーやコミュニティが利用できるレベル作成ツールによってユーザー自身が生成することもできます。プロシージャルコンテンツに GIS データを使用することもでき、クリエイターは現実世界の地理情報からデータを取得し、プレイアブルレベルとしてゲームにインポートできます。

プロシージャル生成のルール

プロシージャル生成は、レベルにランダム性を与えるのに適していますが、理解しにくく、フラストレーションを感じさせるレイアウトになることもあります。理解しやすく、導入しやすいルールがプレイヤーにとって最も効果的です。優れたガイドラインは、ランダム性を保ちつつ、プレイヤーが望む結果を出力するのに役立ちます。しかし、効果的な変更を行うためには、プロシージャル生成された要素間の関係を理解する必要があります。

プレイヤーが生成されたコンテンツを探索する際の道標となるような、独特または認識しやすい構造物の周りに、プロシージャル生成されたコンテンツを構築すると良いでしょう。これらのタイプのランドマークは、視覚的なナビゲーションツールとして機能し、システム生成による単調さを解消するのに役立ちます。

自動テスト

プロシージャル生成では、大量の結果が生成されることが多いため、時間と労力を節約するために自動テストを設定するとよいでしょう。

自動テストを設定し、数百回のイテレーションを通してプロシージャルレベルの生成を実行しましょう。これは、使用したルールが意図したとおりに動作しているか、変更が必要かどうかをテストするためです。コンピューターはユーザーの入力なしでテストを実行するため、短時間で多くのレベルのイテレーションを生成することができます。手元にデータがあれば、ルールに必要な調整を行うことができます。

テストの反復

コンテンツをテストすることは非常に重要です（このガイドを通して何度も言及しているのはそのためです）。

レベルのプレイ体験はどのようなものですか？デザインの長所と短所は何ですか？それらをどのように活用および改善できますか？コンテンツを最後までプレイする（チームにもやらせる）ことで、ユーザビリティの問題、バグ、ソフトロックを QA 段階前に特定することができます。

コンテンツを設計する際に、どのようなツールを使えばテストをより効率的に行えるかを考えましょう。自分のキャラクターと他のプレイヤーとの距離を知る必要がありますか？それなら距離を表示するスクリプトを作成してみる価値があるかもしれません。マップ上で、プレイヤーが最も多く失敗している場所を知りたいですか？プレイヤーが失敗した場所を記録し、ヒートマップを作成しましょう。

ゲームの特定の側面のテストや、レベルに関するフィードバックが必要な場合、テストチームは優れた情報源になります。他にも、実施するテストの内容や、どのような種類のツールがテスト作業を効率化するかなどの情報を提供してくれるでしょう。

Iron Gate studio の『Valheim』は、プロシージャル生成されたワールドから成るゲームです。レベルのサイズは常に同じで、プレイヤーは中央からスタートします。

パート II：レベルデザイナー向け UNITY 入門

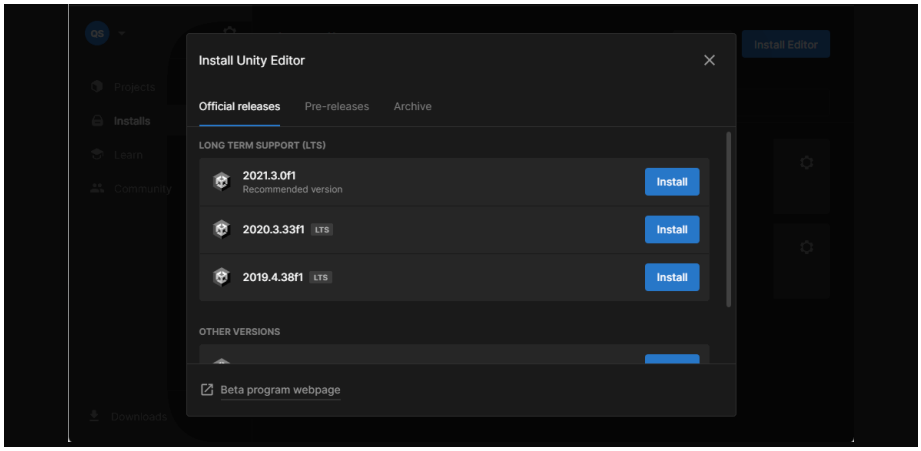
レベルデザイナーは、必ずしも Unity での制作経験があることを期待されているわけではありません。しかし、トレーニングや経験を通して、Unity を使い始めるのに必要な論理的スキルは備わっているはずです。

このセクションは、Unity のインストール、エディターインターフェースの理解、Unity で構築するシーンの基本要素の理解、アセットとプロジェクトの整理など、Unity で作業をするための入門編となります。Unity と Unity Asset Store 上のレベルデザインに特化したツールセットについて書かれた 3 番目の最後のセクションに入る前に、このセクションを読んでおくことをお勧めします。

Unity 初心者向けの最も包括的な学習教材は、Unity Learn で入手可能です。Unity のエキスパートが手掛けた Pathway コースは、最初に取り組むのに最適なリソースです。また、2D 初心者には Create with Code シリーズ、Ruby's Adventure、3D 初心者には John Lemon's Haunted Jaunt もお勧めです。短めのチュートリアルを希望の場合は、RPG、パズル、FPS ゲームの Creator Kit プロジェクトをお試しください。すべて 1 ～ 2 時間で完了できる長さになっています。

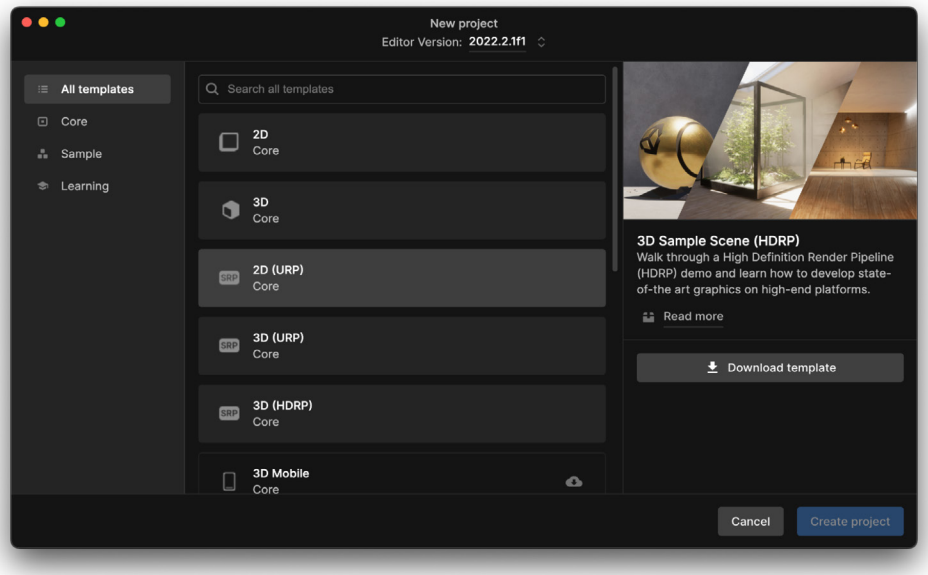
Unity のインストール

Unity にはダウンロードできる 2 つのバージョン、長期サポート（LTS）と TECH ストリームがあります。LTS エディションは、1 年間に実装された機能や改善を単一のインストールとしてまとめたデフォルトのリリースで、最大の安定性を提供します。TECH ストリームは、新機能への早期アクセスを提供し、主に開発のプリプロダクション、ディスカバリー、プロトタイピングのフェーズでの使用が推奨されます。Unity のリリースについての詳細は、こちらをご覧ください。



Unity のインストール、プロジェクト、モジュールは Unity Hub 上で管理できます。

最新の Unity LTS リリースのインストールが推奨されています。インストールは、ダウンロードウェブサイトから入手できる Unity Hub ランチャーで実行できます。



Unity Hub の「Projects」ページ

Unity Hub の「Projects」ページには、Unity プロジェクトが表示されます。「Projects」ページで新しいプロジェクトを作成したり、既存のプロジェクトを管理したり、プロジェクトを Unity エディターで開いたりできます。

「**New project**」を選択すると、複数のオプションが表示されますが、インストールされている Unity バージョンによって異なる場合があります。Unity 2022 LTS がインストールされている場合、リストに以下の空のテンプレートが表示されます。

— **2D**：古いバージョンの Unity ビルトインレンダーパイプラインを使用し、2D パッケージがプリロードされている 2D ゲーム開発用の設定

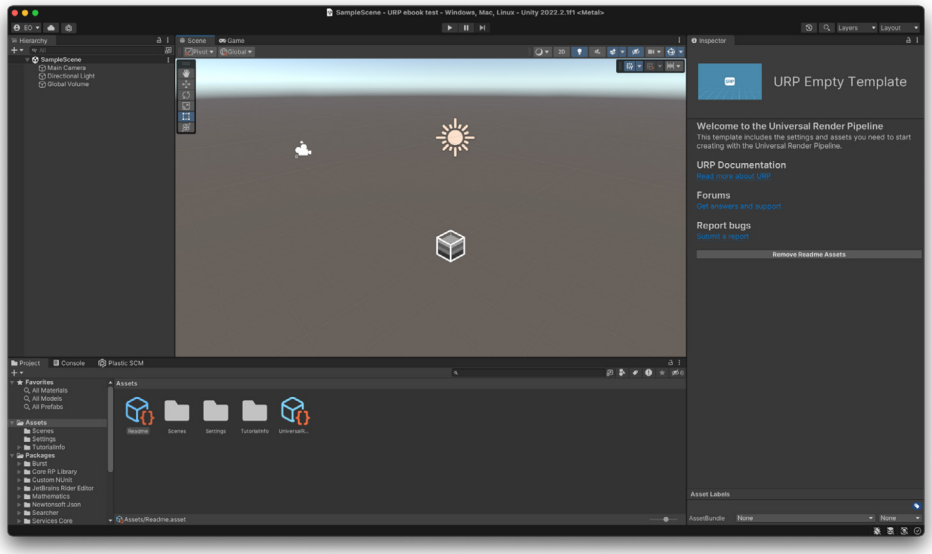
— **3D**：ビルトインレンダーパイプラインを使用している 3D プロジェクト用の設定

— **2D（URP）**：2D ライトやシャドウなどのアップデートされた 2D グラフィック機能を可能にするユニバーサルレンダーパイプライン（URP）を使用する新しい 2D テンプレート

— **3D（URP）**：URP を使用し、パフォーマンス、幅広いプラットフォーム、グラフィックスカスタマイゼーションへのサポートが事前に設定された 3D テンプレート

— **3D（HDRP）**：ハイエンドプラットフォームをターゲットとし、HD レンダーパイプライン（HDRP）が提供する機能を必要とするプロジェクトをサポートする設定が含まれる

3D モバイルゲーム用のテンプレートなど、ターゲットプラットフォーム固有の追加のテンプレートも Hub から入手できます。



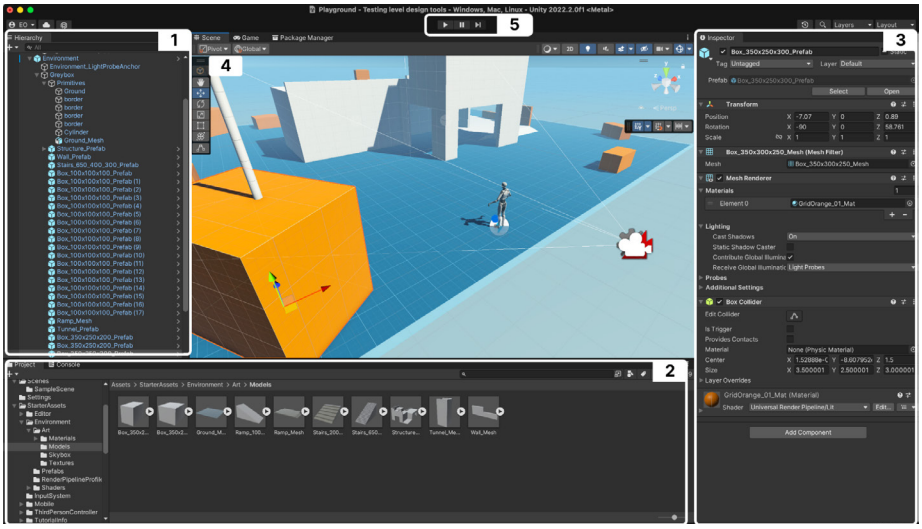
Unity で利用可能な多くの機能に慣れていない場合、空のプロジェクトからの作成は大変かもしれませんが、心配は無用です。独自のアセットをすぐに作成し始めてもよいですが、Unity Asset Store から既存のものをインポートすることもできます。

Unity エディター

Unity エディターでは、特定のニーズに合わせてウィンドウを再編成、サイズ変更、非表示、表示できます。カスタマイズしたレイアウトを保存し、素早く切り替えることができます。

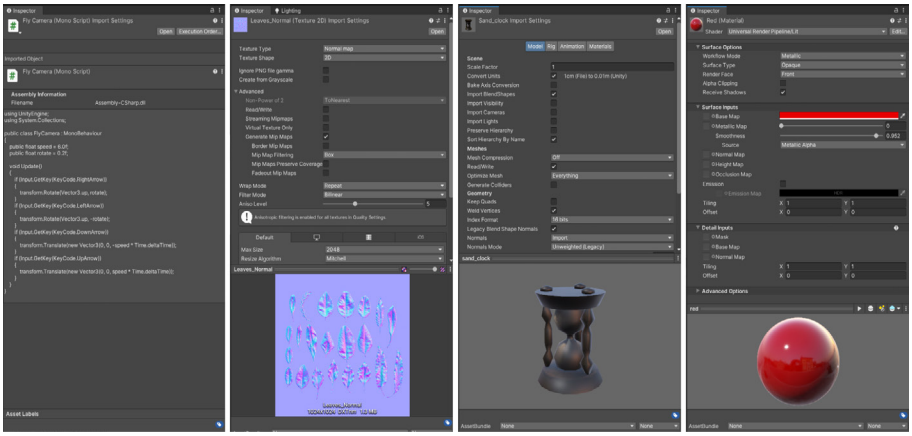
以下の画像は、エディターのデフォルトビューです。

1. 「Hierarchy」ウィンドウ：モデル、カメラ、プレハブなど、シーン内のすべてのゲームオブジェクトを表示する
2. 「Project」ウィンドウ:スクリプト、3D モデル、プレハブ、テクスチャなど、プロジェクト内のすべてのアセットを表示する
3. 「Inspector」ウィンドウ：アセットやゲームオブジェクトの調整可能なプロパティと、関連スクリプトを表示する
4. Scene ビュー：開発中のシーンを表示する
5. Game ビューの再生、一時停止、ステップボタン：ツールバーからアクセス可能なこれらのボタンは、Game ビューをアクティベートし、エディター内でゲームを実行して、プレイ、テスト、イテレーションを可能にする

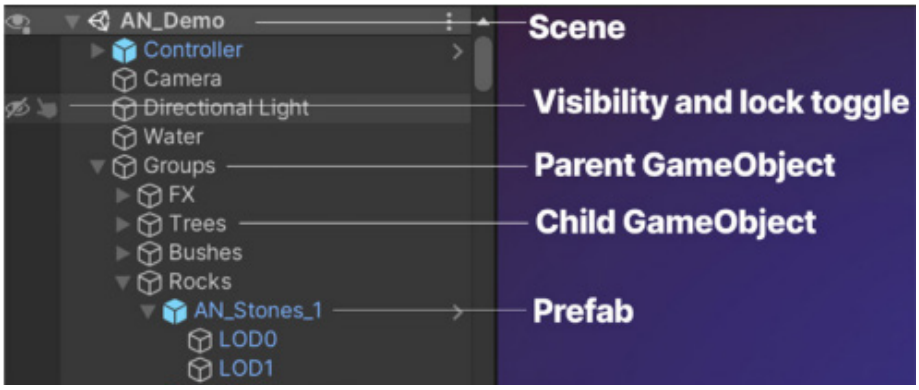


エディターのウィンドウ（「Hierarchy」、「Inspector」、「Project」）と Scene ビュー

各エディターウィンドウとビューのレイアウトや機能の詳細については、ドキュメントの「Unity のインターフェース」セクションをご確認ください。



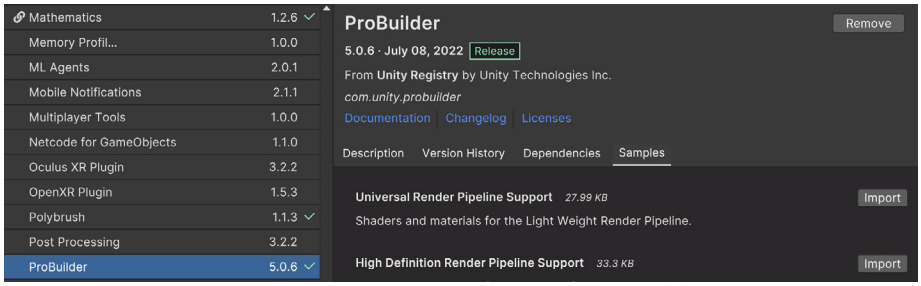
「Project」ウィンドウに異なるアセット、「Inspector」ウィンドウにその設定が表示されている



「Hierarchy」ウィンドウ：この例では、再利用可能なプレハブであるゲームオブジェクトが青いアイコンで表示されている。「Hierarchy」ウィンドウのゲームオブジェクトリストの順番は、プロジェクトに影響を与えないが、Unity UI キャンバスの中でゲームオブジェクトを扱う場合は例外。

Package Manager

Unity の機能の多くは新規プロジェクトにプリロードされていませんが、代わりにエディターの**「Window」>「Package Manager」**からアクセス可能な Package Manager でモジュールパッケージとして提供されています。Package Manager では、利用可能な各パッケージのバージョンを確認し、各プロジェクトのパッケージをインストール、削除、無効化、アップデートすることができます。



Package Manager の Unity Registry

「Package Manager」ウィンドウの最上部にあるドロップダウンリストに、利用可能なすべての公式 Unity パッケージがリストアップされたデフォルトメニュー、**「Unity Registry」**があります。すでにプロジェクトにインストールされているパッケージは、**In Project** にリストアップされています。Unity Asset Store で購入したパッケージは **My Assets**、プリインストールされている機能やパッケージは **Built-in** にリストアップされています。

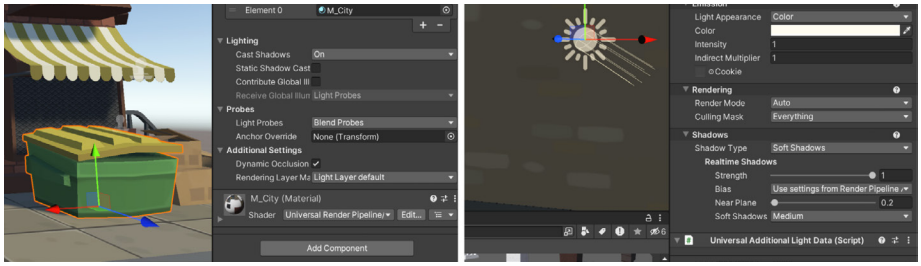
Unity 2022 LTS で利用可能なすべてのパッケージについては、こちらからご確認ください。

ゲームオブジェクト

キャラクター、収集可能アイテム、ライト、カメラ、特殊効果など、Unity シーン内のすべてのオブジェクトは最初にゲームオブジェクトとして作成されます。ゲームオブジェクトは単体では空のコンテナです。ゲームオブジェクトがゲーム内で意味のある存在となり、役割を果たすためには、コンポーネントをアタッチして機能を割り当てる必要があります。そしてコンポーネントは、機能実装のための編集可能なプロパティのセットを提供します。

ゲームオブジェクトには、常に Transform コンポーネント（3D 空間での位置、スケール、向きを表す）がアタッチされています。

エディターの**「Add Component」**メニューから、コンポーネントを追加しましょう。ドロップダウンリストで事前定義されたコンポーネントを選択するか、スクリプトで独自のコンポーネント機能を定義できます。

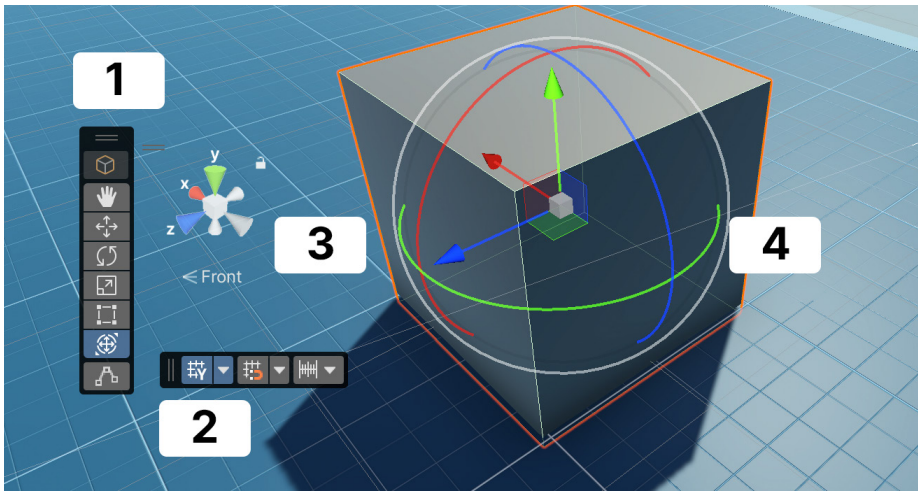


2 つの異なるゲームオブジェクトのタイプ：3D モデルとディレクショナルライトにおいて、Transform コンポーネントとギズモが同じであるのに対し、他のコンポーネントが異なる点に注目してください。

Scene ビューでのゲームオブジェクトの操作

Unity の Scene ビューでは、視覚的なグリッドが提供されていて、ゲームオブジェクトをグリッドにスナップ（移動）して整列できます。

シーンの操作やオブジェクトの操作は、Unity を使用するレベルデザイナーの作業の重要な部分です。トランスフォームツールのショートカットに慣れ親しみ、必要であればカスタマイズすることを検討しましょう。さらに、「Overlays」メニューで Scene ビューにツールを表示するパネルオーバーレイを設定することもできます（ショートカットメニューで、お使いの OS 用のショートカットキーを確認しておきましょう）。



Scene ビューのデフォルトオーバーレイパネル

ツールパネルで利用可能なオプションは、選択したオブジェクトによって異なりますが、以下の表に頻繁に使用されるコアツールとデフォルトのショートカットをまとめました。

ツール	説明	Windows	Mac
	Scene ビューでカメラをパン	Q または shift	Q または shift
	マウスのスクロールまたはショートカットを使用してズームイン、ズームアウト	Alt + 右クリック	Option + 右クリック
	カメラが Scene ビューの中央を中心に周回	Alt + 左クリック	Option + 左クリック
	アイテムを選択、移動	W	W
	オブジェクトの回転角度を変更	E	E
	選択したオブジェクトを拡大縮小	R	R
	オブジェクトの境界ボックス（Rect ツールとも呼ばれる）を使用してスケールを変換	T	T

	画像の（4）のように、ギズモを使用してオブジェクトの位置、回転、スケールを変換	Y	Y
	ゲームオブジェクトに Collider コンポーネントがアタッチされているときに表示されるツール。オブジェクトのヒットボックスやコライダーの境界を操作できる。	-	-

グリッドとスナップツールは、ゲームオブジェクトを最も近いグリッド位置にスナップ（移動）することで、ゲームオブジェクトを整列させます。

ツール	説明
	グリッドのオンとオフを切り替え、グリッド軸と不透明度を変更、グリッド平面を選択したオブジェクト（対象）、またはシーン内の各軸の座標 0 の位置（原点）に合わせる。
	グリッドスナップのオンとオフを切り替える。ドロップリストでオブジェクトのスナップ間隔を設定可能（デフォルトの間隔はすべての軸で 1 単位だが、軸ごとにカスタマイズできる）。また、選択したゲームオブジェクトをすべての軸または選択した軸のグリッドに合わせることもできる。
	インクリメンタルスナップは、ゲームオブジェクトを移動、回転、拡大縮小する際にスナップできる設定で、Windows では Control、Mac では Command キーを使用して利用可能。また、ドロップダウンリストで間隔を設定できる。
頂点スナップ	デフォルトでは、ゲームオブジェクトの操作は、毎回オブジェクトの中心またはピボットポイント（ハンドル位置トグルをご確認ください）から行われますが、メッシュの頂点から操作したい場合は、V キーを押したままマウスを動かして頂点を選択します。これは特に、壁の隅の 1 つを操作して、他の建築構造物とうまくフィットするように整列させる際などに便利です。ゲームオブジェクトの位置合わせの詳細については、こちらのドキュメントをご確認ください。レベルデザイナーが教授してくれたコツは、DCC ツールで 3D モデルのピボットを（0,0,0）の位置に設定することです。これにより、ハンドルモードがピボットポイントに設定されている際、常に下の角からドラッグすることができます。

オリエンテーションギズモには、キューブの両側に円錐状のアームがあります。最前面にあるアームは X、Y、Z とラベル付けされていて、これは Scene ビューのカメラの現在の向きを表しています。これらの円錐をクリックすることで、視野角や投影モードを素早く変更できます。ギズモの真ん中にある白いキューブ、または下部にあるテキストをクリックすると、透視カメラと平行投影カメラ（アイソメトリックカメラまたは 2D カメラとも呼ばれる）を切り替えることができます。ギズモの下のテキストは現在のビューを示しています。鍵マークは、カメラビューの回転を有効または無効にします。これは、ゲームのカメラの角度が固定されていて、ほとんどの間、その角度で作業する場合に便利です。

3D 空間でゲームオブジェクトを操作するシーンギズモは、常に一定のカラーコードに従い、赤は X 軸、青は Z 軸または奥行き軸、緑は Y 軸を意味します。

透視カメラビュー（左）、平行投影ビュー（右）

ゲームオブジェクトの作成

ゲームオブジェクトは 2 通りの方法で作成可能です。1 つは、**「Menu」>「GameObject」**から、もう 1 つは「Hierarchy」ウィンドウまたは Scene ビューに直接アセットをドラッグアンドドロップする方法です。

Unity では、ゲームオブジェクトをグループ化するために、親子階層またはペアレント化の概念を採用しています。ペアレント化、ゲームオブジェクトの整理、子ゲームオブジェクトの作成方法などについては、Unity Documentation の「Hierarchy」ウィンドウページをご確認ください。

すべてのアセットが自動的にゲームオブジェクトに変換できるわけではありませんが、最も一般的なタイプでは変換可能です。FBX ファイル（3D モデル）をシーンに追加すると、モデルの視覚化に必要なコンポーネントを持つ新しいゲームオブジェクトが Hierarchy に表示されます。

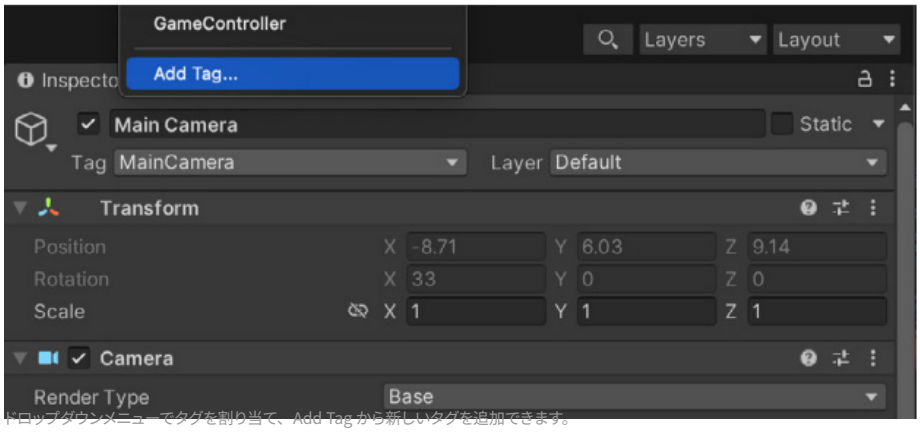
静的および動的なゲームオブジェクト

プロップやモデルなど、ランタイム時に動かないゲームオブジェクトは、静的ゲームオブジェクトと呼ばれます。ゲームオブジェクトの name フィールドの右側にある **Static** ボックスにチェックを入れることで、静的ゲームオブジェクトとしてマークできます。動的ゲームオブジェクトは、ランタイム時に動くゲームオブジェクトのことです。

Unity の多くのシステムでは、静的ゲームオブジェクトに関する情報がエディター内で事前計算されています。つまり、Unity はランタイム時の計算処理を軽減し、パフォーマンスを向上させられます。親ゲームオブジェクトの静的設定を変更すると、更新後の静的設定をすべての子ゲームオブジェクトに反映するかどうかを尋ねるプロンプトが表示されます。これにより、ネスト化されたプロップが大量にあるプロジェクトでは、多くの時間を節約できます。

アクティブ / 非アクティブなゲームオブジェクト

ゲームオブジェクトを非アクティブとしてマークすることで、一時的に Scene ビューから削除することができます。一般的な使用例として、シーンに非アクティブなゲームオブジェクトを配置しておき、ゲームプレイ中、プレイヤーが一定の地点に達した時点でアクティベートするというものがあります。非アクティブなゲームオブジェクトにアタッチされたコンポーネントも同様に無効化されます。親ゲームオブジェクトを無効化することで、関連付けられたすべての子ゲームオブジェクトも無効化されます。



ドロップダウンメニューでタグを割り当て、Add Tag から新しいタグを追加できます。

タグ

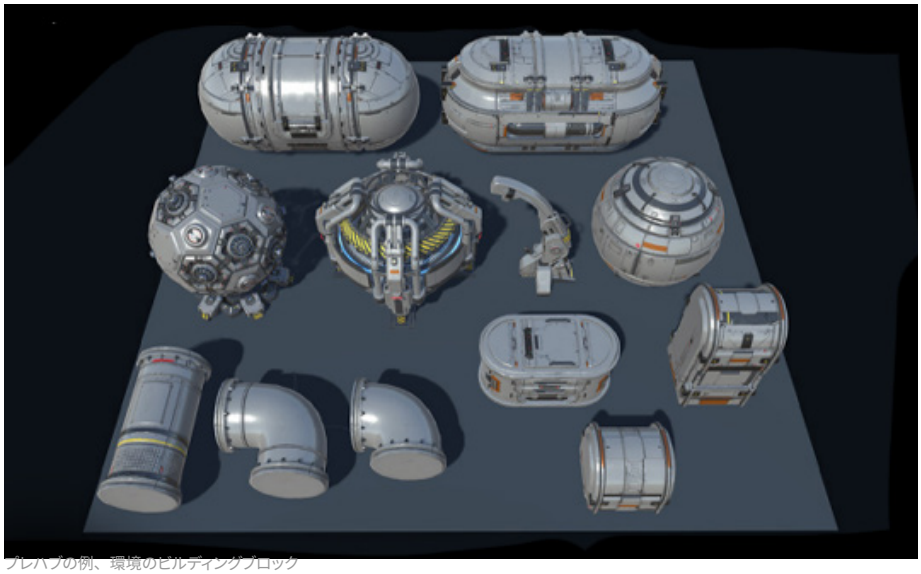
タグは、1 つ以上のゲームオブジェクトに割り当てられる参照用語です。例えば、プレイヤーが操作可能なキャラクターには「プレイヤー」タグ、プレイヤーが操作不可能なキャラクターには「敵」タグ、プレイヤーが収集できるアイテムには「収集可能」タグを追加したりできます。

タグは、スクリプト作成時にゲームオブジェクトを識別するのに役立ちます。オブジェクト名は開発中に変更される可能性があるため、オブジェクト名よりも、タグ指定によるゲームオブジェクトの参照の方が最適です。タグは、衝突判定ロジックに役立ちます。例えば、衝突したゲームオブジェクトに「敵」タグが割り当てられていたら、無効化するなど、何らかのロジックを実行したい場合があるかもしれません。開発チームは、オブジェクトのタグ付け方法について、ゲーム制作の早い段階で共通の認識を形成しておくべきです。

プレハブ：再利用可能なゲームオブジェクト

プレハブシステムは、確実かつ効率的にレベルを埋めるためのツールであり、レベルデザイナーが使用する最も重要なツールの 1 つです。

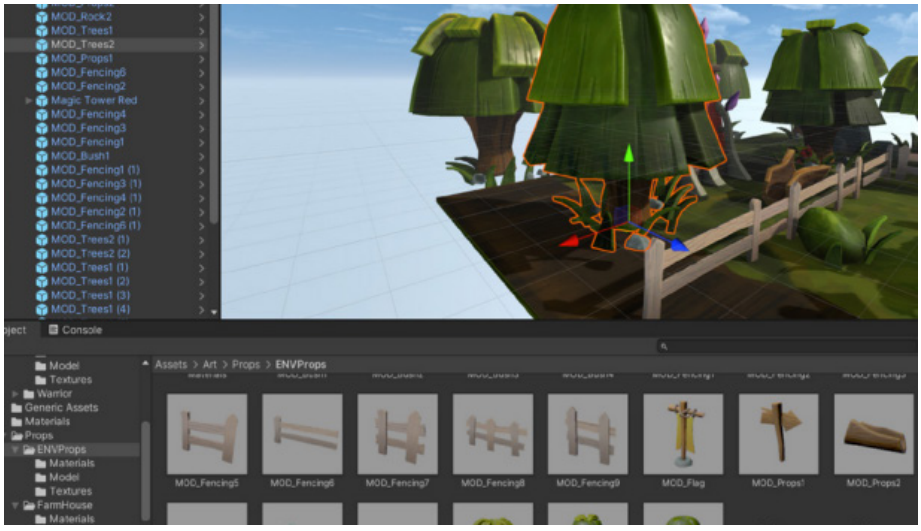
Scene ビューで新しく作成されたゲームオブジェクトは、そのシーンにのみ属します。オブジェクトは複製可能ですが、後で変更を加える必要がある場合、複製されたすべてのオブジェクトに対して手動で行わなければなりません。複数のシーンで同じような多くの要素が繰り返し登場するゲームの制作において、この方法は明らかに現実的ではありません。



プレハブの例、環境のビルディングブロック

Unity のプレハブシステムでは、ゲームオブジェクト、およびそのコンポーネント、プロパティ、子ゲームオブジェクトをすべて再利用可能なアセットとして作成、設定、保存することができます。プレハブアセットは、シーン内に新しいプレハブインスタンスを作成できるテンプレートとして機能します。生成されたこれらのアセットは、再度設定する必要なくシーン間やプロジェクト間で共有できます。

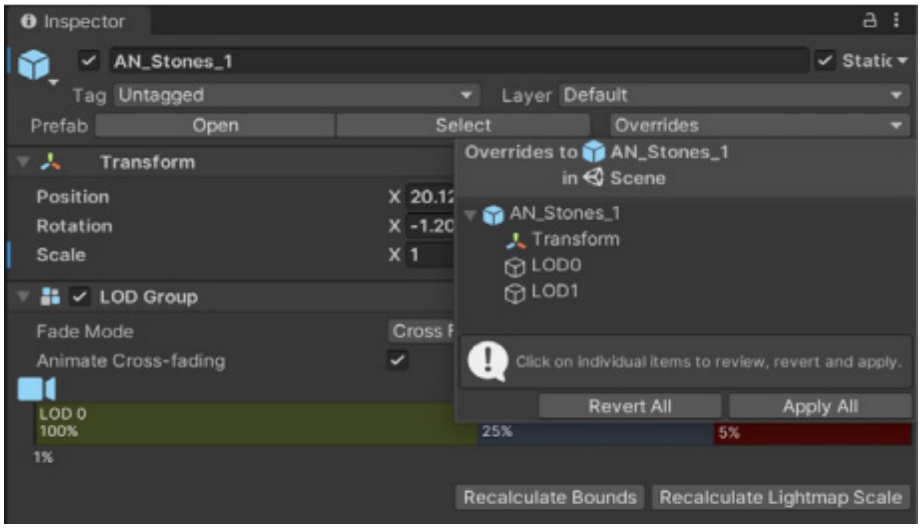
プレハブは編集可能です。オブジェクト単位での編集ができ、シーン内の 1 つのプレハブインスタンスを変更することも、変更をすべてのインスタンスに適用することもできます。これにより、オブジェクトエラーの修正や、アートのスワップ、その他の変更をより効率的に行えます。



「Hierarchy」ウィンドウで選択したプレハブは、「Project」ウィンドウでもアセットとして表示され、何度でも再利用できます。

プレハブインスタンスが修正されると、元のプレハブのプロパティがオーバーライドされ、その隣に青い線が表示されます。ドロップダウンメニューで、すべてのオーバーライド情報を一度に表示することもできます。オーバーライドは、Apply コマンドを使用して元のプレハブ

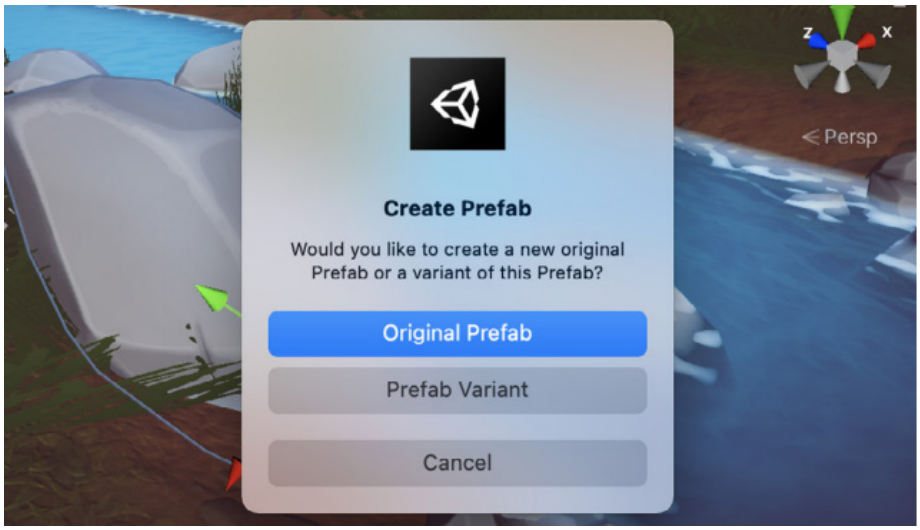
アセットに適用することができ、他のすべてのプレハブインスタンスにも同じ変更が適用されます。別のオプションとして、Revert を使用して元のプレハブ値に戻すことも、修正されたインスタンスをシーン内にそのまま残すこともできます。



スケールの変更に関する「Overrides」ドロップダウンメニュー（青い線が横に表されている）。このメニューからプレハブに変更を適用するか、元のプレハブ値に戻すことができます。

プレハブのネスト化は、プレハブの中に別のプレハブを挿入し、より大きいものを作成できます。この例として、部屋や家具などのより小さなプレハブで構成された建物などが挙げられます。これにより、より簡単にアセットの開発を複数のアーティストや開発者のチームに分け、それぞれがコンテンツの別の部分を同時に作業できるようになります。

プレハブバリアントは、他のプレハブからプレハブを派生させることができます。例えば、異なるステータスやマテリアルを持つ敵キャラクターのバリエーションを作成する場合など、事前定義されたプレハブバリアントのセットがある場合に便利です。バリアントは、シーン内で変更した既存のプレハブを「Project」ウィンドウにドラッグすることで作成できます。



変更されたプレハブを「Project」ウィンドウにドラッグすると表示されるポップアップメニュー

プレハブバリアントは、別のプレハブ（ベースと呼ばれる）のプロパティを継承します。プレハブバリアントのオーバーライドされた値は、ベースプレハブの値より優先します。プレハブバリアントは、モデルプレハブや他のプレハブバリアントを含む、他のプレハブをベースとして持つことができます。また、いつでも変更をすべて取り消し、ベースのプレハブに戻すことができます。

ネスト化したプレハブとバリアントは、バージョン管理システムともうまく連動します。チームメンバーは異なるプレハブで同時に作業したり、競合を発生させずに更新を行ったりでき、開発者が異なる部分のバックアップを常に保持できるようになります。

3D または 2D

Unity では、3D ゲームと 2D ゲームの両方を作成したり、さらには同じプロジェクト内で両方のグラフィックスタイルを組み合わせたりできます。エディター、コーディングワークフロー、一部のツールは、2D 視点と 3D 視点で同じように機能します。ただし、異なる点や、3D の Terrain、2D の Tilemap など、それぞれのゲームタイプに固有のツールもあります。Unity では、2D アセットと 3D アセットの扱いも異なります。以下の表は、2 つの視点の違いと共通点を示したものです。

機能エリア	**3D プロジェクト**	**2D プロジェクト**
共通アセット	3D モデル	スプライト
レンダーパイプライン	ビルトインレンダーパイプライン、URP、HDRP	ビルトインレンダーパイプライン、URP
プロトタイピング	ProBuilder	2D Tilemap Editor
ライティング	Lights	2D Lights
物理演算	Physics	2D Physics
キャラクターリギング	DCC の事前にリグ付けされたモデルや Animation Rigging	2D Animation
環境デザイン	Terrain	2D Tilemap Editor
スプライン	Splines	2D SpriteShape
アニメーション	「Animation」ウィンドウ、Animation Controller	
カメラシステム	Cinemachine	
特殊効果	Particle System と VFX Graph	
シェーダーオーサリング	Shader Graph	
コーディング	C# と Unity Visual Scripting	
UI	Unity UI と UI Toolkit	
入力コントール	Legacy Input または Input System	

3D アセット

3D メッシュは、複数のポリゴン形状で構成された、3D モデルの構造体です。Unity では、ゲームオブジェクトに 3D モデルをアタッチして画面にレンダリングする場合、Mesh Filter と Mesh Renderer の 2 つのコンポーネントを追加する必要があります。

マテリアルは、3D メッシュの表面のレンダリングを定義するもので、テクスチャ、色、シェーダーなど、表面の外観に関する情報を組み合わせます。

シェーダーは、GPU 上で実行される一連の命令で、Unity がゲームオブジェクトを画面上にどのように表示するかを決定します。Unity のすべてのレンダーパイプラインにはあらかじめシェーダーが組み込まれているため、各ピクセルはライティング入力とマテリアル設定に基づいてレンダリングされます。

ファイル形式

Unity 内部では、FBX ファイル形式が使用されています。そのため、制作には可能な限り FBX ファイル形式を採用し、独自のモデルファイル形式の使用は避けることをお勧めします。例えば、Blender を使用して、.blend ファイルを Unity プロジェクトのアセットフォルダーに保存すると、プロジェクトで作業する他のメンバー全員が同じバージョンの Blender をインストールして使用する必要が出てきてしまいます。

Unity に他の DCC ソフトウェアで作成されたメッシュがインポートされる際、保存された位置、回転、スケールを保った状態でインポートされます。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、アニメーションとともにインポートされます。サポートされている他の DCC ソフトウェアでアセットの調整を行うと、Unity は対応するゲームオブジェクトを更新し、ファイルを保存するたびに Unity エディターで変更を反映します。

3D モデルのゲームオブジェクト（左）、2D スプライト（右）

2D アセット

スプライトは 2D グラフィックオブジェクトです。3D での作業に慣れている場合、スプライトは基本的にテクスチャですが、効率的な管理のため、開発中にそれらを組み合わせる特別なテクニックがあります。プロジェクトにテクスチャをインポートする際に、Unity がテクスチャをスプライトとして扱うようにするには、**「Project Settings」>「Editor」>「Default Behavior Mode」** で 2D を選択します。2D テンプレートからプロジェクトを作成した場合は、すでにこの設定が適用されています。

Unity には、アセットの解像度の設定、メッシュやアウトラインの設定、スライスオプション、スキニングやリギング、2D ライティング用の法線マップやマスクマップの追加といったソースアセットの設定を行うためのツールが用意されています。スプライトエディターがその一例です。

3D アセットの場合、アセットの解像度を定義するのはポリゴン数やテクスチャサイズですが、2D アセットでは PPU（Pixels Per Unit）という、ユニットあたりのスプライトの解像度を Unity に伝える指標が用いられます。カメラがユニットに近づきすぎて解像度のピクセル数が足りなくなると、ピクセレーションが発生します。一方、必要以上に高い解像度を使用すると、パフォーマンスとメモリの浪費につながります。2D アセットの解像度についての詳細は、こちらのブログをご覧ください。

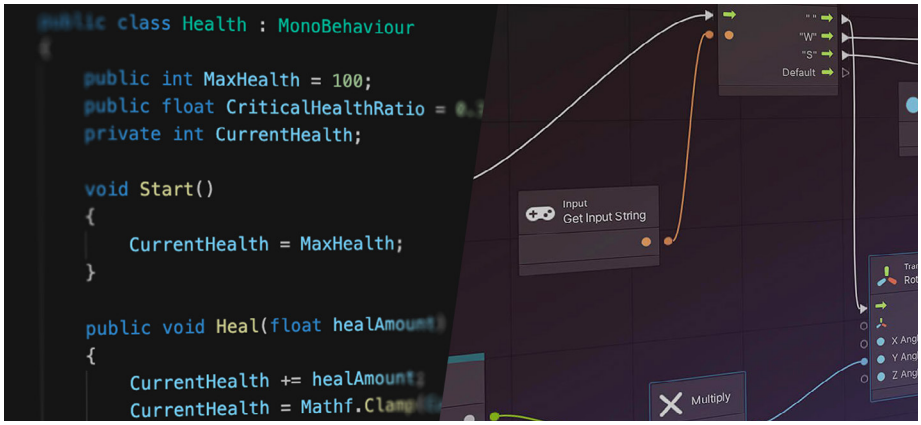
ファイル形式

スプライトは PNG のような可逆圧縮形式でのインポートが推奨されます。Unity では、BMP、TIF、TGA、JPG、PSD など、最も一般的な画像形式をサポートしています。レイヤー化された Photoshop ファイル（.psd）を Asset フォルダーに保存すると、2D PSD Importer パッケージがインストールされていない場合は、それらがフラット化された画像としてインポートされます。2D PSD Importer は、レイヤー情報を読み取り、インポートプロセスを効率化するためのパッケージです。2D PSD Importer についての詳細は、こちらのブログをご覧ください。

Unity の 2D デモ、『Dragon Crashers』は Unity Asset Store で入手可能です。

コーディング

Unity でゲームロジックを実装する方法は、2 つあります。C# スクリプトを書くか、Unity の Visual Scripting システムでノードとグラフを接続してグループ化する方法です。ゲームコードの大部分はプログラマーによって提供されることが一般的ですが、初期のプロトタイピングプロセスの一環として、デザイナーも Unity でのスクリプトの動作に関する基本を理解しておくことは役に立ちます。

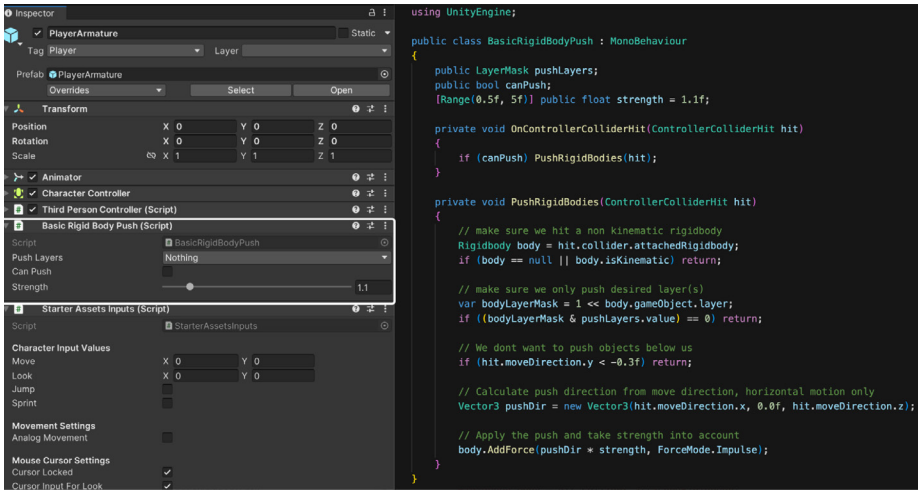


Unity C# スクリプト（左）と Visual Scripting（右）

C# の概要

スクリプトはゲームオブジェクトの動作を定義します。ゲームオブジェクトにアタッチされたスクリプトとコンポーネント、そしてそれらの相互作用によってゲームプレイが生まれます。

スクリプトは、MonoBehaviour と呼ばれるビルトインクラスから派生したクラスを実装することで、Unity 内部の動作に接続することができます。このクラスは、ゲームオブジェクトで使用され、ゲーム開発を可能にする関数を提供します。Unity にはコードを書くためのツールや IDE は含まれていませんが、Package Manager に Microsoft Visual Studio、Microsoft Visual StudioCode、JetBrains Rider を使用するためのパッケージがあります。

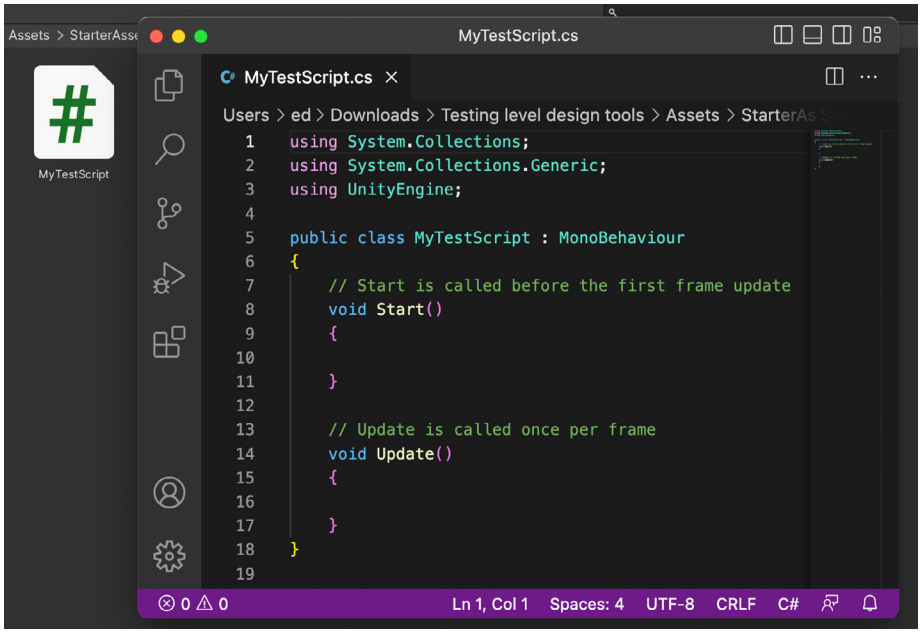


Inspector でハイライトされた Unity コンポーネント（左）と、それに対応するコードの記述（右）

新規スクリプトの作成

エディターの**「Assets」>「Create」>「C# Script」**から、または「Inspector」ウィンドウの**「Add Component」>「New Script」**から新しいスクリプトアセットを作成できます。

Unity によって新規スクリプトアセットが作成されると、スクリプトにデフォルトのコードが挿入されます。



IDE Microsoft Visual Studio Code の Unity スクリプトアセット

上記のスクリプトアセットについて、詳しく見ていきましょう。

— **using** というキーワードを使用して、名前空間をインポートしています。名前空間には、コード内で使用できるクラスや関数が含まれています。**UnityEngine** のインポートは、Start や Update といった Unity 関数にアクセスするのに必須です。

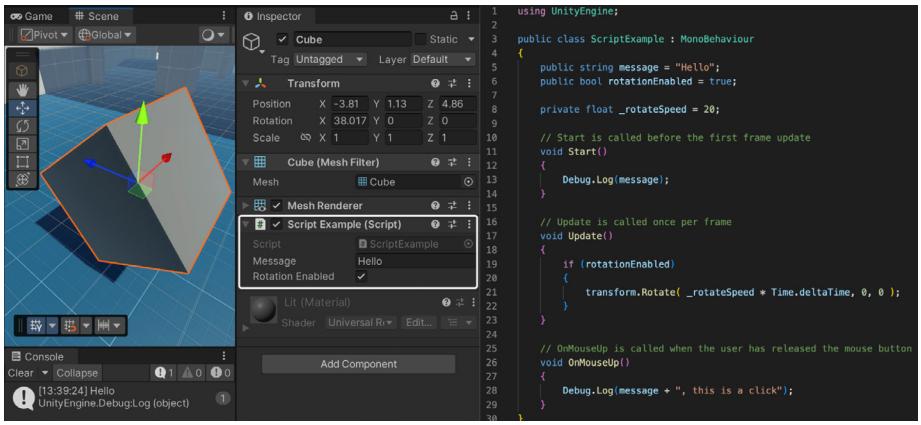
— 行の先頭に 2 つのスラッシュが付いているものは、コメントです。変数や関数の宣言の前にコメントを入れ、コードの内容を簡単に記述するのが一般的です。ゲームの実行時、コメントは Unity に無視されます。

— クラス名（アセットのファイル名と同じにする必要があります）は、MonoBehaviour クラスから派生しています。これにより、スクリプトをコンポーネントとしてゲームオブジェクトにアタッチできるようになります。このクラス内のコードはすべて、かっこや変数、関数の中に記述されています。

— ゲームの実行時に自動的に実行される特別なイベント関数もあります。**Start** 関数はゲームオブジェクトがロードされると自動的に実行され、**Update** 関数はゲームフレームがレンダリングされる度に実行されます。この詳細グラフで、イベント関数の実行順序を確認できます。必要に応じて、正しい箇所に関数を記述します。例えば、Start 関数内で他のコンポーネントへの参照を変数に格納しておき、後でその変数を Update 関数内で使用するのが一般的です。

スクリプトの例

以下の例では、キューブオブジェクトに、キューブを回転させ、コンソールにメッセージを表示するスクリプトがアタッチされています（画像の「Inspector」ウィンドウ内の枠部分）。このスクリプトをさらに詳しく見てみましょう。



エディターや IDE のシンプルなスクリプト

— ScriptExample クラスに、文字列、ブーリアン、float 型の**変数**が含まれています。これらは **public** 変数であり、Inspector で公開され、他のスクリプトから調整したりアクセスしたりできます。**private** 変数は、Inspector に表示されず、他のスクリプトからはアクセスできません。public と private の両方の変数に、宣言時点で事前定義された値が代入されています。

— **Start** 関数は 1 度のみ呼び出される関数で、その内部には「Console」ウィンドウにメッセージを表示する命令が記述されています。

— **Update** 関数の中身は、**if** ステートメントの基本的なロジックです。条件を満たすと、キューブが _rotateSpeed の Time.deltaTime 倍の速さで X 軸方向に回転し、フレームレートに依存しない動きを実現できます。Transform は、アタッチされたスクリプトでゲームオブジェクトの **Transform** プロパティにアクセスするためのショートハンド（短く記述するための方法）です。これはオブジェクトの回転を変更し、毎フレーム実行されます。

— **Event** 関数（デフォルトのコードには含まれない）が追加されていて、マウスのクリック時に Unity がトリガーされるようになっています。その後、「Console」ウィンドウに別のメッセージを表示するよう命令しています。

理想的には、1 つのスクリプトは、1 つの目標のみを達成すべきです。これはプログラミングの単一責任の原則です。各モジュール、クラス、関数が 1 つの目的に責任を持ち、ロジックのその部分のみをカプセル化します。

これにより、コードをモジュール化できます。モジュール性が高く、単一責任の原則に従ったスクリプトは、再利用、拡張、他システムに対するテストをより容易に行えます。スクリプトが複数の課題を解決しようとしている場合、自己完結型の機能を持つ小さいスクリプトに分割した方が上手く機能するかもしれません。

Unity プロジェクトにおけるデザインパターンやプログラミング原則の適用に関する詳細は、このブログ投稿をご覧ください。

ゲームプログラミングパターンでコードをレベルアップ

この eBook では、よく知られたデザインパターンを説明し、Unity プロジェクトでそれらを使用するための実践的な例を紹介しています。

Unity プロジェクトに一般的なゲームプログラミングデザインパターンを実装することで、整理されたクリーンで読みやすいコードベースを効率的に構築、維持することができ、これにより、ゲームや開発チーム、ビジネスの規模を拡大するための強固な基盤を作ることができます。

eBook をダウンロード

学習リソース

Unity のプログラミングは非常に幅広いトピックです。スクリプトを使用すれば、あらゆるゲーム機能を作成でき、カスタムエディターツールや、Unity API を介して、ランタイム時に任意のオブジェクトを作成することもできます。スクリプティングが全く初めての場合は、いくつかの初心者向けリソースで学び始めることができます。

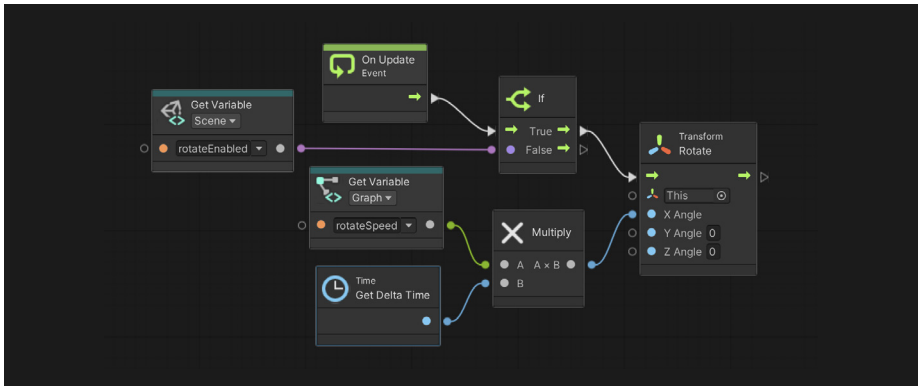
— Coding in C# in Unity for beginners

— Beginner scripting tutorials from Unity Learn

— Creating and using scripts from the Unity manual

Unity Visual Scripting

Unity Visual Scripting は、ノードベースのグラフツールです。これを使うと、プログラマーもノンプログラマーも、プロトタイピングやゲーム制作において、コードを書かずにゲームプレイのロジックやインタラクティブ性をデザインできます。



ゲームプレイ中、Unity Visual Scripting でデータの流れを視覚的に確認できます。ノードの、左から右へ向かう緑の矢印は実行の流れ、オレンジのポートは変数、グレーのポートは変数の入出力値を表します。

Visual Scripting には、デザイナーにとって役立つ重要な機能がいくつかあります。

— **視覚的なノード：**ノードを使用してロジックを作成すると、有効な接続の設定に集中できるため、試行錯誤する手間を省くことができます。さらに、非アクティブなノード、ノード間のロジックフロー、および渡されるデータを確認できます。ノードについてはドキュメントをご覧ください。

— **単純化されたロジック：**ステートグラフは、ステートマシンシステムです。オブジェクトのアクティブステートに基づいて実行するロジックと、別のステートに遷移するタイミングを視覚的に定義できます。ステートグラフは、敵の AI など、ロジックの分岐を必要とするオブジェクトで役立ちます。

— **再生モードでのノードへの変更：**再生モードを終了せずに、ゲームの実行中にその場で動作を変更できます。

新しいスクリプトグラフの作成

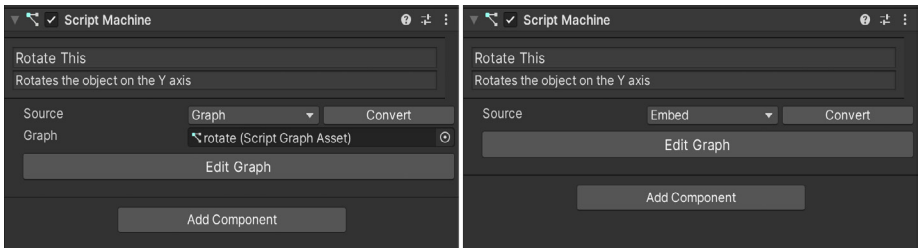
グラフはロジックを視覚的に表現するもので、Visual Scripting の中核です。グラフには 2 つの種類があります。

— スクリプト（フロー）グラフは、個々のアクションと値を特定の順序で接続します。実行順序は、スクリプトの流れです。

— ステートグラフは、様々なステートとステート間の遷移を作成します。

ゲームオブジェクトに対して最初のスクリプトグラフ（動作）を作成するには、**「Window」>「General」>「Hierarchy」**に移動するか、Ctrl+4（macOS の場合は Cmd+4）を押して「Hierarchy」ウィンドウを開きます。Hierarchy でゲームオブジェクトを選択し、**「Add Component」**をクリックして**「Script Machine」**を選択します。

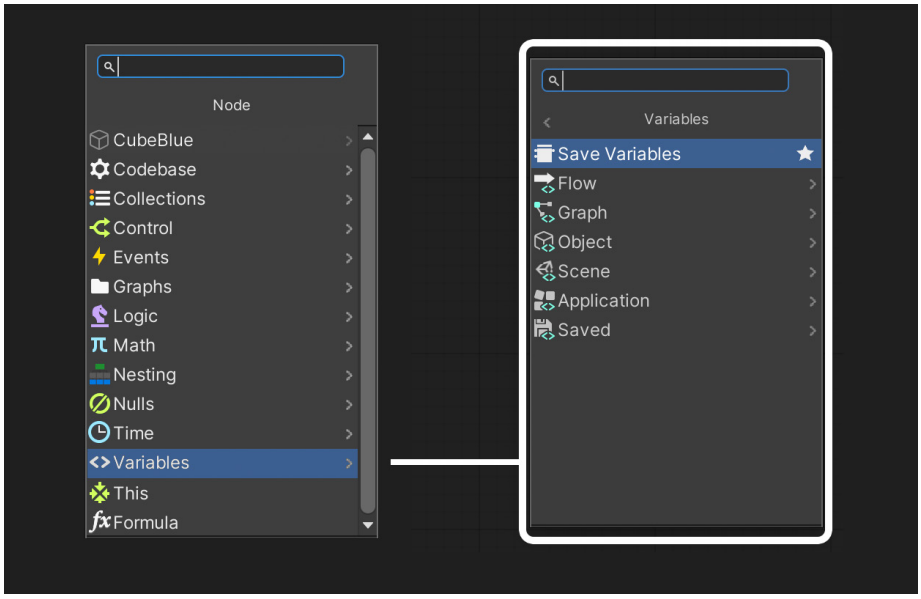
以下の画面では、既存の動作を再利用するか、ソースとしてスクリプトグラフアセットを選択するか、ソースを埋め込んだ新しい動作を作成するかを指定できます。埋め込まれた動作は、**「Convert」**ボタンでいつでもアセットに変換可能です。



スクリプトグラフアセットを使用したコンポーネント（左）と自己完結型の動作を使用したコンポーネント（右）

最初の動作を作成するには、**Edit Graph** でグラフを開くか、「Project」ウィンドウでアセットを開きます。グラフエディターが空になるので、ノードを追加してロジックを作成する必要があります。動作をトリガーするために、最初に Event ノードを追加するとよいでしょう。

Visual Scripting ドキュメントの ノードリファレンスセクションに、使用可能なすべてのノードの一覧が掲載されています。

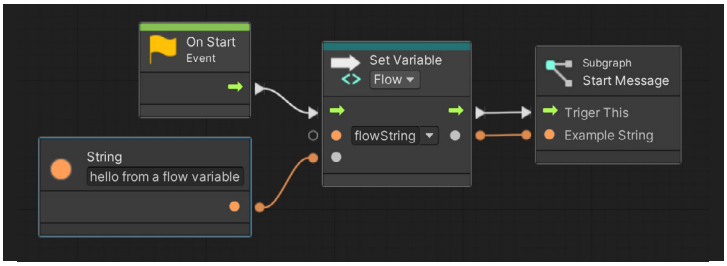
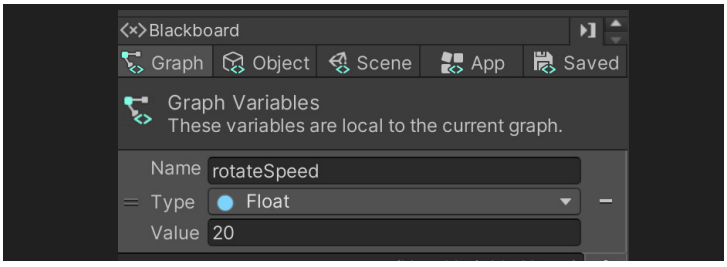


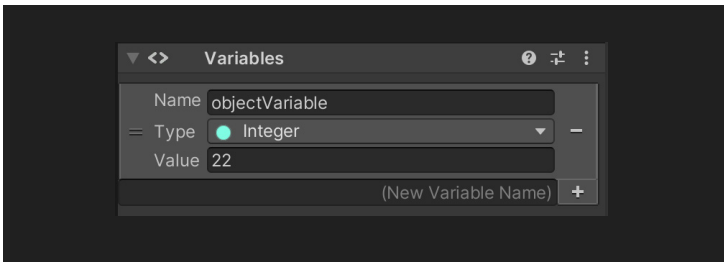
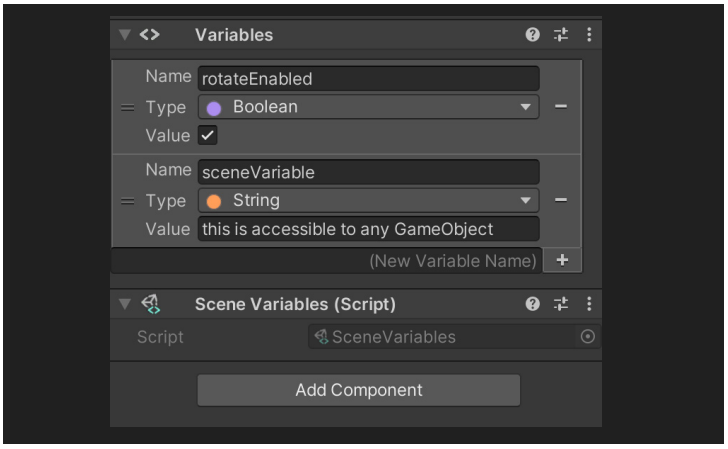
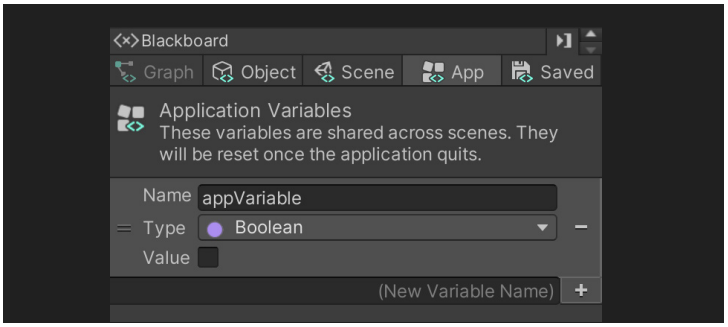
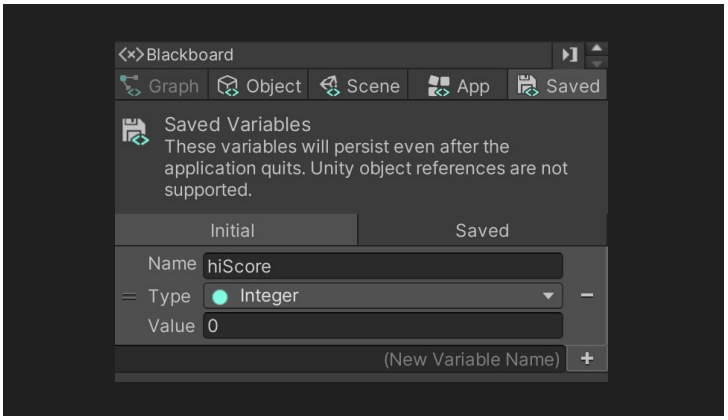
右側の Variable ノードのオプションを使用して新しいノードを作成する様子

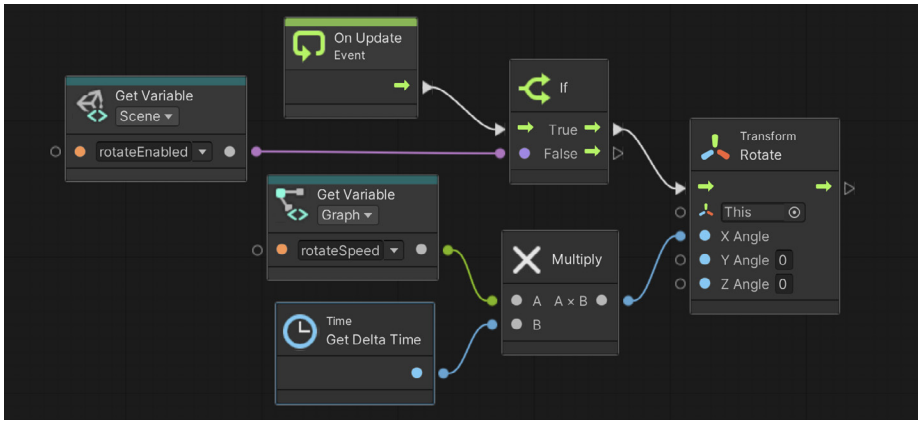
Visual Scripting における変数

ロジックを作成するには、変数を使用する必要があります。これは、アプリケーションの実行中に変更される可能性のある情報のコンテナとして機能します。変数には名前、データ型、デフォルト値が必要です。変数はスコープも持ちます。変数のスコープは、スクリプトグラフのどの箇所からどの変数にアクセスして、その値を参照または変更できるかを決定するものです。このスコープは、別のスクリプトグラフから変数にアクセスできるかどうかも決定します。

以下の表は、様々な変数のスコープを示しています。

スコープ	説明	Unity でのビジュアライゼーション
フロー変数	同じフロー内で直接または間接的に作成および使用する必要がある。フロー変数は最小のスコープを持つ。	 グラフエディターからのみアクセス可能
グラフ変数	同じスクリプトグラフ内のどのノードでも使用可能。Blackboard 要素から作成および追加できる。	 Blackboard エリアからアクセス可能

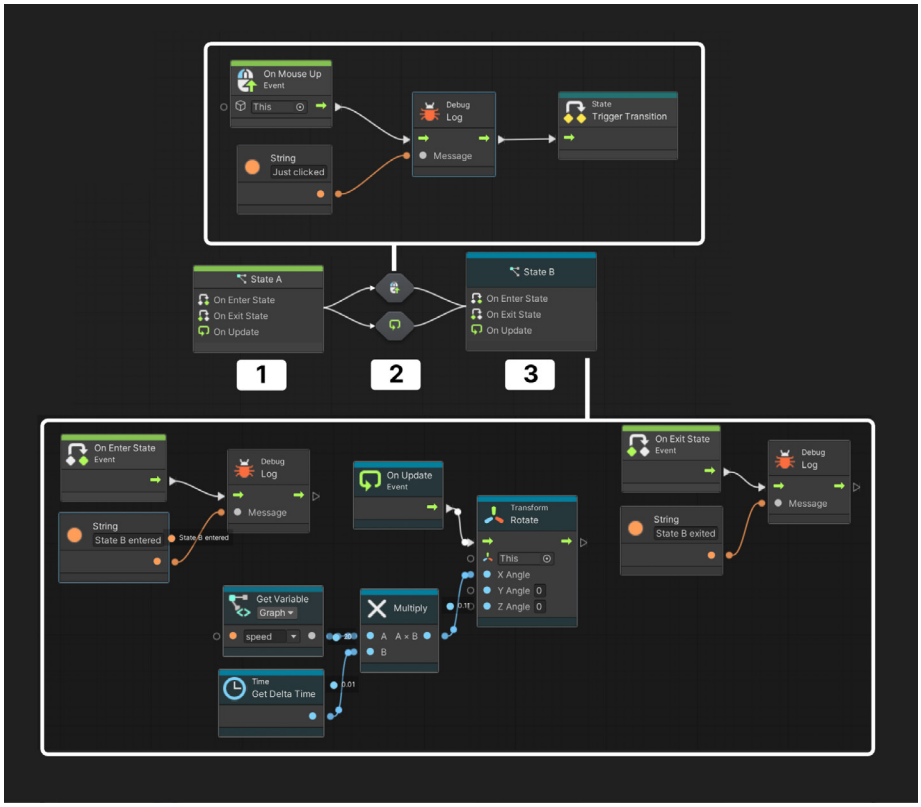
オブジェクト変数	特定のゲームオブジェクトにアタッチされたどのスクリプトグラフ、サブグラフ、ステートグラフでも使用できる。	 Blackboard エリアとゲームオブジェクトの Inspector からアクセス可能
シーン変数	現在のシーンに属する。シーン変数には、1 つのシーンの異なるゲームオブジェクトにアタッチされたスクリプトグラフからアクセスする。	 シーン変数
アプリ変数	アプリケーション全体に属し、アプリケーション実行中に複数のシーンでアクセス可能。	 Blackboard エリアからアクセス可能
保存された変数	スコープはアプリ変数と同じだが、アプリが終了しても持続するという違いがある。Unity では PlayerPrefs と呼ばれる。	 Blackboard エリアからアクセス可能



アタッチされたゲームオブジェクトを回転させるシンプルなスクリプトグラフで使用されるシーン変数とグラフ変数

ステートグラフを使用したステートマシン

ステートグラフは、ゲームオブジェクトにアタッチされるコンポーネントで、ロジックをアセットとして用意するか、またはゲームオブジェクトに埋め込むことができます。作成方法は、スクリプトグラフと同様です。ステートグラフは AI の動作やシーン構造を作成します。ステートは、オブジェクトがそのステートを持つ間の動作を指示します。例えば、動作を敵 AI の「パトロール」や「追跡」、またはドアの「ロック」、「アンロック」、「オープン」ステートなどと考えてみてください。あるステートから別のステートへ遷移するロジックは、transitions、つまり遷移をトリガーするスクリプトグラフにあります。例えば、「プレイヤー」が「敵」に近づきすぎると、「パトロール」から「追跡」ステートに変わります。



上記画像のステートグラフの段階を追ってみましょう。

1. これは、ノード上部が緑のバーでマークされた開始ステートです（ステートを右クリックすることで、開始ステートに切り替えることができます）。新しいステートは、**Enter、Exit、Update** の 3 つのイベントを作成します。これらを使用してロジックの作成を開始できます。
2. これらは 2 つの transition です。1 つは、オブジェクトがクリックされたときに transition ステートをトリガーし（図の上部参照）、もう 1 つは Update 時に特定の条件を評価し、その条件を満たしている場合、遷移を実行します。トランジショングラフの六角形のアイコンは、transition の動作がどのイベントを使用するかを視覚的に示すヒントになります。
3. これは現在アクティブなステートで、青いヘッダーラインによってそれが示されています。このステートに入った、またはステートから出た際、デバッグメッセージが表示されます。このステートではオブジェクトが回転します（図の下部を参照）。

Visual Scripting と C# スクリプトは、同一プロジェクトで併用することができます。例えば、チーム内のプログラマーがカスタムの Visual Scripting ノードを作成し、デザイナーはそれを使用して管理された環境で機能を操作できます。多くのスタジオでは、このように開発者にカスタムの Visual Scripting ノードを作成させ、デザイナーがゲームのコードベースで処理される機能やイベントを簡単に利用できるようにしています。

このプレゼンテーションでは、プログラマー向けに Unity Visual Scripting で可能なことを紹介しています。

その他のリソース

— Unity Visual Scripting のページ

— Visual Scripting application：Clive the Cat’s

— Visual Scripting ドキュメント

— Unity Visual Scripting for artists and designers

物理演算

Unity はリアルな物理シミュレーションのための 3D と 2D の完全な物理演算システムを提供しています。ビルトイン 3D Physics システムはメッシュベースのゲームオブジェクトに使用されます。2D Physics システムは、スプライトベースのゲームオブジェクトを使用します。

物理パラメーターのバランスと調整は、ゲームプレイをデザインする上で重要な要素です。重力、オブジェクトの衝突、オブジェクト同士の反応など、ほぼすべてのゲームに物理シミュレーションと相互作用が必要になります。

Creator Kit:Puzzle（Unity Asset Store で入手可能な Unity の新規ユーザー向けのチュートリアルプロジェクト）の画像

衝突の作成

Collider コンポーネント

コライダーはゲームオブジェクトに適用され、物理シミュレーションで物理的な形を表現します。プリミティブ形状のボックス、カプセル、スフィア、ホイール型のコライダーコンポーネントがあり、「**Edit Collider**」ボタンで変更できます。通常、これらのプリミティブ形状は、コアとなるゲームプレイのメカニクスのイテレーションには十分です。

物理システムは現実世界の物理を再現するために Unity ユニットを参照として使用します。ユニットのスケールは 1 メートルに相当します。異なるサイズのオブジェクトは、正確なスケールでモデリングする必要があります。例えば、人間のキャラクターの身長は 2 ユニット程度であるべきです。適切なサイズのメッシュをゲームオブジェクトに使用することが重要です。崩れそうな高層ビルと、おもちゃのブロックでできたタワーとでは、シーンでの崩れ方が異なる必要があります。

Collider コンポーネントはオブジェクトの物理的な境界を定義します。ゲームオブジェクトに静的コライダーを追加すると、物理システムはオブジェクトを強固で動かないものとして扱います。

Rigidbody コンポーネント

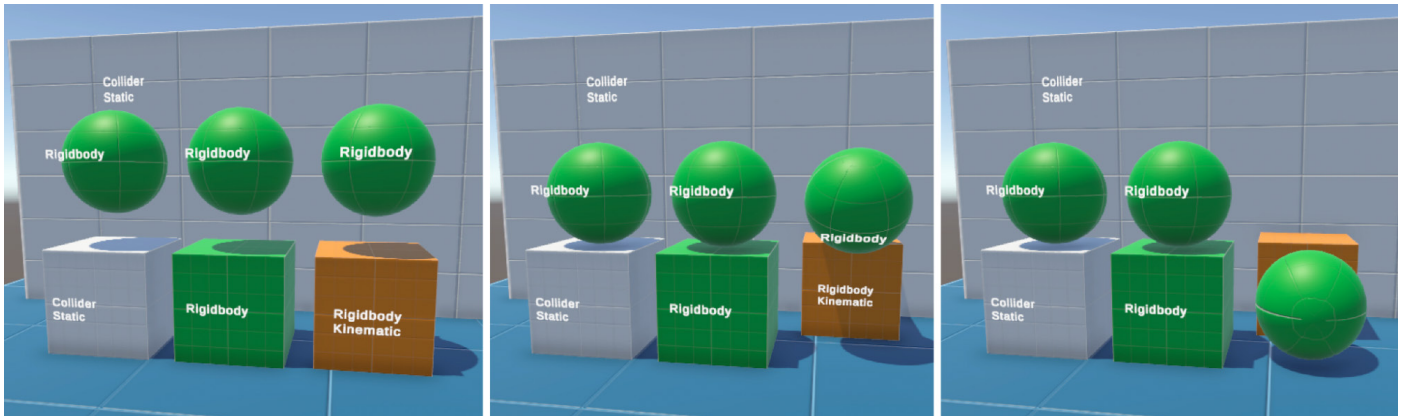
移動、重力、衝突、ジョイントなどの物理ベースの動作のシミュレーションを行うには、シーン内のアイテムをリジッドボディとして設定する必要があります。これは、ゲームオブジェクトに Rigidbody コンポーネントを追加することで可能です。Rigidbody コンポーネントは、ゲームオブジェクトの動きと位置を物理演算に基づいて制御する方法を提供します。

Rigidbody コンポーネントを持つゲームオブジェクトは、静的オブジェクトとの現実感のある衝突を再現できます。また、動的なリジッドボディオブジェクト同士を衝突させることもできます。例えば、2 つのスヌーカーボールがぶつかり合う場合などです。

Rigidbody コンポーネントの 2 つの重要な設定を見てみましょう。

— **Is Kinematic** プロパティは、リジッドボディが物理を介して、それ自体は影響を受けずに、他のオブジェクトに影響を与えることを可能にします。例えば、VR ゲームの手のアバターは物理を介してオブジェクトと相互作用しますが、手やプレイヤーがジャンプする動くプラットフォームは物理の影響を受けないようにしたいはずです。

— **Use Gravity** プロパティはその名の通り、ゲームオブジェクトに作用する重力です。このプロパティにチェックが入っていない場合、オブジェクトは他の物体に押されることはできますが、重力による減速がないため無重力に見えます。



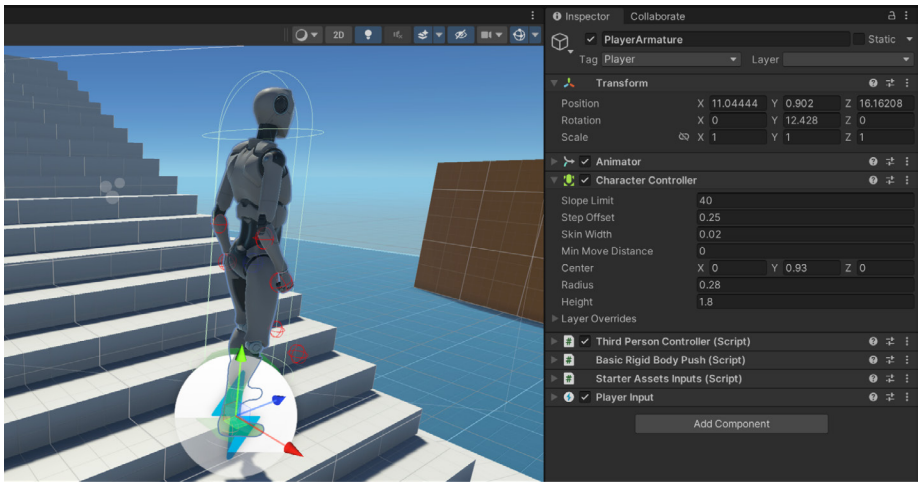
上の画像の緑色のスフィアは、他の動的、静的、またはキネマティックオブジェクトと相互作用できる動的なリジッドボディです。再生モードでは、シミュレートされた重力が動的なリジッドボディを下方に引っ張り、他の要素と衝突させます。オレンジの立方体はキネマティックオブジェクトです。スクリプトにより、他の力に反応することなく前後に動き、その結果、静的な壁を通り抜け、緑のスフィアを押す際に抵抗がなくなります。

Unity の物理衝突の仕組みを理解し始めるにつれ、チュートリアルやドキュメントを通して、いくつかの共通用語があることに気づくはずです。

用語	必須コンポーネント	動き	一般的な実装	ユースケース
静的コライダー	コライダー（ボックスコライダー、スフィアコライダー）など	なし	レベルデザイン過程で Scene ビューに追加	壁、境界線、大きな小道具、地形の地面、静的オブジェクト
キネマティックリジッドボディコライダー	コライダーとリジッドボディ（**Is Kinematic** プロパティが有効化）	あるが、外力には反応しない	移動パターンはコード内またはアニメーションで実装	動くプラットフォーム、アニメーションがついた大きな小道具、スクリプト化された、または VR ゲーム内のキャラクターやオブジェクト（例：オブジェクトと相互作用可能なバーチャルハンド）

リジッドボディコライダーまたは動的リジッドボディ	コライダーとリジッドボディ（**Is Kinematic** プロパティは無効）	質量、重力、抗力が、力への反応に影響する。静的、キネマティック、動的オブジェクトと衝突する	Scene ビューで無生物オブジェクトに設定するか、コード内で力や速度を適用	パズルオブジェクト、インタラクタブルオブジェクト、重力に反応する必要がある小道具など、物理演算ベースのゲームプレイ要素
キャラクターコントローラー	キャラクターコントローラー	正確な動き（ただし常に現実的とは限らない）。力に反応し、静的、キネマティック、動的オブジェクトと衝突する	コード内のプレイヤー入力によって制御	リアルな物理演算シミュレーションよりも精度が必要なプレイアブルキャラクター（例：一人称視点シューティングゲーム内）

ほとんどの場合、プレイヤーキャラクターには Character Controller というコンポーネントが使用されます。Character Controller は、3D キャラクターが周囲の物理世界と相互作用できるようにするもので、必ずしも現実的に動作するわけではありません（ゲームのスタイルやジャンルによっては、現実的な動作よりも、プレイヤーにより細かいコントロールや精度を提供したい場合があるはずです）。

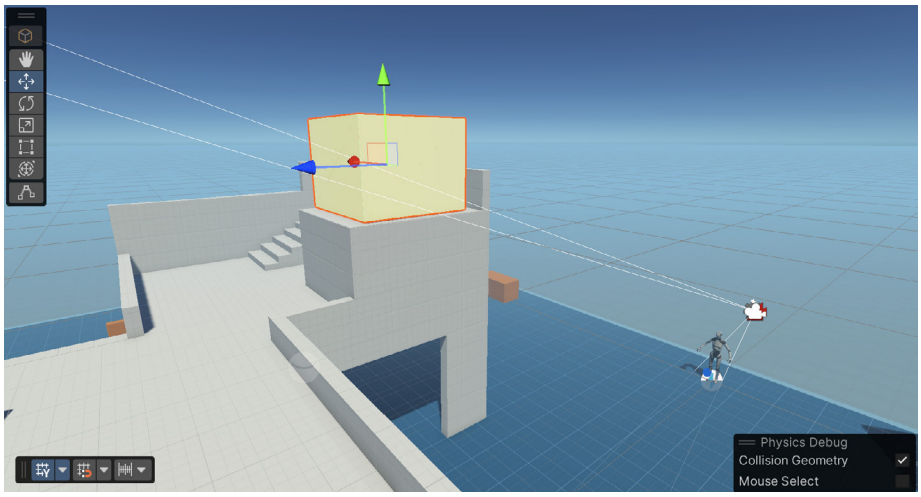


Unity のサンプルプロジェクトである Third Person Character Controller と First Person Character Controller には、プレイアブルキャラクターコントローラーが事前に設定されています。

トリガーコライダー

トリガーとして設定されたコライダー（**Is Trigger** プロパティを使用しているもの）は、オブジェクトが物理的な接触なしに他のオブジェクトを検出したり、反応したりする必要があるときに使用できます。このプロパティは、オブジェクトがセンサーのように動作することを可能にします。Is Trigger プロパティの使用場面の例としては、エントリー、オートセーブ、脱出エリアの指定が挙げられます。

コードでの機能の実装は、リジッドボディの衝突シナリオと非常に似ていますが、オブジェクトは衝突時に、衝突メッセージ（OnCollision）の代わりにトリガーメッセージ（OnTrigger）を受け取ります。



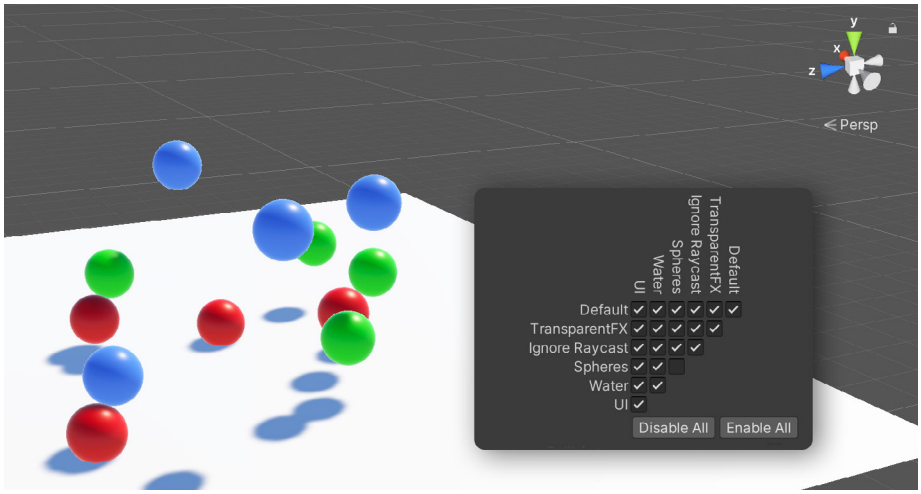
トリガーコライダーは、オブジェクトがコライダーに入ったり、留まったり、離れたりするタイミングを認識できます。この例では、プレイヤーが黄色いキューブに入ってビルの最上階に到達した時に、レベル終盤のカットシーンやシーケンスがトリガーされます。

Is Trigger が有効になった静的、リジッドボディ、キネマティックコライダーが、トリガーが有効になっていないリジッドボディやキネマティックコライダーと相互作用すると、トリガーメッセージが発生します。静的なトリガーコライダーどうしや、静的な非トリガーコライダーとは相互作用しません。

物理演算レイヤー

レイヤーベースの衝突は、ゲームオブジェクトを特定のレイヤーに設定された別のゲームオブジェクトと衝突させる方法を提供します。Project Settings の Collision Matrix から、どのゲームオブジェクトをどのレイヤーに衝突させるかを定義できます。例えば、レイヤー「Bullets」にある弾オブジェクトは、同じレイヤー内では衝突しませんが、レイヤー「Ground Enemies」のオブジェクトとは衝突するなどと設定できます。

ヒント：物理演算のパフォーマンスを維持するには、必要な衝突レイヤーのみを有効にしてください。

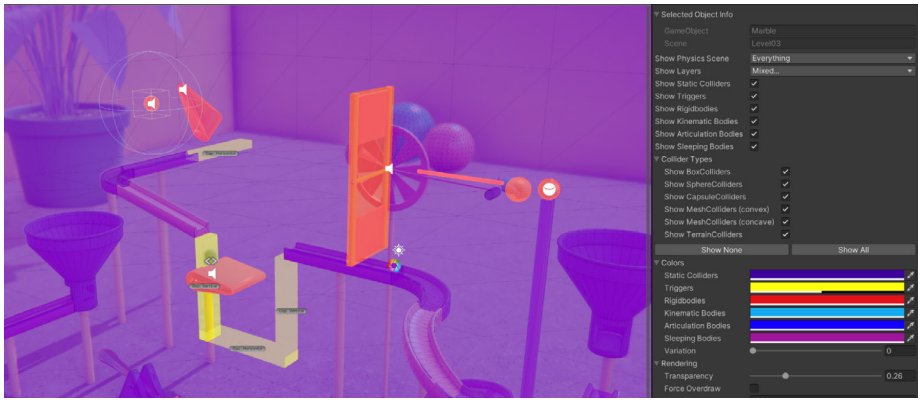


スフィアレイヤーのカラフルなスフィアは、デフォルトレイヤーの床に衝突しますが、スフィア同士は衝突しません。

物理演算を使ったレベルデザイン

静的オブジェクトと動的オブジェクトが混ざった大規模なシーンを作成すると、ゲームプレイのテストが難しくなり、キャラクターがレベル境界の外に出てしまったり、オブジェクトがトリガーとして機能しないといったエラーの原因を理解するのが難しくなります。

エディターの**「Window」>「Analysis」>「Physics Debugger」**で物理演算デバッグを有効にできます（シーンギズモが表示されるようにしてください）。Physics Debugger は、シーン内の様々なタイプの Collider コンポーネントと Rigidbody コンポーネントを表示し、ハイライトします。いくつかのタイプを非表示にしたり、物理演算レイヤーでフィルタリングしたりできます。これにより、シーン内に多くのオブジェクトがあっても、問題の根本原因を効率的に見つけることができます。



「Physics Debug」モードのウィンドウから、ビジュアル設定をカスタマイズし、ビジュアライザーで表示または非表示にするゲームオブジェクトのタイプを指定できます。

Physics Debugger は、ゲームデザイナーが衝突の問題を素早く可視化するために、定期的に使用されます。Physics Debugger によるデバッグは、UModeler や Unity の ProBuilder ツールなどで開発している場合にも便利です。メッシュの修正を行う際、衝突設定を更新し忘れることがあります。Physics Debugger は、更新が必要な場所をピンポイントで特定するのに役立ちます。

物理演算リフレッシュレート

Unity では、ゲームオブジェクトは **Update** サイクルと呼ばれるレンダリングフレームごとに実行されるコードをスクリプトに実装できます。60fps で動作するゲームの場合、Update ループ内のコードは毎秒 60 回実行されます。物理演算関連のコードの問題は、フレームのレンダリングに常に同じ時間がかかるとは限らないことです。低速になるとフレームがスキップされることさえあり、物理演算の決定的動作としては信頼できません。物理計算には **FixedUpdate** サイクルが使用されます。デフォルトでは、FixedUpdate は 0.02 秒ごとに発生します（1 秒あたり 50 回のコール）。ゲームのニーズによっては、チームのプログラマーと密接に連携し、プロジェクトのニーズに合った適切なプロジェクト設定を行うことが重要になります。

パラメーター **Time.deltaTime** を使用して、Update サイクルでフレームに依存しないロジックを実現することもできます。例えば、1 秒ごとに直線的に位置を変えるオブジェクトに、時間経過に伴う滑らかな動きを適用したいとします。Time.deltaTime を乗算することで、Update サイクルで使用されたときに動きがガタつかないよう、ロジックが最後のフレームのレンダリングにかかる時間を考慮するようになります。前のフレームのレンダリングに時間がかかる場合、Time.deltaTime の乗算値はそれを補うために大きくなります。

```
void Update()
{
    Vector3 position = transform.position;
    position.x += SpeedValue * Time.deltaTime;
    transform.position = position;
}
```

FixedUpdate と Time.deltaTime を使ってオブジェクトを直線的に動かすスクリプトの例

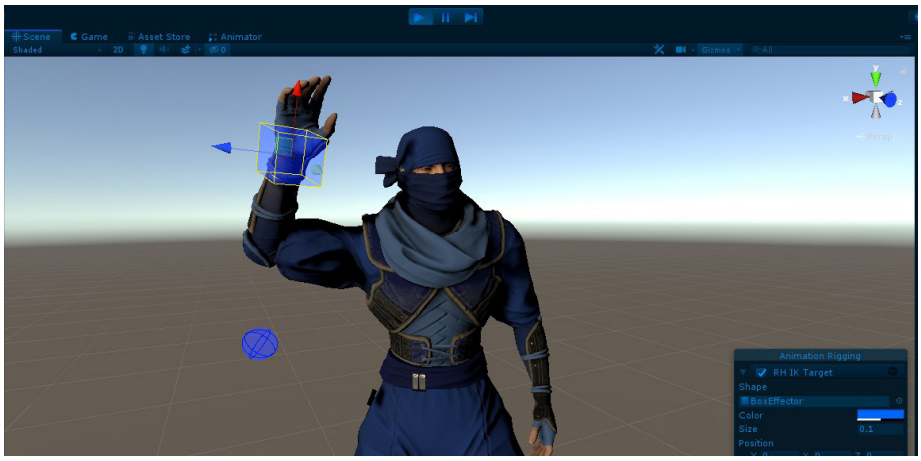
Interpolations を使用すると、FixedUpdate イベントと通常の Update イベントを組み合わせたときに、異なるリフレッシュレートに起因するゲームプレイのスタッターを回避できます。表示状態で動いているゲームオブジェクトのリジッドボディに interpolate と extrapolate プロパティを適用できます。このテクニックについてはこちらのコミュニティブログをご覧ください。

Unity の 3D 物理演算はゲームプレイ作成に関する多くのことを可能にします。ジョイント、アーティキュレーション、ラグドールなどの物理演算については物理演算のドキュメントを参照してください。

Collider 2D、Rigidbody 2D、2D Joints など、2D プロジェクト特有の物理演算コンポーネントがあります。原理は 3D と同じで、実装方法も似ています。3D と 2D のシステムは相互作用しないため、どちらかの視点を選ぶ必要があることに注意してください。2D 物理演算についてはドキュメントを参照してください。

2D 物理演算機能の概要については、こちらの動画を参照してください。

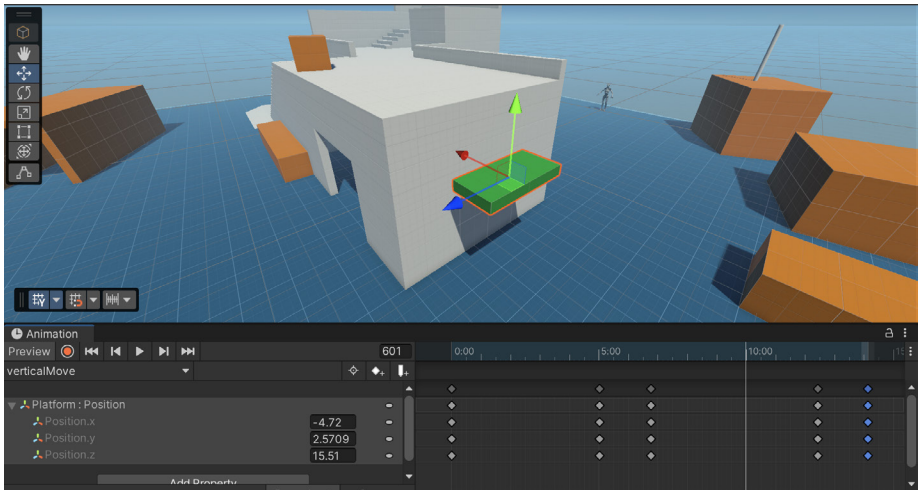
アニメーション



アニメーションシステム

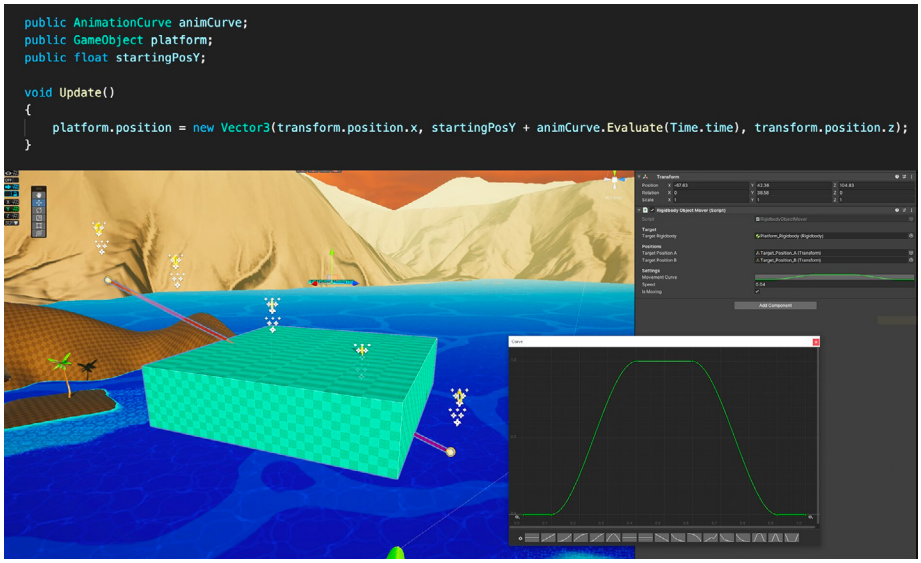
Unity のキャラクターアニメーションは通常、Blender、Autodesk Maya、Autodesk 3ds Max などの DCC ソフトウェアから作成されます。Unity のアニメーションシステムは、制作中のゲームやインタラクティブな体験に命を吹き込むために、アニメーションを修正、洗練、プロシージャル調整、ブレンドするためのツールを提供します。

ただし、レベルデザインのプロトタイプでは、動く要素に複雑なリギングシステムやスクリプト化されたシーケンスは必要ありません。Unity のアニメーションツールを使用すれば、プロトタイピングに適したアニメーションやシーケンスを作成することができ、レベルデザイン決定後は、チームの専属アニメーターに最終的な実現と調整を任せられます。



Unity の「Animation」ウィンドウでプラットフォームに垂直方向に動くアニメーションをつける様子

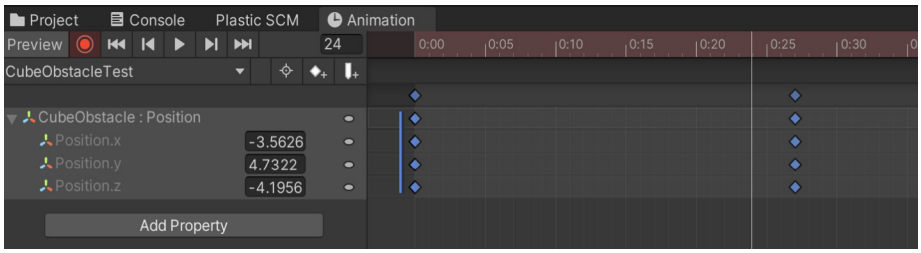
Unity のアニメーションシステムは**アニメーションクリップ**という概念に基づいています。これは、特定のオブジェクトの位置や回転、その他のプロパティが、どのように時間経過に伴って変化するかという情報を持っています。各クリップは、1 つの直線的な記録と考えることができます。オブジェクトが「idle」「disabled」「active」など異なるアニメーションステートを持つ場合、アニメーションステートマシンとなる **Animator Controller** を使用することができます。



このブログで説明されているように、アニメーションカーブの助けを借りて、プログラムによって要素にアニメーションをつけることができます。

「Animation」ウィンドウ

「Animation」ウィンドウでは、Unity で直接アニメーションクリップを作成および変更できます。外部の 3D アニメーションソフトウェアの代替として使用すること、または開発中に必要に応じて簡単なアニメーションを作成することが想定された設計となっています。**キーフレーム**、**プレイヘッド**、**アニメーションタイムライン**、**アニメーションカーブ**といった、アニメーションに必要な標準ツールセットが備わっています。



「Animation」ウィンドウでは、上部のドットを選択することで、同じタイムスタンプからすべてのキーフレームを簡単に変更できます。すべてのキーフレームを選択し、左右のハンドルを動かすことで、アニメーションの継続時間を変更できます。

動きの他にも、マテリアルやコンポーネントの変数（ほぼすべてのゲームオブジェクトのプロパティ）にアニメーションをつけたり、エディターで、タイムラインに沿った特定のポイントで呼び出される関数である**アニメーションイベント**を使ってアニメーションクリップを補強したりできます。

Unity の「Animation」ウィンドウでは、以下のものにもアニメーションをつけることができます。

— ゲームオブジェクトの位置、回転、スケール

— Material Color、Light Intensity、Sound Volume などの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

— 作成したスクリプト内で関数を呼び出すタイミング

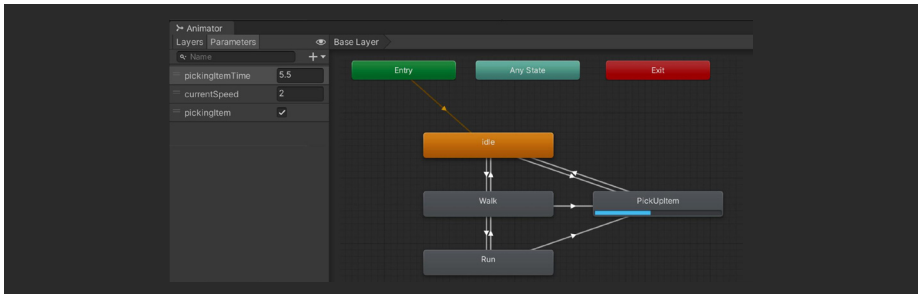
生成されたアニメーションクリップは、アニメーターコントローラーや Animation Rigging ツールキットで使用したり、タイムラインを使ってゲームプレイやシネマティック中に利用したりできます。

キーフレームやカーブなどのアニメーションツールをプロジェクトに使用しましょう。

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して、キャラクターやオブジェクトのアニメーションクリップと関連する **Animation Transitions** のセットを配置し、保持することができます。

複数のアニメーションを用意し、特定のゲーム条件が発生したときに切り替えるのが通例です。例えば、スペースキーを押すたびに、ウォーククリップからジャンプ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、内部で使用されるアニメーションクリップへの参照を持ち、アニメーションステートマシンを使用して、これらの参照とクリップ間の遷移を管理します。ステートマシンは、クリップと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書かれたシンプルなプログラムと考えることができます。



ゲームオブジェクトの動作に関するアイデアは、アニメーションコントローラーのステートマシンを使って視覚的にテストできます。

ブレンドツリーは複雑さを隠すのに効果的です。ブレンドツリーはステートを持たず、コードにコールバックされることもありません。単に、定義したパラメーターに基づいて異なるクリップをブレンドするだけです。これは、ゲームの他の部分が機能しなくなることを心配せずに、ブレンドツリーのイテレーションを行えるという点で重要です。また、ブレンドツリー内のほとんどのアニメーションには動作を関連付けられないため、複雑なステートの集まりを隠してバグを防ぐこともできます。

Unity には、複雑な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があります。例えば、アニメーションレイヤーを使って、歩いたりジャンプしたりする下半身レイヤーと、オブジェクトを投げたり射撃したりする上半身レイヤーを作成します。

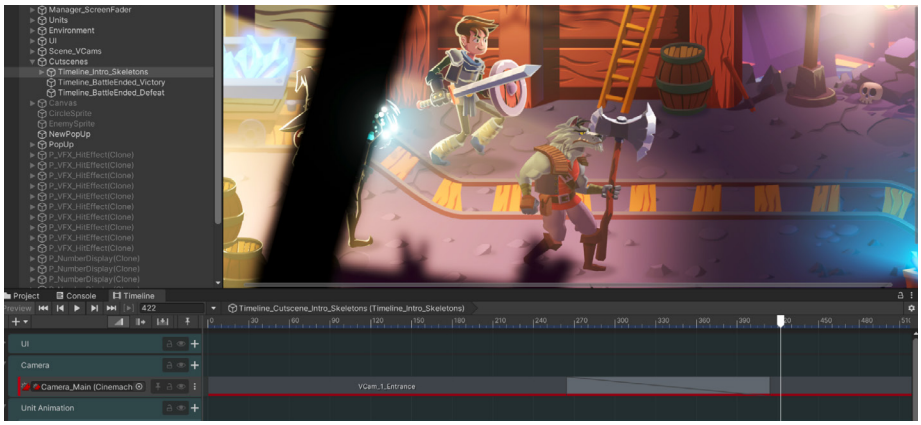
視覚的なアニメーションに加えて、アニメーションステートはサウンドエフェクトや C# コードをトリガーできます。

タイムライン

タイムラインを使用すると、動作、アニメーション、オブジェクトの有効化 / 無効化など、シーンの要素を編成したり、シーケンスやカットシーンを作成したりできます。

ノンリニアビデオ編集のように、Timeline インターフェースの各レイヤーはトラックです。複数のトラックを組み合わせることで、オーディオ、ゲームプレイシーケンス、カメラの角度、パーティクルエフェクトなどで構成されるシネマティックな特徴を作成できます。

スクリプト化されたシーケンスがゲームやレベルのある場面で重要になる場合、新しいドアの出現、パスのクリア、新しいビルやプラットフォームの登場など、イベントのシーケンスをトリガーすることができます。このコミュニティ記事で説明されているように、ゲームオブジェクトの **PlayableDirector** コンポーネントを呼び出してシーケンスをトリガーするには、タイムラインの使用を検討してください。



Unity の 2D デモ『Dragon Crashers』のシーンがタイムラインで調整されている様子

Unity ゲームデザイナーのためのプレイブック

このガイドは、デザイナーが Unity でゲームプレイをプロトタイプ化、作成、改良するために不可欠なツールを紹介しています。C#、Unity Visual Scripting でのスクリプティングを始める上での詳細と、入力制御、キャラクターコントローラー、グレーボクシングなど、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べます。

eBook をダウンロード

その他のリソース

— Working with Animation Clips

— アニメーションリギングを使用してアニメーションワークフローを改善する

— リグ間でのアニメーションの再利用とリターゲット

— ゲームプレイとストーリーテリングをタイムラインと融合：カットシーンとゲームグラフィックス

Unity Asset Store

Unity Asset Store には、すぐに使えるアセット、制作ツール、ゲームシステム、テンプレートが多数揃っています。特にゲーム、レベルデザイン、プリプロダクションの初期段階で、レベル、レベルデザインツール、アート、ゲームメカニクスが未完成な状態でも基本的なゲームプレイのアイデアを素早くテストし、イテレーションする必要がある場合に、貴重な時間節約リソースとなります。アセットストアを初めて利用する場合は、まずはこちらのガイドをご覧ください。

Unity Asset Store で入手可能なアセットのコラージュ

アセットストアは、プリプロダクション過程でアセットを閲覧、発見しやすいように分類されています。初めて使う場合のおすすめを紹介します。

— templates セクションでは、カスタマイゼーションの詳細な説明を含んだ完全なゲームテンプレートを提供しています。テンプレートをショートカットとして使用し、進行中も修正可能なより詳細なプロトタイプを作成します。例えば、完全なゲームループを設定してテストしつつ、ゲームの主要な柱を構築できます。

— マテリアルとテクスチャは、3D および 2D プロジェクト用に多数用意されています。レベルをブロックアウトする際、ゲームプレイを表示、または見栄えの良い小道具や環境を作るために、既製のマテリアルを使用して異なる表面を特定します。

— 3D または 2D アセットのセクションで、プリミティブ形状を最終バージョンのルックアンドフィールを模倣したアセットに交換します。環境からキャラクターや小道具まで、すべて用意されています。さらに、ビジュアルエフェクトやオーディオエフェクトは、実現したいムードを伝えるのに役立ちます。

— 素早くブロックアウトされたプロトタイプレベルで動くカプセルだけでは、ストーリーテリングやゲームプレイの意図を伝えるのは困難です。アニメーションセクションには、意図したデザインをより正確に表現することに役立つ、ダミーキャラクターを使ったアニメーションセットがあります。

パート III：UNITY のレベルデザインツール

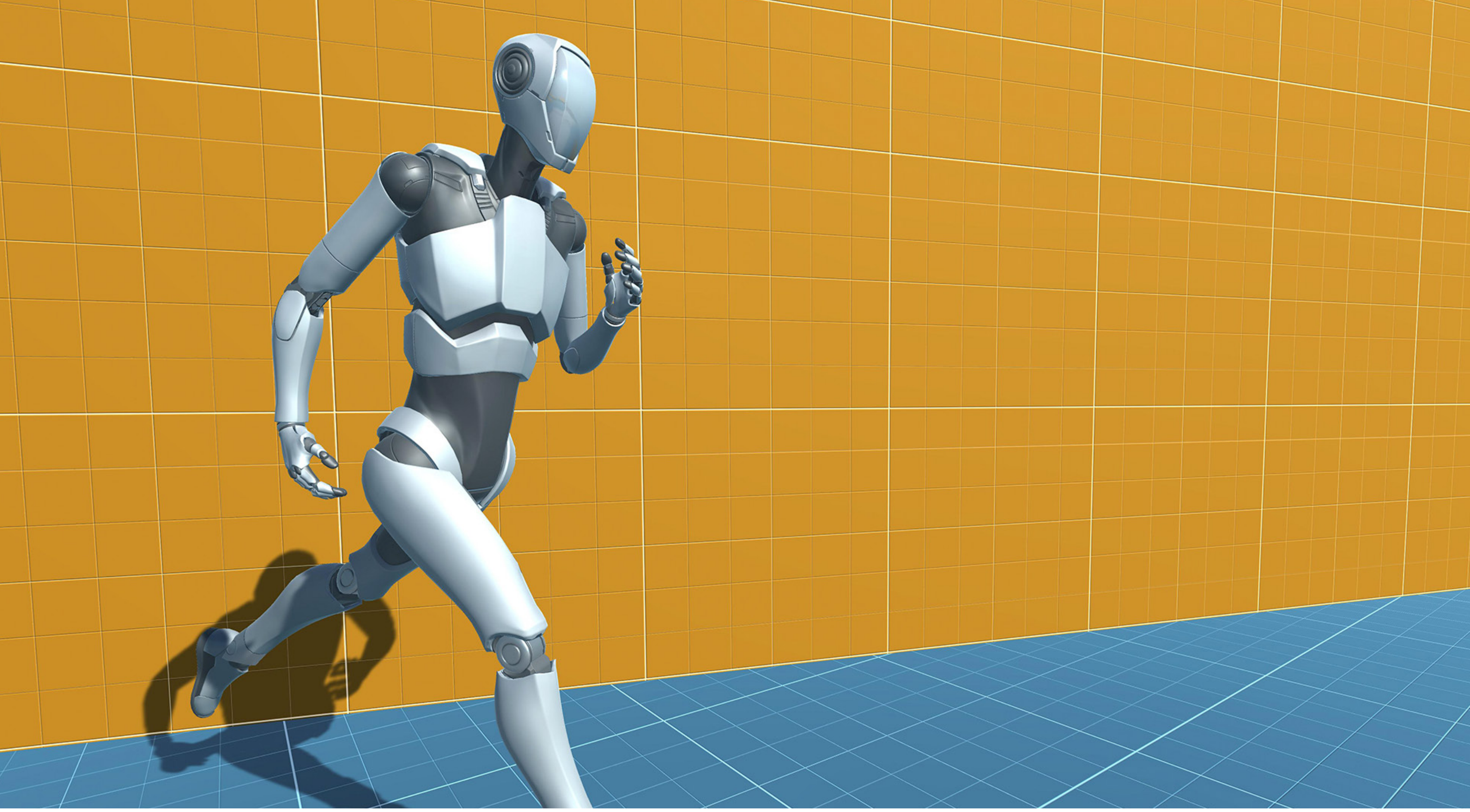
どのゲームも、レベルの構築方法に関して独自のニーズを持っています。マッチ 3 パズルゲームでは、レベルデザイナーが多くの動的要素が含まれるレベルの構築に集中できるよう、カスタムメイドのレベルデザインとデバッグツールが必要になる場合があります。他のゲームでは、大規模な屋外の没入型環境を作成するために、多くのツールを組み合わせる必要があるかもしれません。

Unity は、すぐに使えるレベルデザイン用の機能を数多く提供しています。また、Unity Asset Store では、よりカスタマイズされたニーズ向けの様々な制作ツールが入手可能です。このセクションでは、Unity の主なツールと、アセットストアで利用可能なリソースの一部を紹介します。

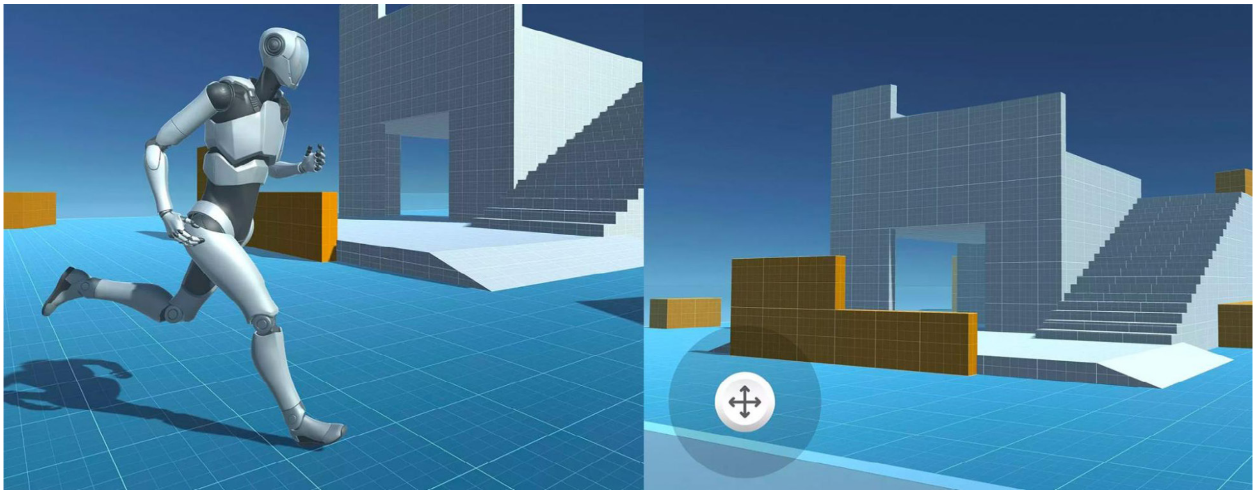
Starter Assets

新しい環境をテストする有効な方法は、プレイアブルキャラクターで操作することです。Starter Assets は、Unity が提供し、Unity Asset Store で入手可能な無料のアセットで、一人称と三人称カメラでプレイアブルキャラクターを使って素早くプロトタイプを作成するのに便利です。これらのパックは、あらゆるゲームジャンル向けの強固な基盤として機能するモジュール方式で構築されたコントローラーを提供します。

Starter Assets は Character Controller コンポーネントと Input System を使用して、走ったり、ジャンプしたり、周囲を見回したりといった動きをキャラクターにさせることができます。また、後述する Unity の強力なカメラシステムである Cinemachine も利用しています。

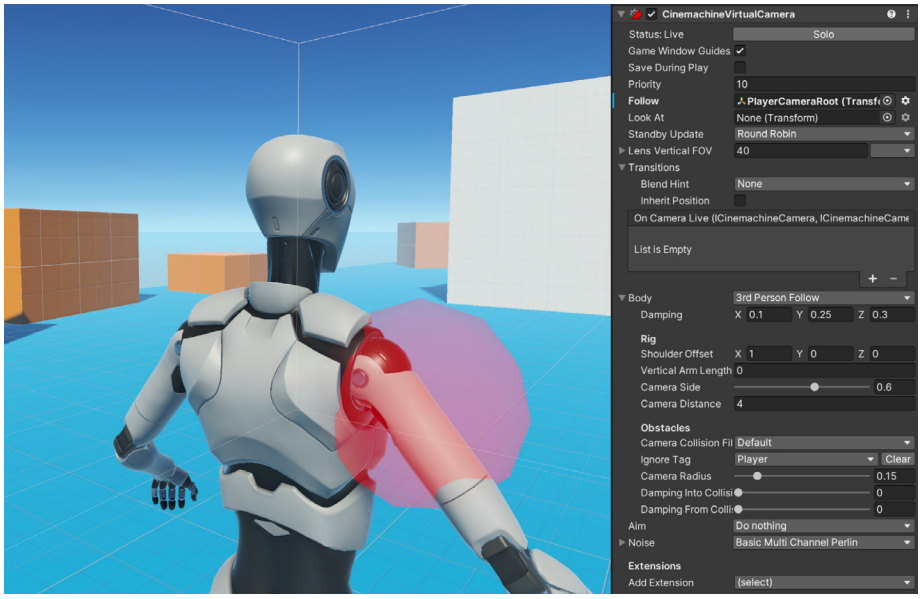


キャラクターのセットアップと準備を行いましょう。



一人称および三人称視点のアセットパックには、すべてのプラットフォーム用のコントロールが含まれています。Playground という名前のシーンを選択して、すぐに試してみましょう。

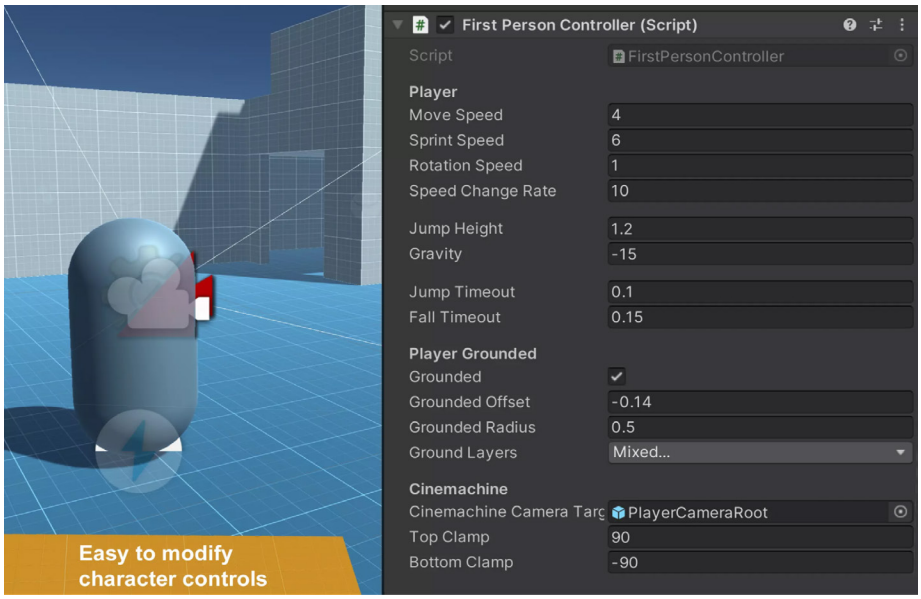
Starter Assets をプロジェクトにインポートして、マテリアルがマゼンタ色で表示された場合は、必ずユニバーサルレンダーパイプライン（URP）または HD レンダーパイプライン（HDRP）にアップグレードしてください。アップグレードを行うには、**Environment/Art/Materials** フォルダーに移動します。また、Materials フォルダー内のマテリアルを選択し、**「Window」>「Rendering」>「Render Pipeline Converter」**から「**Material Upgrade**」にチェックを入れ、**「Initialize Converters」>「Convert Assets」**を選択することもできます。



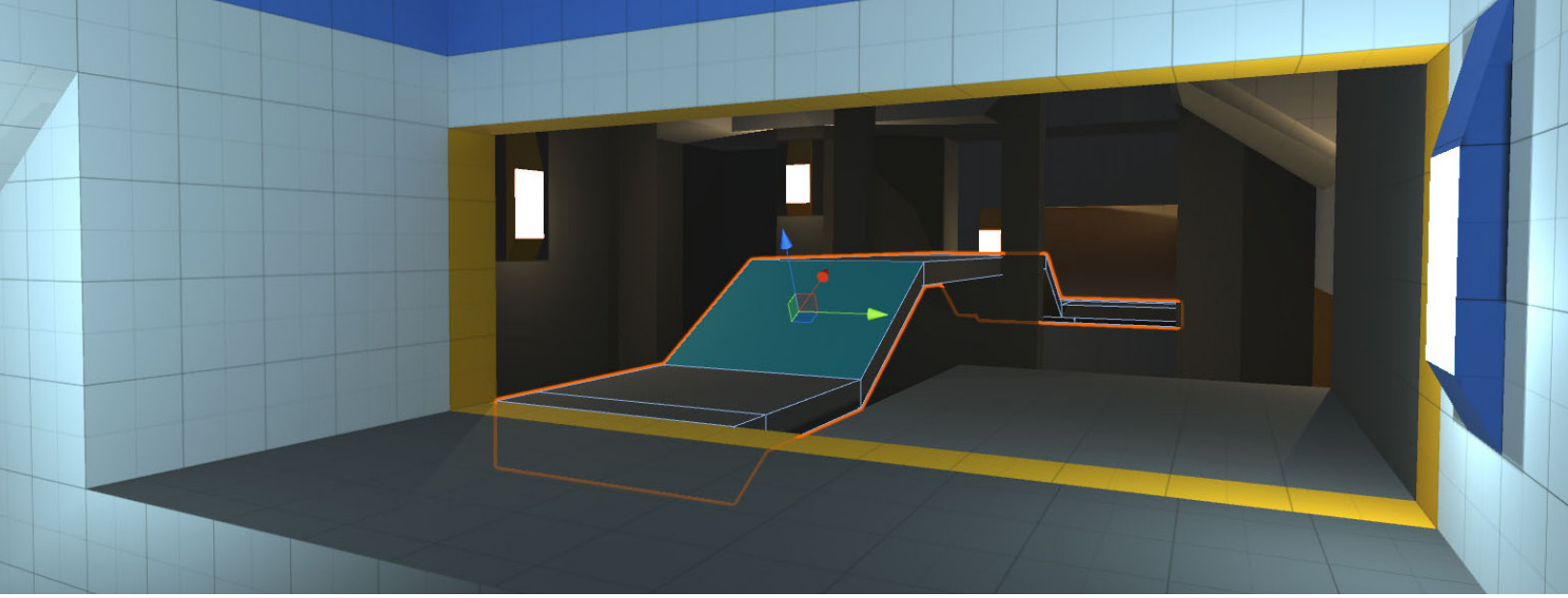
三人称視点のアセットパックは、Cinemachine カメラを使用し、カメラの障害物を避けるオプションなど、Third Person Follow 用に設計された設定があります。

一人称視点または三人称視点でレベルをテストする場合のために、同じプロジェクト内に両方のアセットを持つことができます。必ずレベルを両方の（**Playground** という名前の）シーンに移動するか、キャラクター、コントロール、カメラプレハブをシーンに移動してください。

各スターターアセットには、**StarterAssets** フォルダーにドキュメントが含まれています。Project フォルダーで **Playground** というシーンアセットを探して開き、そのアセットを使用できます。



一人称と三人称のコントローラースクリプトは、キャラクターの動きを調整するための値を公開します。Character Controller を使用し、単体では力に反応せず、アタッチされた BasicRigidBodyPush スクリプトのようにスクリプトで指示しない限り、自動的に Rigidbody コンポーネントを押しのけることもありません。

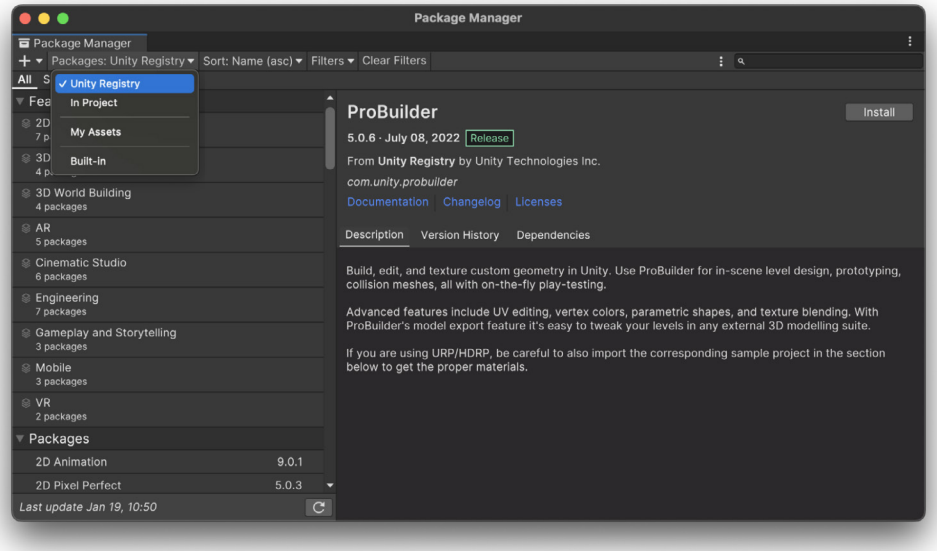


ビルトイン 3D モデリングツールの ProBuilder を使って Unity でレベルデザインを始めましょう。

ProBuilder

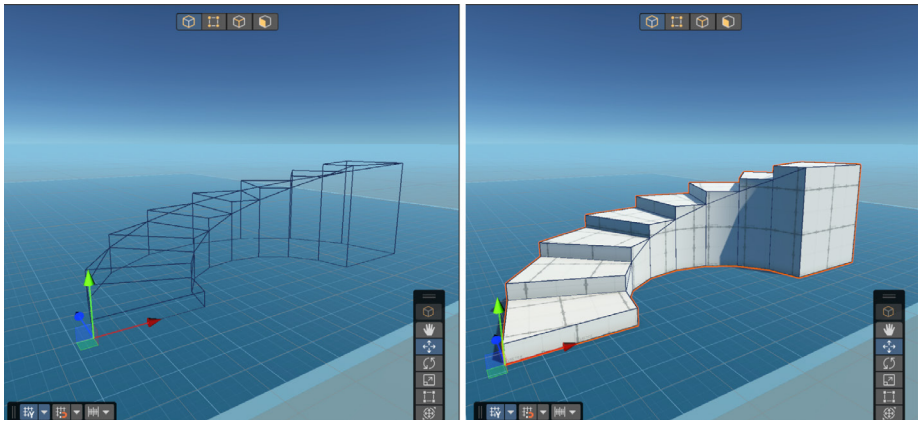
ProBuilder と Polybrush は、3D モデリングソフトウェアやプロの 3D アーティストに頼らずに、Unity でレベルをデザイン、グレーボックス、プロトタイプ、プレイテストできるツールです。ProBuilder でレベルをモデリングした後、Polybrush でテクスチャをペイントしたり、既存のメッシュをスカルプトしたりして、ディテールを追加します。

『SUPERHOT』の開発チームのように、ProBuilder で作成したレベルでゲームをリリースすることもできます。



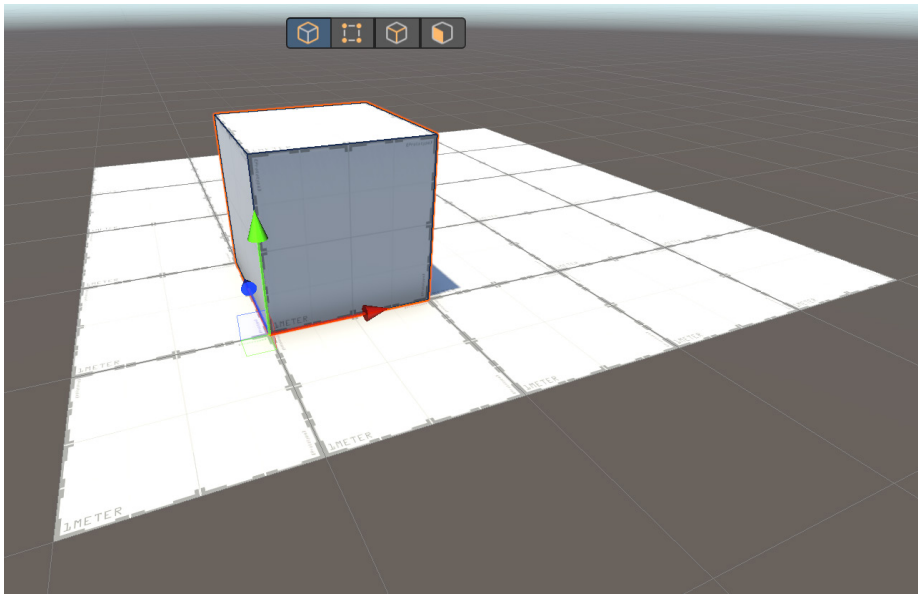
プロジェクトでレンダーパイプラインを使用する場合は、URP または HDRP のサポートファイルを忘れずにインストールしてください。

ProBuilder パッケージをインストールする際は、必ず URP または HDRP に対応するサポートファイル（マテリアル）をインポートしてください。これを怠ると、オブジェクトがレンダリングされない場合があります。ビルトインレンダーパイプラインを使用する場合、対応は必要ありません。



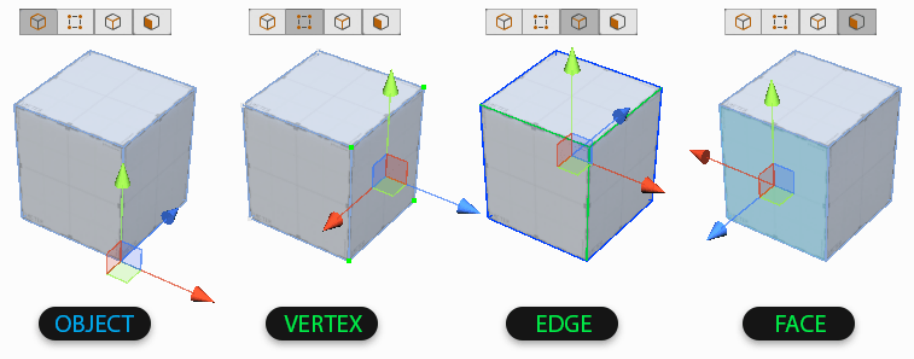
URP を使用した Unity 2022 LTS の新規プロジェクト、サポートファイルのインストール前とインストール後に作成された階段

トップメニューから、**「GameObject」>「ProBuilder」>「Cube」**（例）で ProBuilder プリミティブオブジェクトを作成します。ProBuilder には、クイックプロトタイピングに便利なプリミティブ形状が多数用意されています。物理演算用の Mesh Collider コンポーネントも含むこれらの形状は、Scene ビューで追加できます。



ProBuilder プリミティブは、1 Unity ユニット（現実世界の 1 メートルに相当）のグリッドパターンのテクスチャを持つマテリアルを使用します。これにより、オブジェクトのスケールを把握しやすくなり、現実世界の任意の空間の大きさをシーンで再現できます。

ProBuilder で作業するためのメインウィンドウは、**「Tools」>「ProBuilder」>「Probuilder Window」**内にあります。有効にすると、ProBuilder メッシュを編集するためのツールバーが Scene ビューに追加されます。



ProBuilder でメッシュを操作するには 4 つの方法があります。

— **オブジェクトモード：**ゲームオブジェクトを選択して操作する Unity の標準的なモード

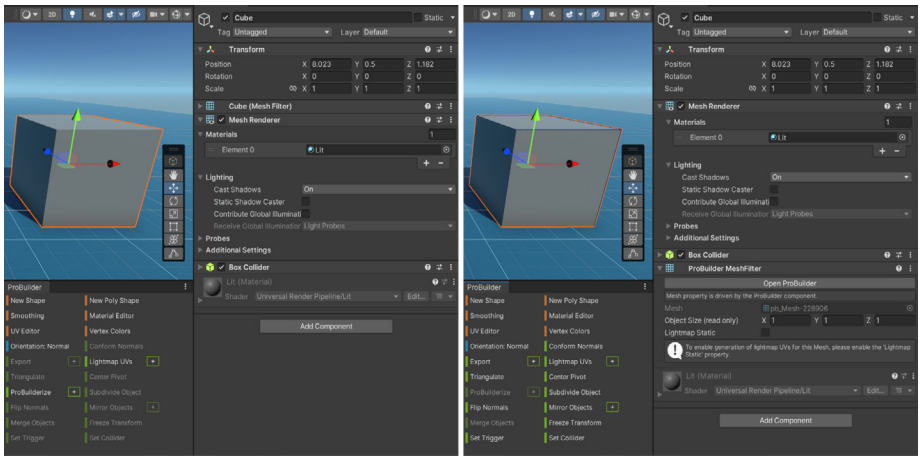
— **頂点モード：**頂点（ポイント）を選択して操作する要素モード

— **辺モード：**辺（線）を選択して操作する要素モード

— **面モード：**面（ポリゴン）を選択して操作する要素モード

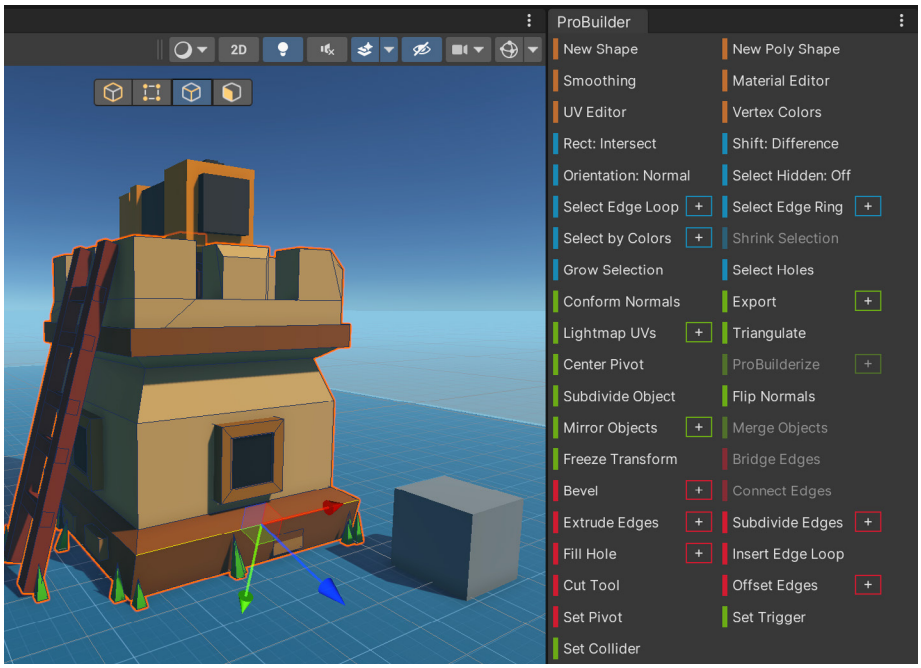
操作モードを素早く切り替えられるよう、ショートカットメニューから ProBuilder アクション用の ショートカットを設定することを検討しましょう。

ProBuilder のメッシュは、Unity の通常のゲームオブジェクトのように動作します。Transform 値を適用したり、コンポーネント、物理、スクリプトを追加したり、アニメーションをつけたりできます。ただし、標準の Unity メッシュは ProBuilder メッシュとは異なります。ProBuilder オブジェクトに変換するまで、ProBuilder で編集することはできません。



ProBuilderize アクションで ProBuilder メッシュに変換される前と後のゲームオブジェクトキューブ

ProBuilder メッシュは、ProBuilder ツールやアクションを一貫して使い、作成や編集により構築するのが一般的ですが、既存アセットの修正にも便利です。例えば、Unity Asset Store からアセットをインポートして、ProBuilder で修正できます。



Unity Asset Store からインポートしたタワーアセット。ProBuilderize アクションで変換すると、ProBuilder メッシュのように修正できる。

メインウィンドウ「ProBuilder」は、ツールがタイプごとに色分けされていて、選択しやすくなっています。

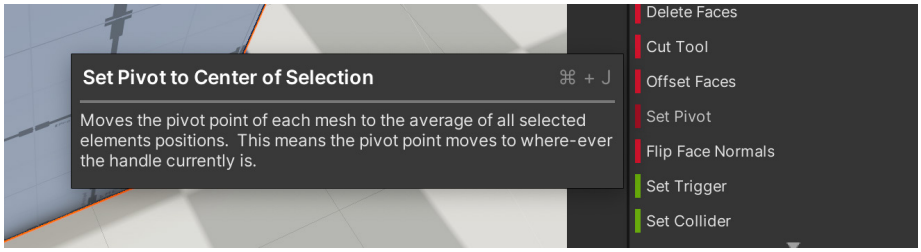
— **オレンジ**：ツールパネル（個別に開くウィンドウ）

— **青**：選択を設定、変更する機能

— **緑**：オブジェクト全体に影響するメッシュ編集アクション

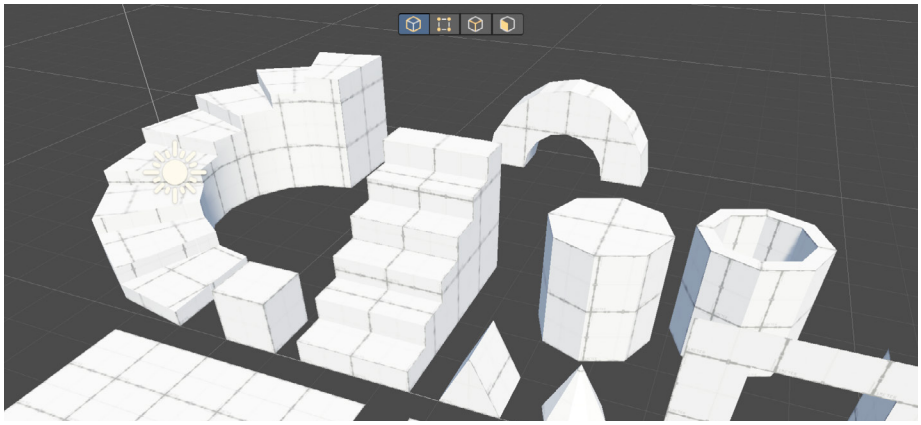
— **赤**：頂点、辺、面ジオメトリに作用するメッシュ編集機能

ProBuilder には、機能の追加情報にアクセスできるツールチップも含まれていて、ツール上にカーソルを置くことで表示できます。

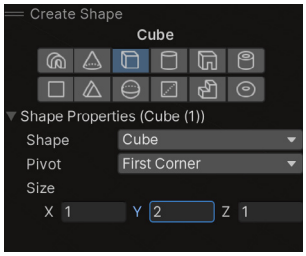
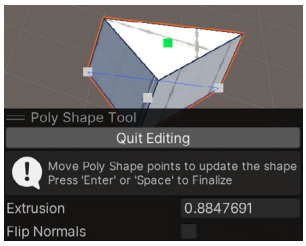
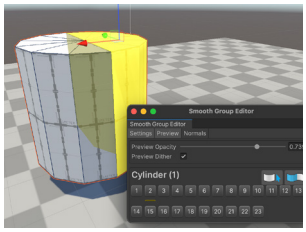


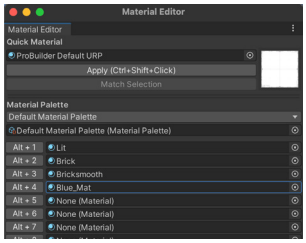
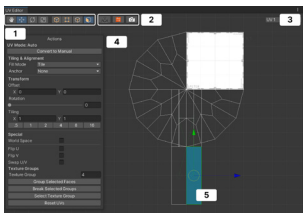
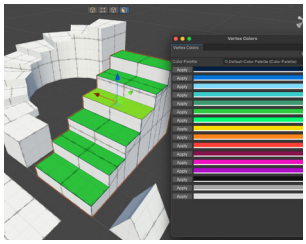
ツールチップは ProBuilder ツールの上にマウスカーソルを置くと表示され、ツールの横の + 記号は追加オプションを示す。

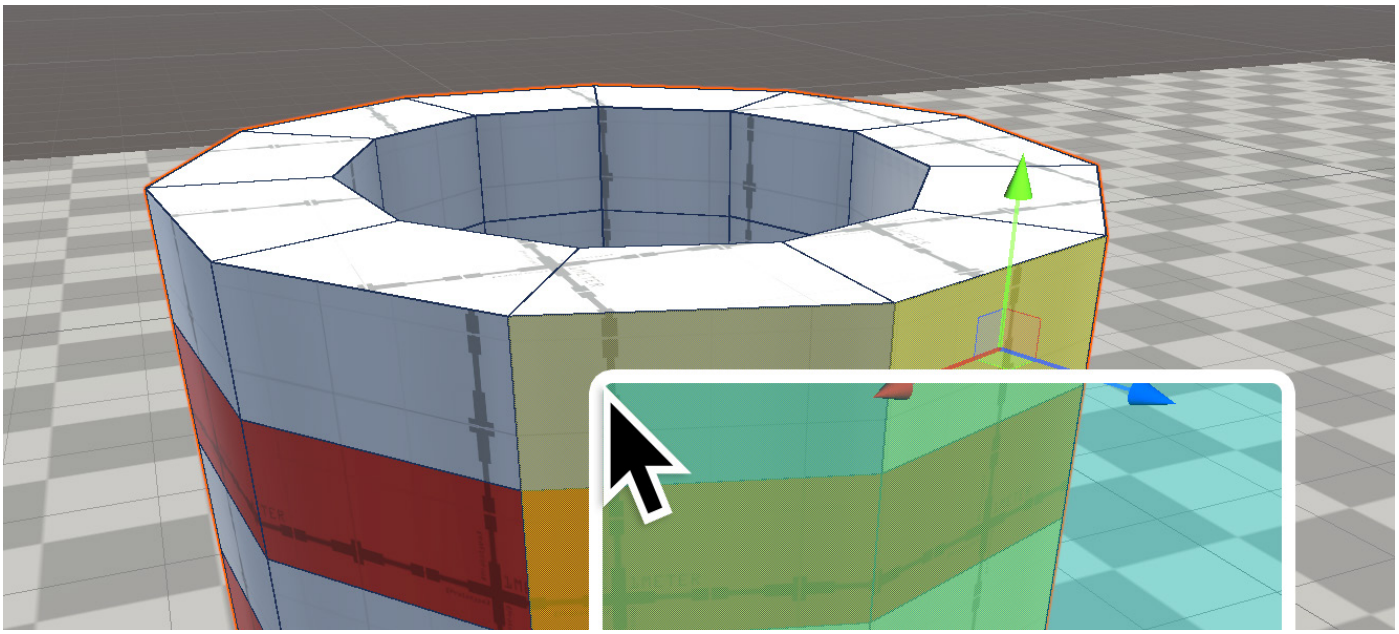
以下の表は、ProBuilder の各ツールの機能についての情報です。これらのツールの多くには、調整可能なオプションが含まれることを示す + 記号が横に表示されています。ここに記載されているツールのいくつかは、このセクションの後半で詳しく説明します。



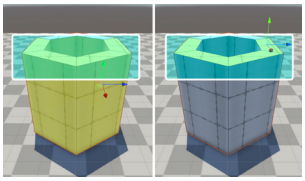
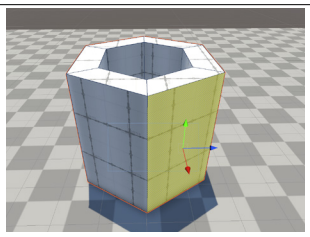
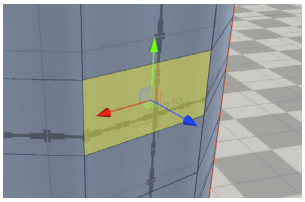
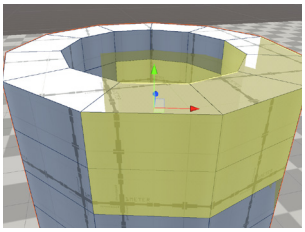
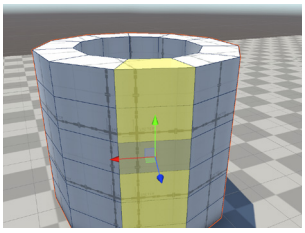
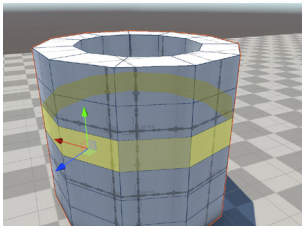
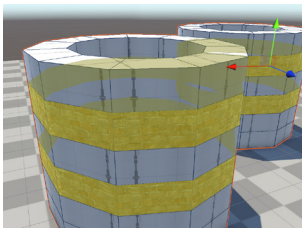
ProBuilder に含まれるデフォルト形状の例

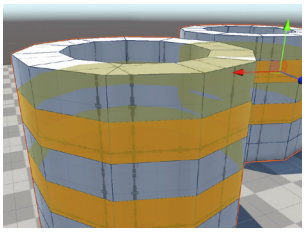
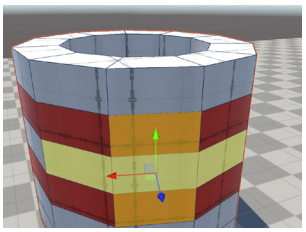
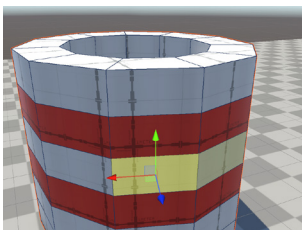
色	名称	参照画像	説明
	New Shape		— ProBuilder でレベルを作成する際に使用する最も一般的なツール — 形状を 1 つ選択した後、任意のピボット位置を選択し、Scene ビューまたはこのウィンドウでサイズを定義する — メッシュの定義を調整するために、形状に基づいた追加オプションが利用可能
	New Poly Shape		— カスタム 2D 形状を作成し、3D メッシュに押し出す — 不規則な構造を素早く構築するのに最適 — 図面を基に 3D マップを作成する際にも便利
	Smoothing		— ポリゴングループのスムージングを行い、滑らかで均一な表面を形成する — 詳細はこちら

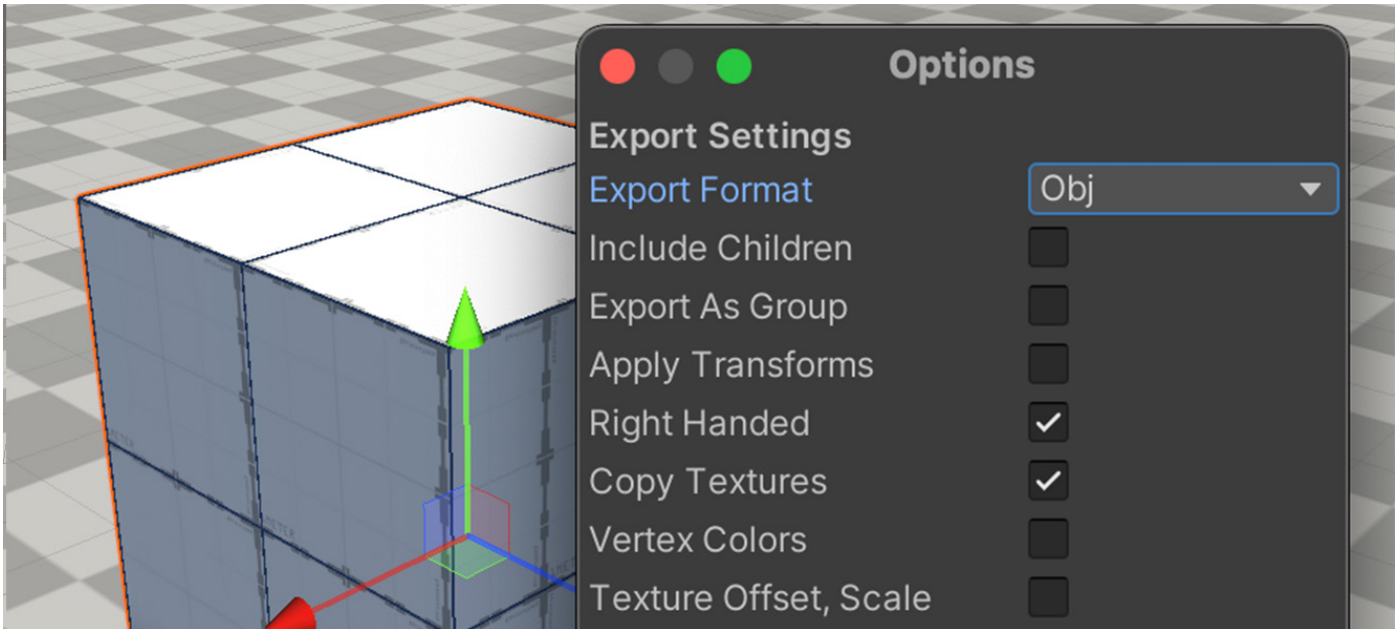
	Material Editor		— 定義されたマテリアルライブラリからオブジェクトまたは面にマテリアルをすばやく適用する — プロトタイプでマテリアルを適用し、意図や機能を示すことで、デザインのアイデアをより明確に伝えることができる
	UV Editor		— スムージング機能に似ている。より洗練されたプロトタイプやレベルデザインに最適 — 詳細はこちら
	Vertex Colors		— マテリアルエディターに相当するが、テクスチャのないモデル用 — 面、頂点、または辺に色を適用し、レベルデザインプロトタイプに情報を追加する



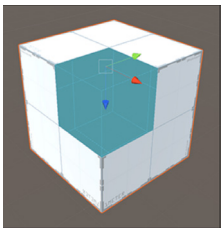
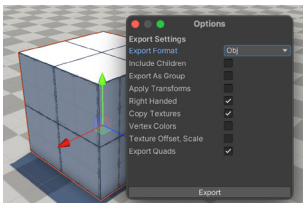
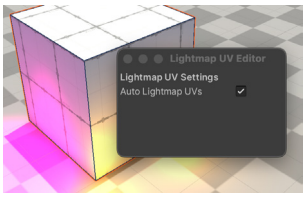
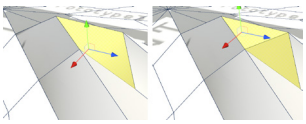
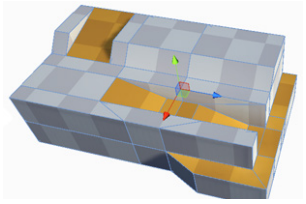
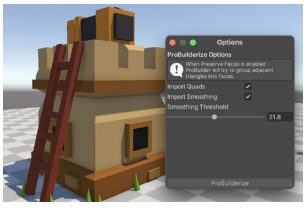
一部の機能を有効にするには、面などのメッシュの要素が選択済みである必要があります。

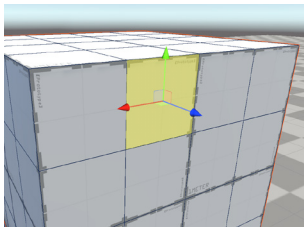
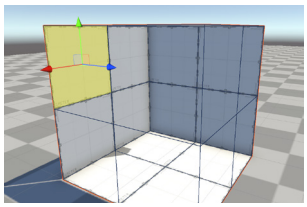
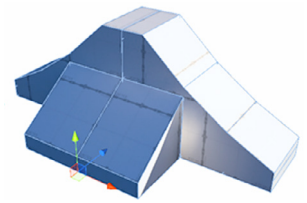
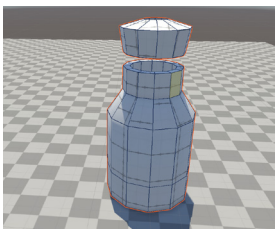
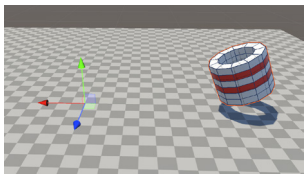
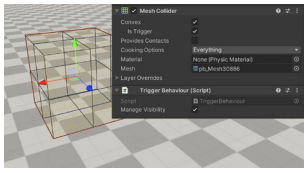
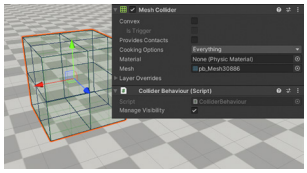
色	名称	参照画像	説明
	Rect		— ドラッグ選択で、長方形内に完全に収まっている要素のみを選択するか、部分的に入っている要素も選択するかを決めるのに使用する
	Shift		— ドラッグ選択した際、頂点、面、辺の選択に適用するアクションを変更する — 選択した要素を追加、減算、または反転する
	Orientation		— オブジェクトや要素を選択する際のハンドルの向きを設定する — Global はワールドの向き、Local はオブジェクトの回転、Normal は面や頂点の法線に従う
	Select Hidden		— ドラッグ選択した際、メッシュの奥にあるような、カメラから隠れた要素を選択または無視する
	Select Face Loop		— 面が選択されると、選択範囲を縦方向に連続するすべての面に拡大する
	Select Face Ring		— 面が選択されると、選択範囲を水平方向に連続するすべての面に拡大する
	Select by Material		— 選択されている面と同じマテリアルを持つシーン内のすべての面を選択する — ポップアップメニューの「**Current Selection**」を有効にすると、選択範囲を現在のオブジェクトに限定できる

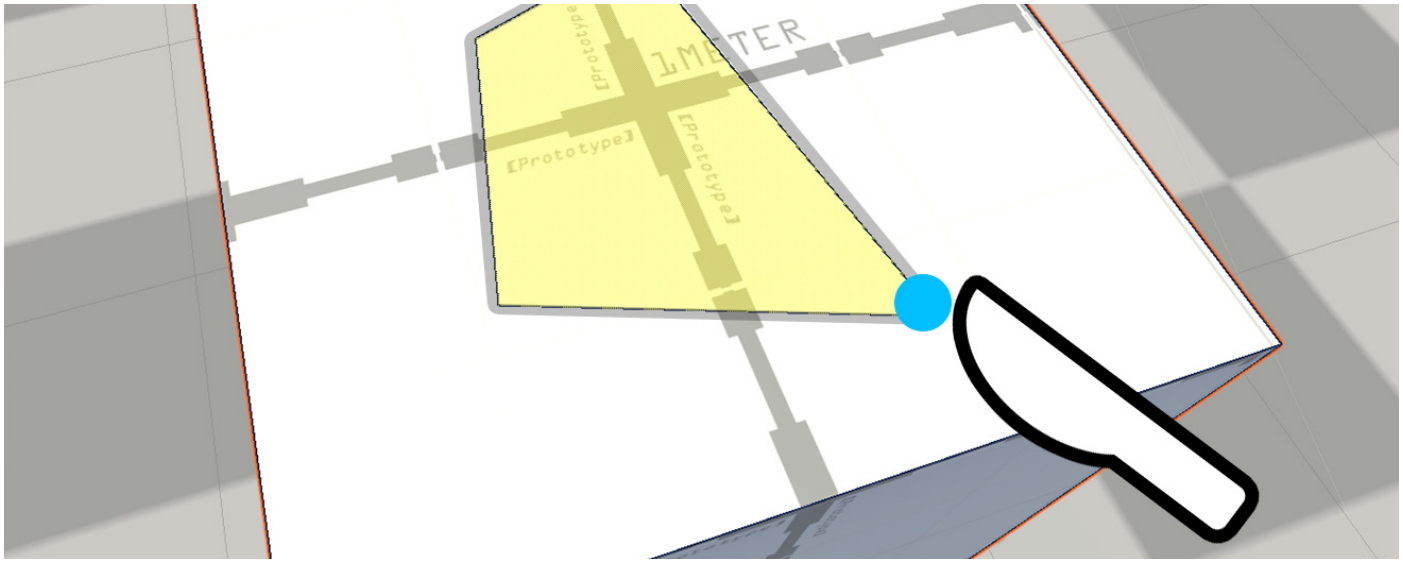
	Select by Colors		—	Select by Material と同様に機能するが、頂点カラーを使用する
	Grow Selection		—	隣接する面、辺、または頂点へと選択範囲を外側に拡大する。選択範囲を指定した角度内の要素に限定する
	Shrink Selection		—	現在の選択範囲の外周上の要素を削除する



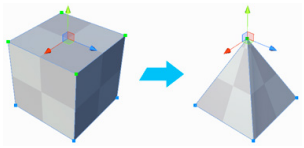
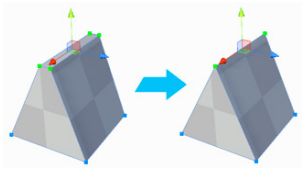
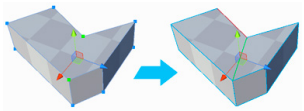
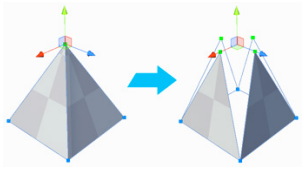
ProBuilder オブジェクトのメッシュ編集機能は、オブジェクトまたはその要素の 1 つを選択すると使用できるようになります。

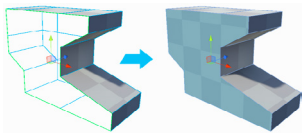
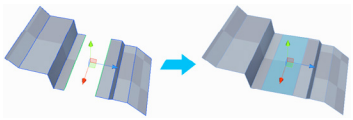
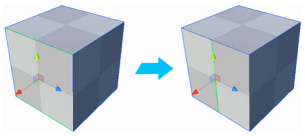
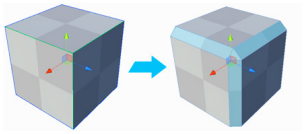
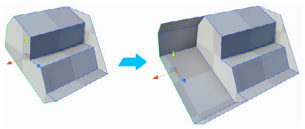
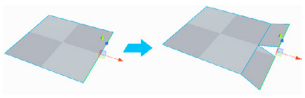
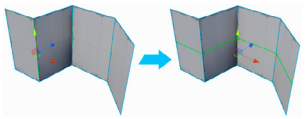
色	名称	参照画像	説明
	Conform Normals		— 三角形の配列順序のミスマッチを修正する — 選択された面の中で間違った方向を向いている面の方向を修正する
	Export		— 選択したオブジェクトを OBJ、STL、PLY、Unity プレハブなどの 3D モデルファイルにエクスポートする — アーティストが選択した DCC で、プロトタイプされた 3D モデルのリファレンスを持つことを可能にする — 詳細はこちら
	Lightmap UVs		— デフォルトでは、メッシュの作業中、ライトマップ UV が自動更新される — これを無効にして、ライトベイク前に手動で行うことも可能
	Triangulate		— 面を基本三角形に分割し、角張った滑らかでない外観を作成する — 必要に応じて、スムージンググループを使用してスムージングを再適用する
	Center Pivot		— メッシュのピボットポイントをオブジェクトの境界の中心に移動する
	ProBuilderize		— 選択した 3D オブジェクトを ProBuilder で編集可能なバージョンに変換する — 既存のアセットの修正やレベルプロトタイプへの追加に便利

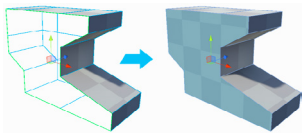
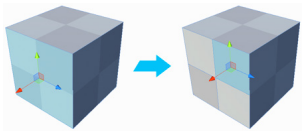
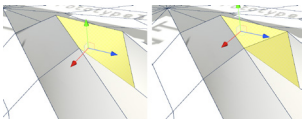
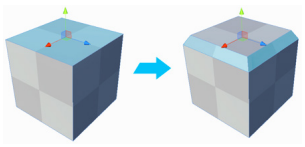
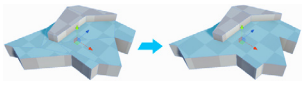
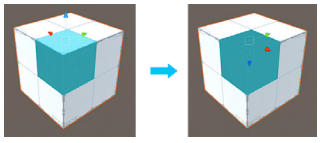
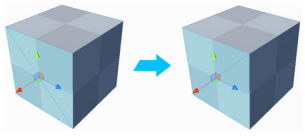
	Subdivide		—	選択したオブジェクトのすべての面を分割し、モデリング時により多くの詳細を追加できるようにする
	Flip Normals		—	選択したオブジェクトの法線のみを反転する
	Mirror Objects		—	オブジェクトのミラーコピーを作成する。左右対称のアイテムを作成する場合に便利
	Merge Objects		—	選択した 2 つ以上の ProBuilder オブジェクトをマージして 1 つにする
	Freeze Transform		—	オブジェクトのピボットポイントを現在のワールドの原点位置（0,0,0）に設定し、回転とスケールをリセットする
	Set Trigger		—	選択したオブジェクトに Trigger Behavior スクリプトを割り当て、ゲーム内でトリガーコライダーにする
	Set Collider		—	選択したオブジェクトに Collider Behavior スクリプトを割り当て、ゲーム内でコライダーにする

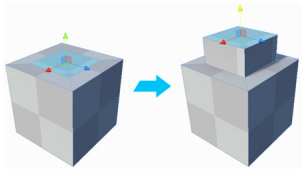
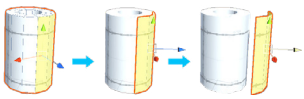
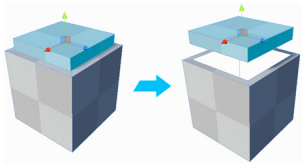
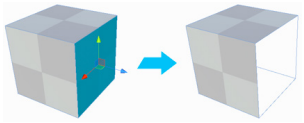
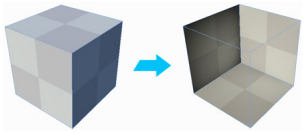
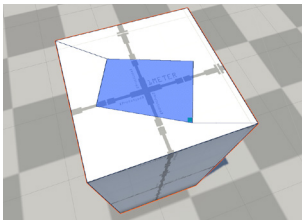


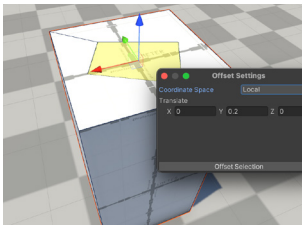
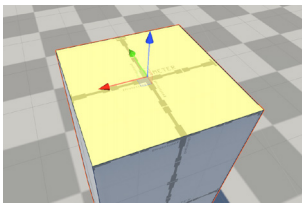
頂点、辺、面ジオメトリに作用するメッシュ編集機能

色	名称	参照画像	説明
		頂点の操作	
	Collapse Vertices		— 選択されたすべての頂点を中心点または最初に選択された頂点の位置に合わせる
	Weld Vertices		— 選択した頂点を指定した距離内でマージする — 近距離のジオメトリ同士の最適化に役立つ
	Connect Vertices		— 選択された頂点を接続する新しい辺を作成する
	Split Vertices		— 頂点を個々の頂点（隣接する面ごとに 1 つずつ）に分割し、各頂点を選択したときに面を個別に移動できるようにする

	Fill Hole		— 選択した頂点に接する穴を埋める新しい面を作成する
辺の操作			
	Bridge Edges		— 選択された 2 つの辺の間に新しい面を作成する
	Connect Edges		— 選択された各辺の中心を結ぶ辺を挿入する
	Bevel		— 選択された辺を 2 つに分割し、その間に新しい面を作成する
	Extrude Edges		— 選択された各辺から新しい辺を押し出し、各辺を繋ぐ新しい面を形成する — 開いている辺にのみ適用可能
	Subdivide Edges		— 選択された辺を複数の辺（デフォルトでは 2 つ）に分割する
	Insert Edge Loop		— 選択された辺から、新しい辺ループ（クアッドで直接接続された辺の連続）を追加する

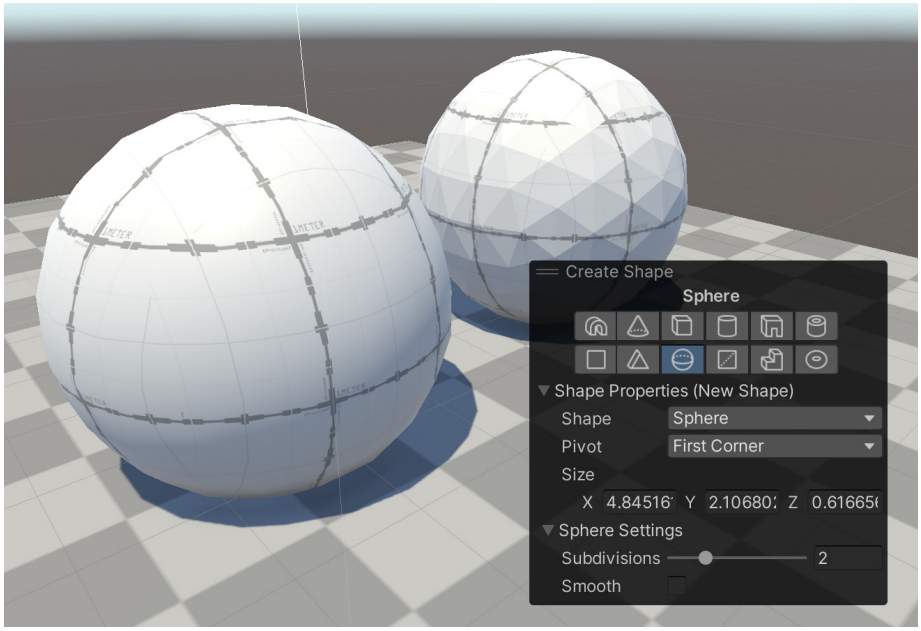
	Fill Hole（Edges）		— 選択された辺に接するすべての穴を埋める新しい面を作成する
ポリゴン面の操作			
	Subdivide Faces		— 選択されている各面を分割し、各辺の中心に頂点を追加して、中心で接続する
	Triangulate Faces		— 選択された面を基本三角形に分割し、角張った滑らかでない外観を作成する — 必要に応じて、スムージンググループを使用してスムージングを再適用する
	Bevel Faces		— 選択された面の辺を 2 つの辺に分割し、その間に新しい面を作成する
	Merge Faces		— 選択した面を 1 つの面にマージし、分割辺を削除する
	Conform Normals		— 三角形の配列順序のミスマッチを修正する — 選択された面の中で間違った方向を向いている面の方向を修正する
	Flip Face Edge		— 選択された 4 つの辺を持つ面の、三角形の方向を入れ替えることで、四角形の中心の辺の方向が反転する

	Extrude Faces		— 現在選択されている面を引き出し、各辺に側面を加えて新しい面を作成する — デフォルトでは、新しい面はそれぞれその頂点法線の方向を向く
	Duplicate Faces		— 選択されている各面をコピーし、新しいメッシュを作成する
	Detach Faces		— 選択された面をメッシュの他の部分から分離する
	Delete Faces		— 選択した面を削除する
	Flip Face Normals		— 選択した面の法線のみを反転する。部屋を作ったり、壁を外側のファサードから内側のファサードに切り替えたりする際に便利
共通の操作			
	Cut Tool		— 面を分割する。点で切り出す形状を定義すると、それがメッシュ上の新しい面になる

	Offset		— 頂点、辺、面を移動する正確な値を入力できる
	Set Pivot		— このメッシュのピボットポイントを、選択した頂点、辺、面の平均中心に移動する

スムージンググループ

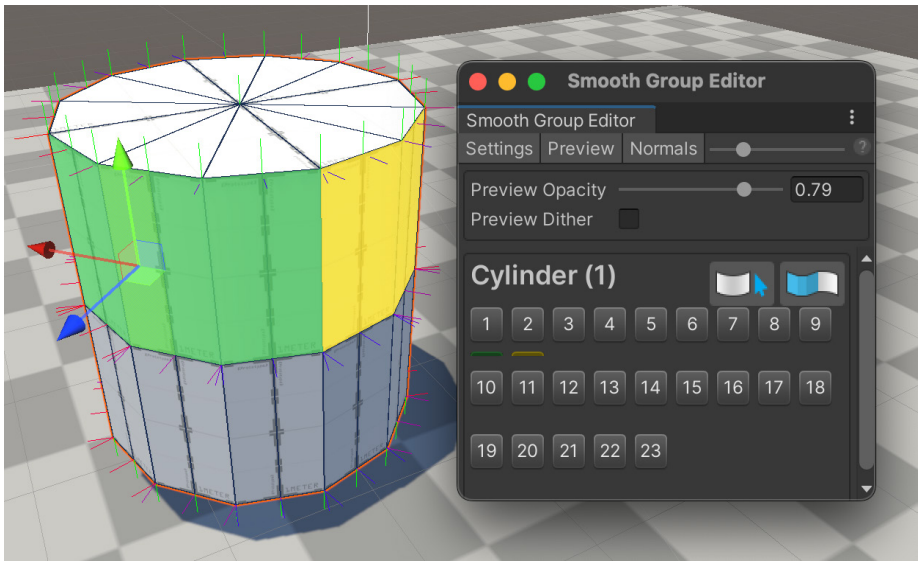
スムージンググループは、面と面の間にシャープまたはソフトエッジを作るのに役立ちます。隣接するポリゴンが同じスムージンググループに追加されていない場合、その間にハードエッジができます。車のフロントガラスを構成するポリゴンを想像してみてください。これらは 1 つのスムージンググループを構成し、ボンネットを構成するポリゴンは別のグループになります。フロントガラスとボンネットのポリゴンがすべて同じメッシュの一部である場合、両方ともスムージングされますが、同じ表面としては扱われません。



2 つ目のスフィアは Smooth オプションが無効化されている。この設定は Inspector からオブジェクトを選択することで変更可能。

ProBuilder では、新しい形状はデフォルトですべての面がスムージンググループされています。スムージンググループのより細かい調整を可能にするには、スムージングオプションのチェックを外します（上記画像参照）。

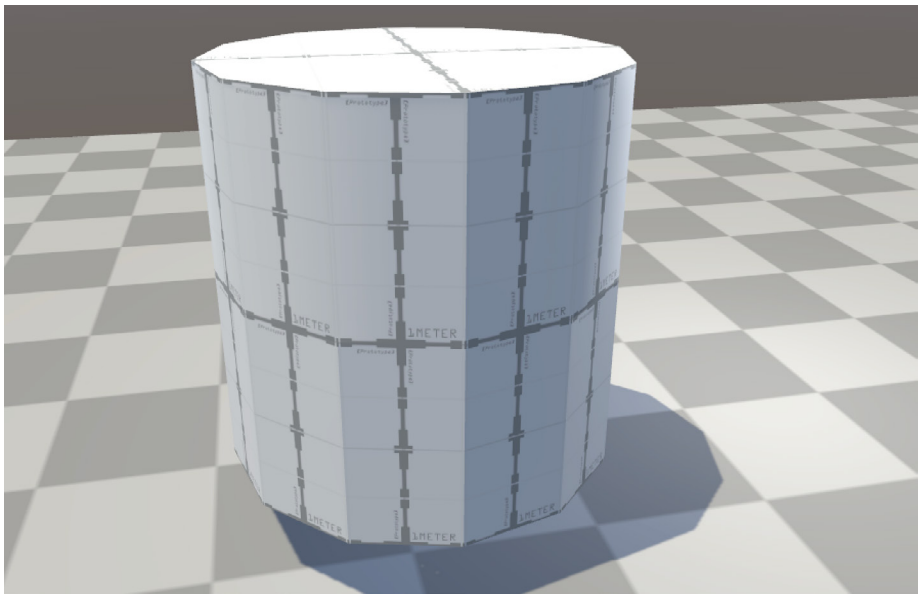
「Smooth Group Editor」ウィンドウには、様々なスムージンググループの設定やプレビジュアライゼーションに役立つオプションがあります。



「Smooth Group Editor」ウィンドウを開くと、メッシュは異なるグループをカラーコード表示する。

「Preview」オプションでは、スムージンググループごとの面のカラーコードの透明度を調整し、ディザ効果で馴染ませることができます。頂点の法線のプレビジュアライゼーションも可能です。

上述の例は、Game ビューでは以下のように表示されます。



円柱の上部と下半分がスムージングされておらず、面のエッジが見えるようになっている。上の 2 つのパーツはスムージングされているが、中央には境目が見えている。これは、2 つのパーツが異なるスムージンググループに属していることで、異なるサーフェスとして扱われるために起こる。

スムージンググループの設定手順を見てみましょう。

1. 同じグループに属するオブジェクトの面を選択し、同じ表面として扱われるようにします。
2. 「Smooth Group Editor」ウィンドウの 1 から 23 までのボタンのいずれかをクリックすると、選択した面がそのグループに割り当てられます。クリックしたボタンの下に色が付いた細い線が表示されます。これはグループが作成されたことを示し、線の色はグループのプリビジュアライゼーションの色と一致します。上の画像のグループ 2 に属する面は、さらに上の方にあるプリビジュアライゼーション画像では、ボタンの下の線と同様に黄色で表示されています。
3. 別の面のセットを選択し、新しいボタンをクリックして、異なる色の新しいグループに割り当てます。この手順を必要に応じて繰り返します。

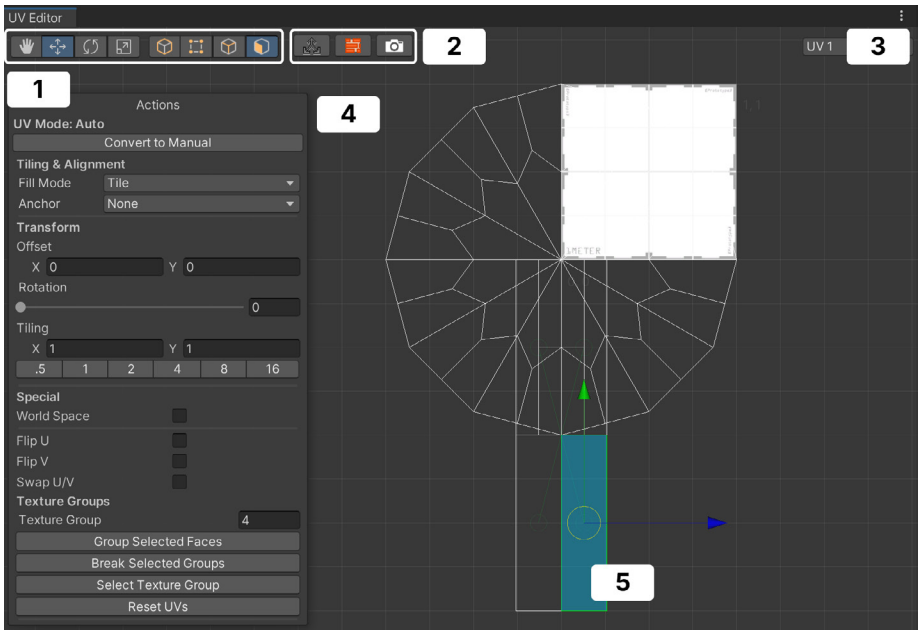
グループの 1 つに属する面を選択するには、2 つのオプションがあります。

— 選択したいグループのボタンを右クリックします。

— オブジェクト内の面の 1 つを選択し、青い矢印のボタンをクリックすると、選択範囲が同じグループに属するすべての面に拡大されます。

グループを削除するには、前述のように同じグループのすべての面を選択し、青い面のボタンをクリックします。

UV Editor



円筒形のオブジェクトの「UV Editor」ウィンドウ。デフォルトの UV マッピングが表示されている。

UV Editor で選択したメッシュのテクスチャマッピングを管理できます。主な作業領域を確認しましょう。

1. これは Scene ビューのものと同じツールバーで、頂点、辺、面を移動、回転、スケールするオプションがあります。

2. これらのボタンを有効にすると、Scene ビューから直接 UV マッピング座標を操作できます。アクティブ状態では色付きで表示され、それ以外の状態では灰色で表示されます。
   a. 矢印のボタンは、Scene ビューのトランスフォームツールをロックし、Scene ビュー内の選択された要素（辺、頂点、面、オブジェクト）に変更を加えることなく UV を操作できます。

   b. レンガのボタンは、座標（0,0）にあるシェーダーのテクスチャのプレビューを有効にします。

   c. カメラのボタンは、DCC ソフトウェアで画像を編集するときに参照できるように、ポリゴンワイヤーフレームをテクスチャに重ねて出力します。デフォルトでは、このファイルはプロジェクトの Assets フォルダーに保存されます。

3. シェーダー用に UV マッピングを編集するには、UV を選択します。UV2（読み取り専用）を選択すると、ベイクしたライトマップやリアルタイムのライトマップを再生成できます。

4. 選択したオブジェクトの UV マッピングを行うには、以下の 2 つの方法があります。

   a.**Auto**：メッシュのサイズを変更する場合も、ProBuilder は Actions パネルの設定に従ってテクスチャマッピングを管理します。これはデフォルトのオプションで、ほとんどのレベルデザイン作業には十分でしょう（特にプロトタイピング目的で繰り返しパターンを扱うだけの場合）。

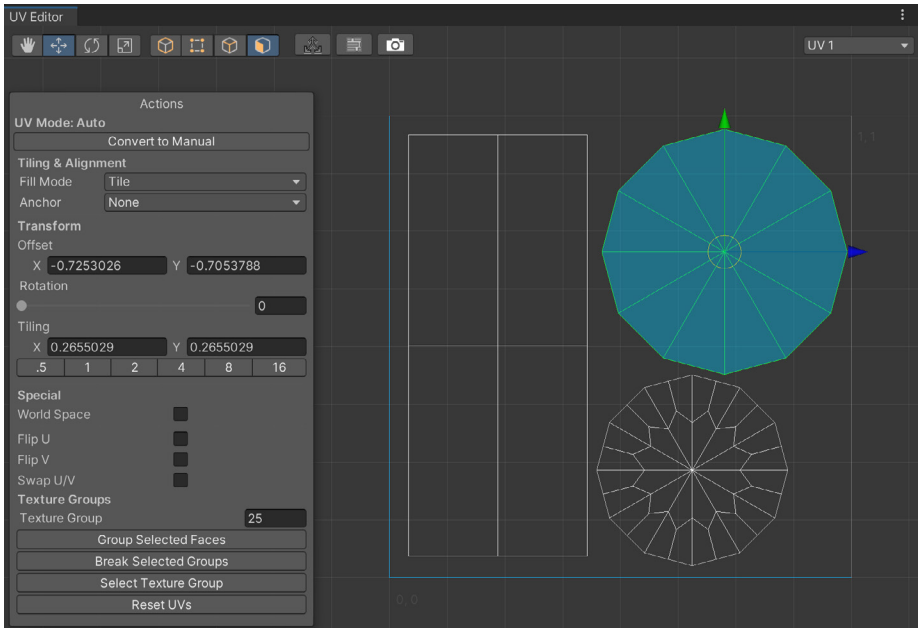
   b.**Manual**：この方法では、UV のアンラップや編集、UV のレンダリングなどを正確に行うことができます。詳細な画像に対して UV 要素を配置する際に推奨されます。このチュートリアル で、マニュアル UV による高度なテクスチャリングについて順を追って説明しています。

5. 要素を操作して、シェーダーのテクスチャにフィットするようにきれいに配置できます。上の画像では、選択された面が青くハイライトされています。

テクスチャリングの簡単エクササイズ

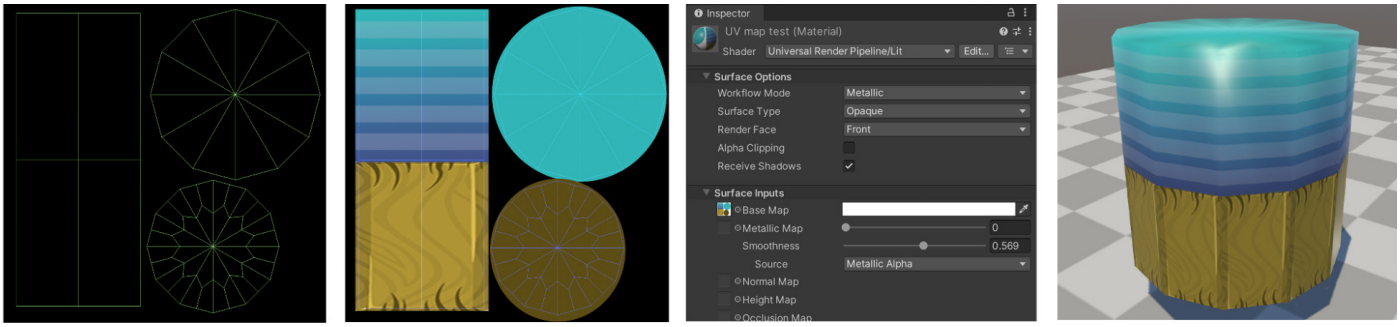
Auto モードで作業する場合、スムージンググループのように要素をグループ化すると、グループ化されたパーツを 1 つの要素として操作できます。これは、選択した要素を移動、スケール、回転するのに便利です。

このエクササイズでは、円柱の上面、下面、側面に対して 3 つのグループを作成しました。このオブジェクトには 1 つのテクスチャが使用され、UV はテクスチャにマッピングされます。これを行うために、3 つのグループを選択し、テクスチャを表す正方形の領域内に再分配しました。



円柱の上部の面はグループ化されているため、まとめて移動やスケールすることができます。テクスチャは後で新しい UV マッピングに基づいて作成されるため、テクスチャプレビューは無効になっています。このエクササイズで、デザイナーが変更に不満がある場合、**Reset UV** オプションでデフォルトの UV マッピングに変更できます。

新しい UV マッピングの準備ができたら、テクスチャをエクスポートして、任意の DCC ツールでオブジェクト上にペイントします。



レベルデザインのプロトタイピングで、オブジェクトに素早くテクスチャを貼る方法を紹介する簡単なエクササイズ。

上の画像で説明されている各ステップを左から順番に見ていきましょう。

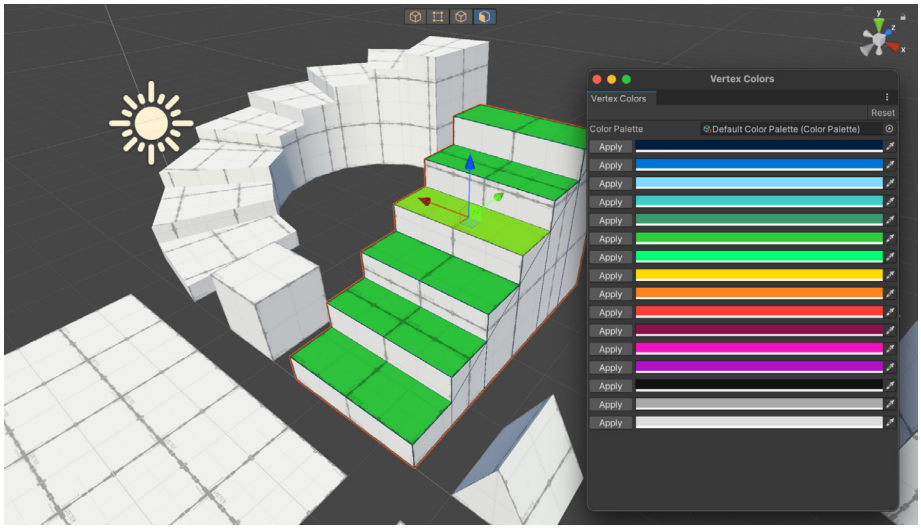
一番左：ペイントするためにテクスチャ UV マップが DCC ツールにインポートされます。
中央左：ワイヤーフレームがペイント時の参照用に使用されています。
中央右：画像ファイルが Unity に再インポートされ、このテクスチャを BaseMap としてマテリアルが作成されます。Unity プロジェクト内にあり、マテリアルに使用されている限り、テクスチャに対して繰り返し作業を行うことができます。画像に変更を加えて保存するたびに自動的に更新されます。
一番右：マテリアルが ProBuilder オブジェクトに適用されます。

UV エディターで利用可能な Auto モードと Manual モードの各オプションの詳細については、ドキュメントを参照してください。

ヒント：カラーコーディングでレベルデザインを迅速化

ステージのボクシングにカラーコーディング（色分け）を用いると、意図やアイデアをより明確に伝えることができます。例えば、破壊可能な要素を赤で色分けすることで、チームの他のメンバーにどの要素が破壊可能かを伝えることができます。

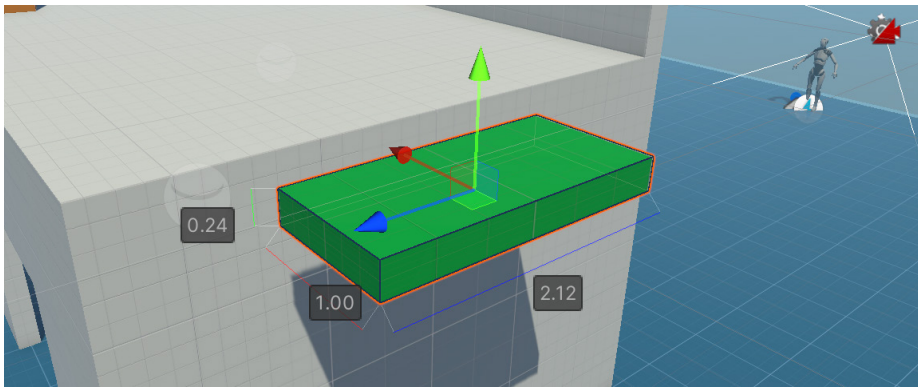
Vertex Color 機能で、「プラス」アイコンを選択してカラーパレットを作成します。パレットをカスタマイズして、シーンに必要な色（と色の数）を定義します。オブジェクトを色付けするには、シーンでオブジェクトを選択して「Apply」をクリックします。個々の面にも色を適用し、保存した色のパレットをチームと共有することで、全員が同じ色分け基準を使用するようにできます。



色分けされたグレーボックスシーン

ヒント：ProBuilder で寸法オーバーレイを有効にする

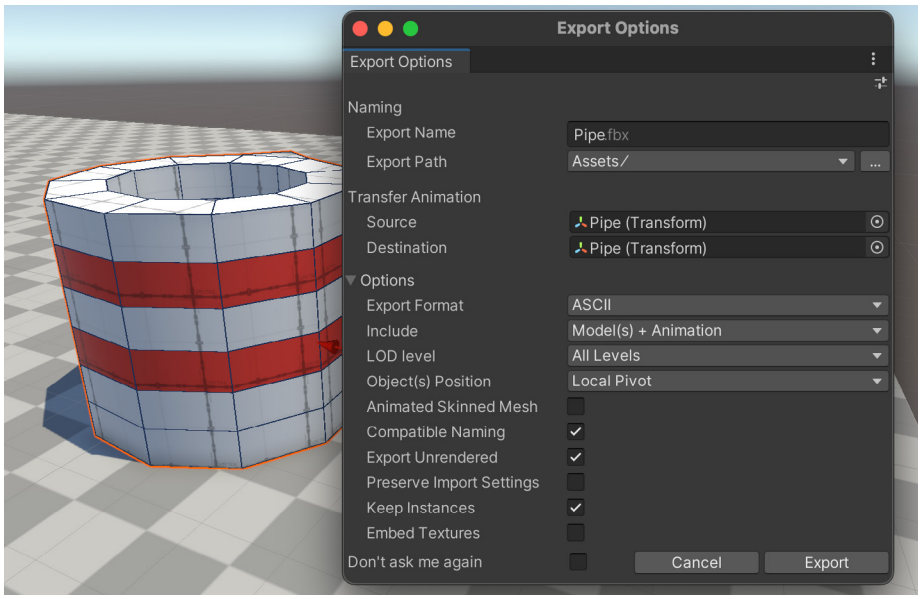
「Tools」>「ProBuilder」>「Dimensions Overlay」から、シーンで現在選択されているオブジェクトのサイズを示すフローティングラベルを有効化できます。



選択されたオブジェクトの高さ、幅、奥行きの便利なビジュアライゼーション

ProBuilder レベルを環境アーティストと共有する

FBX Exporter パッケージをインストールすると、レベルアセットのプロトタイプを正しい寸法で DCC アプリケーションにエクスポートし、アーティストにブラッシュアップしてもらうことができます。



FBX Exporter パッケージをインストールすると、「**GameObject**」>「**Export To FBX**」の下にモデルをエクスポートするオプションが表示され、アーティストとモデルを共有できるようになります。

チームと明確な作業計画を定めることで、アーティストが 3D 環境オブジェクトの適切なサイズと形状に基づいてシームレスに作業できるようになり、スムーズで効率的なデザインプロセスが可能になります。

ProBuilder や Unity Asset Store のツールを使って、「キットバッシュ」（異なるアセットを組み合わせて独創的で新しいものを作ること）することもできます。このアイデアは、モデリングを趣味とする人たちが、鉄道模型や飛行機模型のキットをマッシュアップして独自のカスタムプロジェクトを構築するやり方に由来しています。

『Guns of Boom』（右）と Unity Asset Store のアセットから作成されたプロトタイプ（左）

Polybrush

Polybrush は 3D スカルプティングツールで、地形スカルプティングツールに似ていますが、メッシュ用です。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ためのモードがあります。ProBuilder と組み合わせることで、Polybrush は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を提供し、デザインプロセスにおいて様々な外観の環境を試すことができます。

「Polybrush」ウィンドウは、トップツールバー（「ProBuilder」メニューのすぐ下）にあります。主な作業モードは以下の通りです。

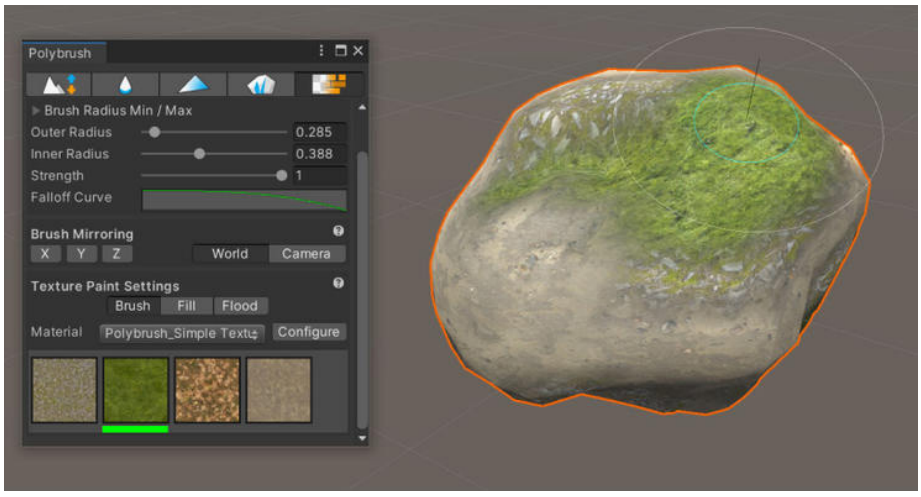
— **Sculpt**：頂点を押したり引っ張ったりしてメッシュを形成するオプション

— **Smooth**：設定内の Direction プロパティで設定された軸に沿った頂点同士の位置の差を平均化するオプション。メッシュ上のギザギザした頂点を均一化できる

— **Color**：ブラシまたはペイントバケツでメッシュ上の頂点カラーを設定する

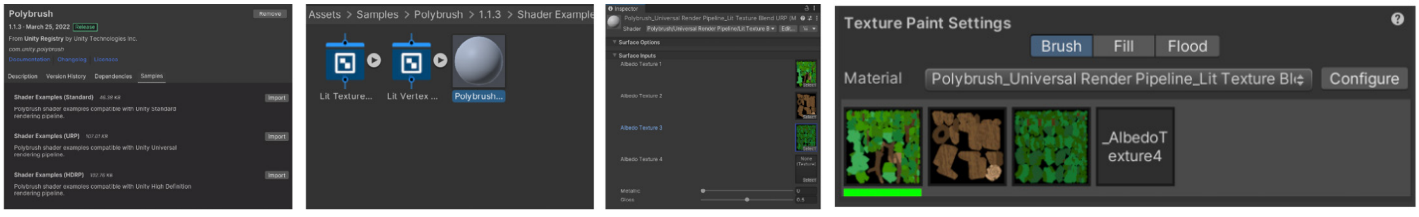
— **Texture**：メッシュ上の複数のテクスチャをペイントおよびブレンドする

— **Scatter**：メッシュの表面にプレハブを配置または散布する



「Tools」>「Polybrush」>「polybrush」からアクセス可能なウィンドウ。テクスチャをメッシュにペイントするなどのオプションがあります。

テクスチャペイントモードでは、メッシュ上でテクスチャブレンドと互換性のあるマテリアルが使用されているか確認するなど、こちらで説明されている追加の手順に従う必要があります。



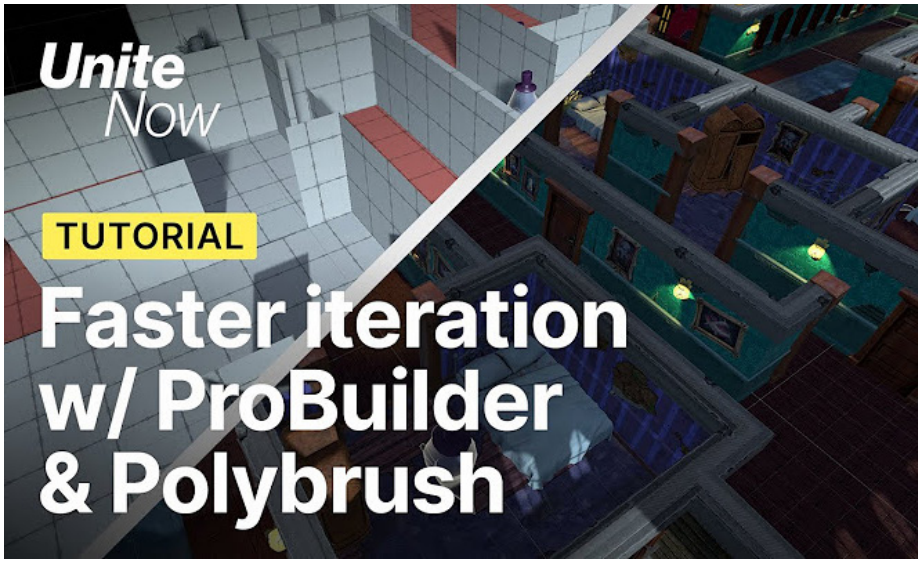
マテリアルが、テクスチャのブレンド方法を定義するシェーダーを使用していることを確認してください。

上記の画像で示されているステップを追ってみましょう。

1. 「Polybrush Package Manager」ウィンドウ内の「**Samples**」タブを開きます。レンダーパイプラインに対応するシェーダーを取得します。
2. シェーダーを右クリックして、「**Create Material**」を選択します。
3. このマテリアルが Inspector 内で使用するテクスチャを追加します。
4. Polybrush がオブジェクトからマテリアルを検知し、ペイントするテクスチャを選択できるようにします。最初に変更を保存する必要があるかもしれないことにご注意ください。

以下の動画を視聴して、ProBuilder と Polybrush を使用したデザイン作成やプロトタイピング、芸術的なシーンの仕上げ方を学びましょう。

Polybrush で惑星を作成する



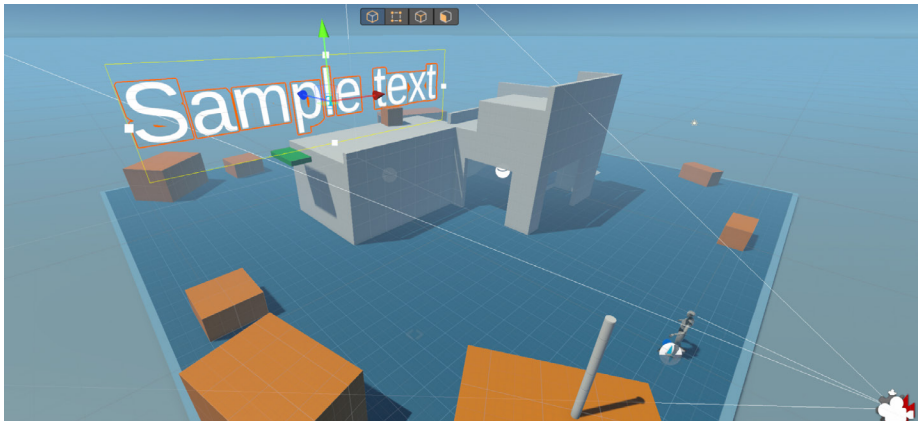
ProBuilder と Polybrush による迅速なイテレーション

ミニヒント：ビジュアライザー

前述のように、カラーコーディングや ProBuilder のマテリアルライブラリを使用することで、チームに意図を伝えることができます。ゲーム開発は複雑であり、プロトタイピングのアイデアを同僚に可能な限り明確に伝えることができれば、上手くいきそうなアイデアとそうではないアイデアについて、十分な情報に基づいた決定を下すことができます。デザインの意図を明確に伝える際に役立つヒントをさらにいくつか紹介します。

3D オブジェクトテキスト

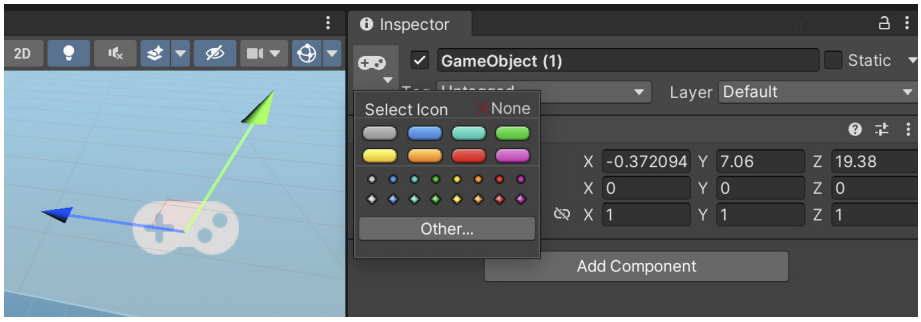
「GameObject」>「3D Object」>「Text」>「TextMeshPro」を使用して、3D 空間にフローティングサインを作成します。これらは、レベルデザインに関する追加情報を同僚（および自分自身）に提供する仮想の付箋として機能します。これらを移動、スケール、親ゲームオブジェクト下にグループ化して整理できます。



3D テキストオブジェクトは、通常のテキストラベルのような UI システムの一部ではなく、スタンドアロン要素であるため、ホワイトボックスまたはグレーボックスのレベルデザインで簡単に移動したりサイズ変更したりできます。

ゲームオブジェクト用のカスタムアイコン

カスタムアイコンを使うと、特に大規模なゲームワールドでは、ゲームオブジェクトを識別しやすくなります。カスタムアイコンの使用場面の例としては、レベルの開始時と終了時、プレイヤーや敵のスポーンポイント、隠しエリアやシークレットエリアなどが挙げられます。カスタムアイコンは、レベルボスの体力やステータスなどの情報を素早く伝えるためにも使用できます。アイコンは、Inspector 内のゲームオブジェクト名の横から変更できます。ギズモの視覚化で可能になることについては、こちらの Unity Learn チュートリアルをご参照ください。

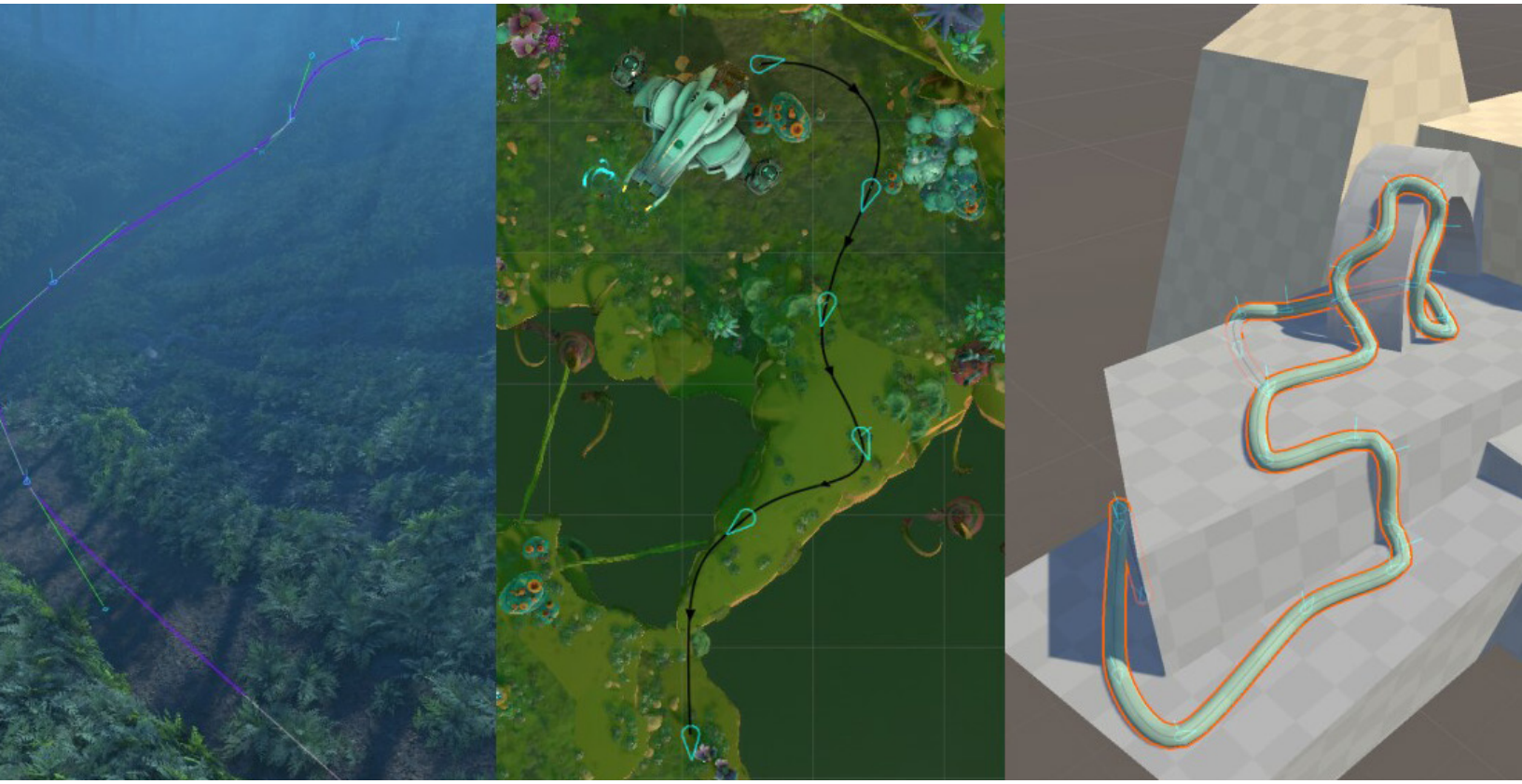


シーンビューに表示されるゲームオブジェクトアイコンをカスタマイズする様子

Unity におけるユーザーインターフェースデザインと実装

Unity の UI オーサリングツールの使用方法を学んでスキルを磨きましょう。この 130 ページ以上にも及ぶガイドでは、アーティスト、デザイナー、開発者向けに、Unity の 2 つの UI システム、Unity UI と UI Toolkit を使って洗練されたインターフェースを構築するためのヒントとベストプラクティスを紹介しています。

e ブックをダウンロード

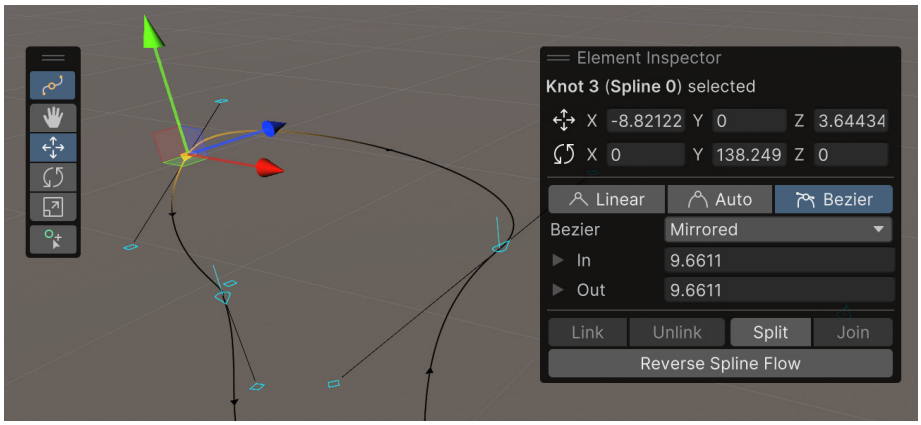


スプラインのユースケースの例：森を通る道、アニメーションパス、チューブ、ワイヤーメッシュ。「Splines Package Manager」ページで、インストール後に実現できるすべてのことを紹介した他のサンプルを確認できます。

スプライン

Unity 2022 LTS で導入された Splines パッケージを使用すると、ゲーム内で川、道路、カメラトラック、その他のパスに関連する機能のスプラインパスを作成できます。作業している環境によっては、スプラインはレベルデザインの重要なコンポーネントになり得ます。

「GameObject」>「Spline」>「Draw Splines Tool」から新しいスプラインを作成します。新しいゲームオブジェクトは、Hierarchy 内に Spline コンポーネントがアタッチされ、ツールが利用可能な状態で作成されます。



Unity におけるスプラインのハンドルとコントロールは、一般的な DCC アプリケーションのベクターまたは 3D 描画ツールに類似しています。

スプラインを作成すると、プログラマーとアーティストの両方が興味深い方法で使用できます。例えば、プログラマーはスプラインのポイントを読み取り、API を使用してゲームロジックに使用できます。

スプラインの活用に利用可能なコンポーネントを以下に示します。

— **Spline Instantiate**:スプラインに沿ってアイテムのコピーを生成。フェンス、並木、石造りの歩道などのオブジェクトを作成する際に使用できる。

— **Spline Animate**：ゲームオブジェクトをスプラインに沿って動かす。カメラやキャラクターと併せて、または Unity で動きを定義する必要がある場面で使用できる。

— **Spline Extrude**：スプラインに沿ってチューブメッシュを構築。ワイヤー、パイプ、ロープ、麺などの形状を作成し、それらの形状を編集する際に使用できる。

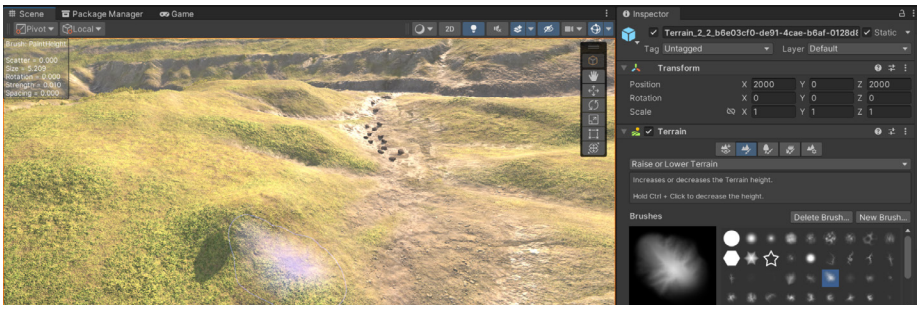
Splines に今後実装される機能については、こちらのブログ投稿をご覧ください。

地形ブラシとスタンプは、リアルなランドスケープの作成に役立ちます。前述のように、GIS データは地形デザインの素晴らしいインスピレーションとなります。

Terrain

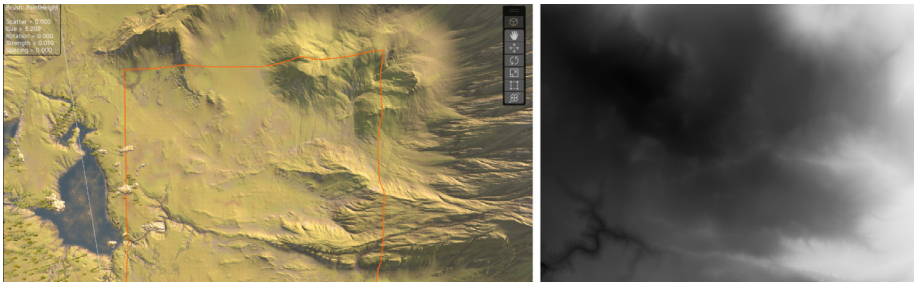
Unity Terrain Editor を使用すると、最適化されたリアルな地形を作成できます。これは有機的で平らでない表面領域を作成するのに適したツールで、前のセクションで説明した他のレベルデザインツールと組み合わせて使用できます。

地形編集ツールを使用するには、**「GameObject」>「3D Object」>「Terrain」**で地形オブジェクトを作成します。Terrain コンポーネントには、Paintbrush ツールで地形のハイトマップをペイントする際に、地形の高さを調節するためのブラシが用意されています。地形の一部を隠したり、現在のハイトマップ上にスタンプブラシを追加したり、または単に地形を仕上げるのに使用します。



ブラシを使用して地形の高低差を作成している様子。Terrain Editor は大規模な地形の修正を目的としているため、デフォルト設定では通常、大きな Unity ユニットの数値が使用されていることに注意してください。

表面が変更されると、コライダー、テクスチャ、またはアタッチされた植生や樹木が自動的に反応し、変更が地形の表面ジオメトリに反映されます。



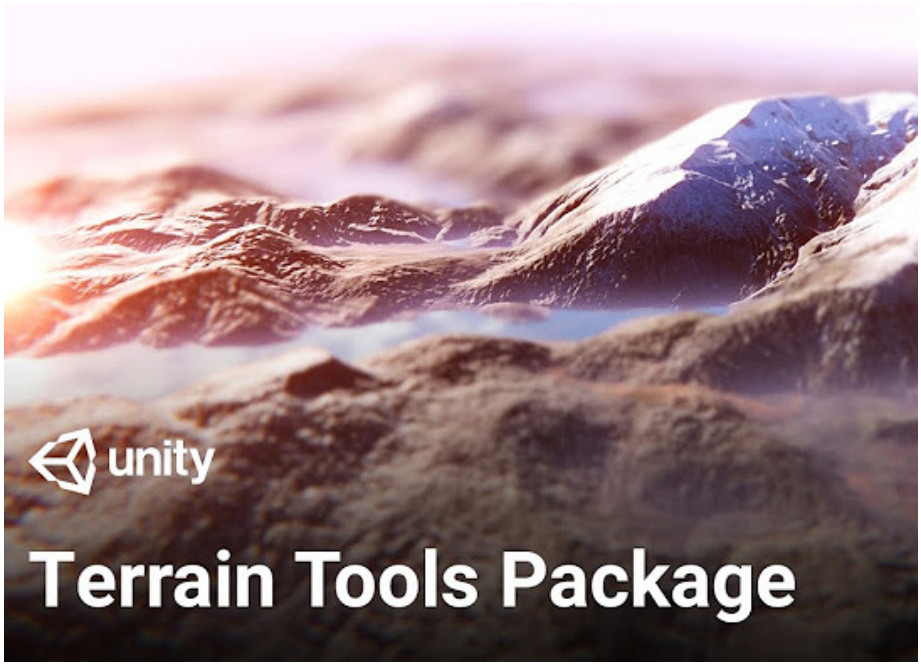
右の画像のように、Terrain Editor で生成されたハイトマップ（左の画像の赤線で囲まれた部分）のエクスポートおよび外部のハイトマップのインポートが可能。GIS や現実世界のデータを Unity にインポートする方法は、こちらのコミュニティ動画を参照してください。

Terrain Editor には、植生、樹木、地形の描画距離を定義する設定に加え、地形メッシュの解像度、ライティング、風エフェクトを制御する設定があります。

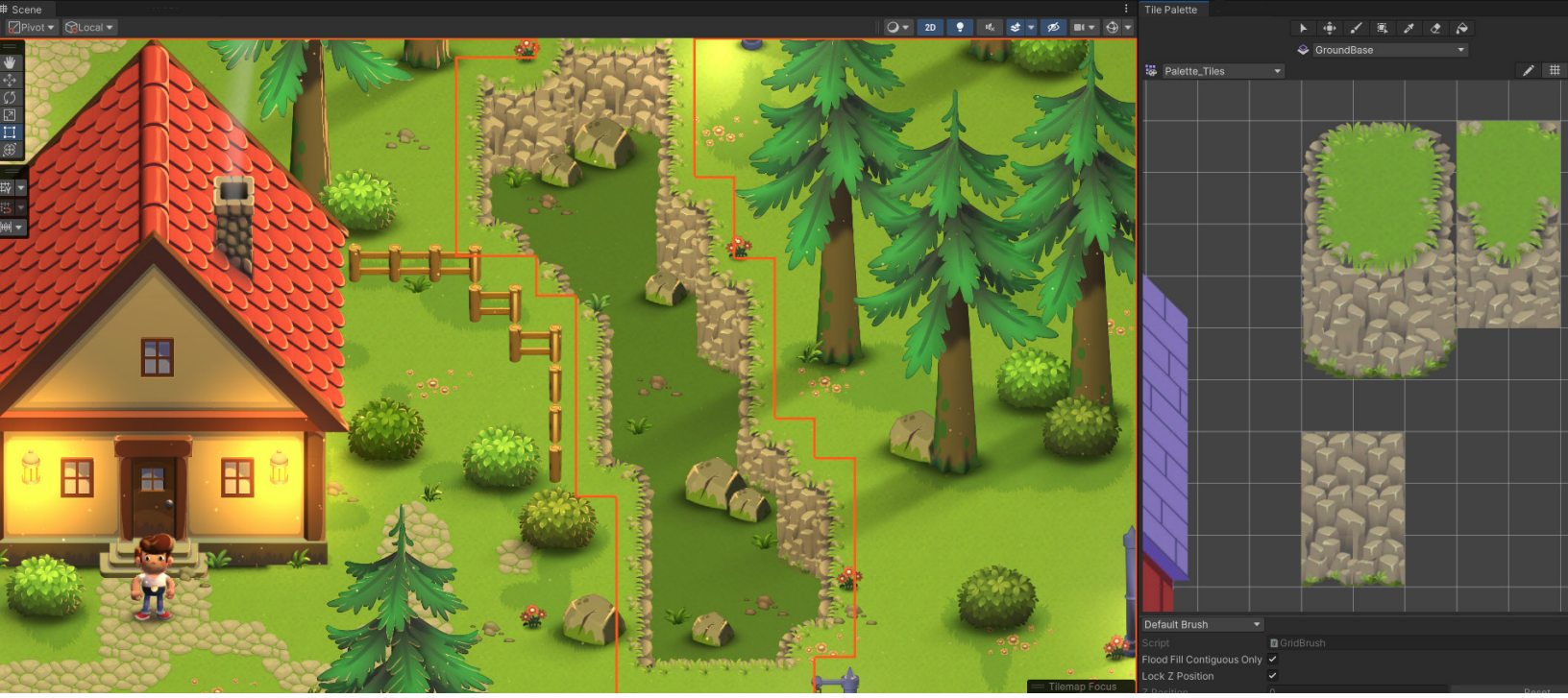
Terrain Editor と一緒に、Terrain Tools パッケージをインストールすると、さらに多くの地形スカルプティングブラシやツールがプロジェクトに追加され、鮮やかな地形アセットの作成に役立ちます。

Unity の Terrain Tools の詳細については、以下の動画をご覧ください。また Package Manager でサンプルシーンをインポートすると、完成シーンがどのように作成されたのかを見ることができます。

Terrain Tools を使用した美しいランドスケープ



Terrain Tools パッケージの動画



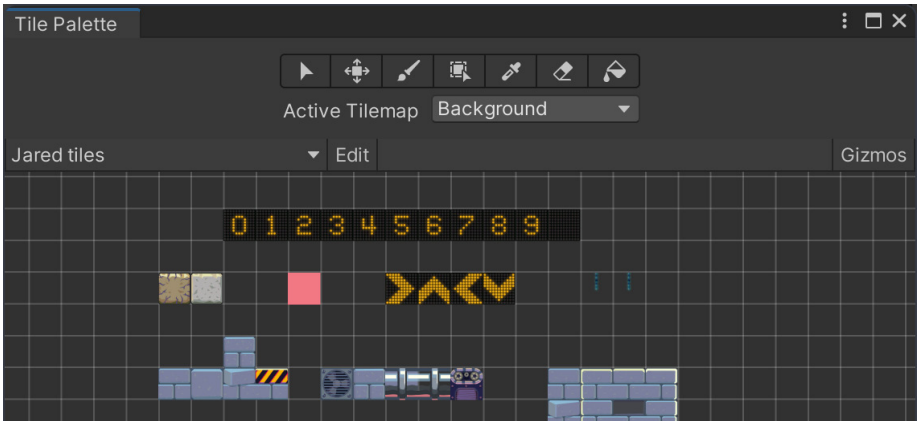
タイルマップを使ってトップダウンゲームを作る様子

2D Tilemap

2D Tilemap Editor は、2D Project Template と一緒にインストールされ、Package Manager からもインストールできます。2D Tilemap はクイックプロトタイピングに最適な機能です。タイルマップは、グリッド上に配置されたタイルと呼ばれる小さなスプライトを使ったゲームワールドの作成を可能にします。1 つの大きな画像がゲームワールドにレイアウトされるのではなく、レンガのような塊に分解され、レベル全体で繰り返されます。2D Tilemap は、長方形、等角形、六角形のタイルをサポートしています。

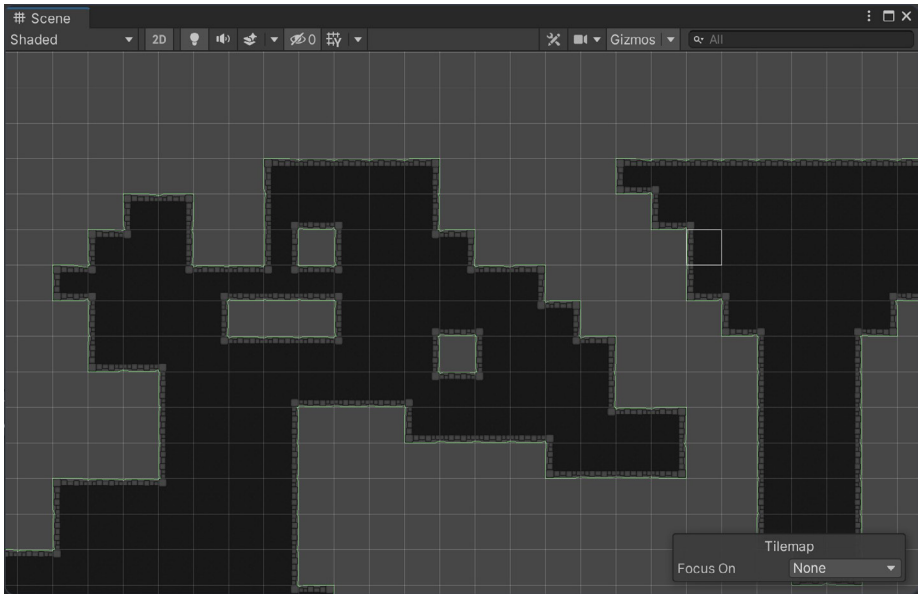
ブラシツールでタイルをグリッド上に効率よくペイントでき、スクリプトを使えばペイントルールを追加できます。また、効率的なテストや編集を可能にする、自動衝突生成機能もついています。

「Tile Palette」ウィンドウには、タイルマップのペイントや編集に役立つすべてのタイルとツールが揃っています。新規パレットを作成するには、「Create New Palette」ボタンをクリックします。オプション付きのドロップダウンウィンドウが表示されます。パレットに名前を付け、オプションを設定し、「Create」をクリックし、選択したフォルダーに保存します。



パレットがロードされた「Tile Palette」ウィンドウ

次に、パレットにタイルを追加します。スプライト（タイルアセットを作成するようプロンプトが表示されます）またはタイルアセットを「Palette」ウィンドウにドラッグします。これでパレットのグリッドにスプライトが表示されるはずです。



プレースホルダータイルを使用したグリッドベースのレベルのプロトタイプ。緑のアウトラインがコライダーの境界を示している。

タイルマップは、2D レベルのプロトタイプを効率的に作成するのに役立ちます。物理演算のためにタイルマップに Tilemap Collider 2D コンポーネントを追加します。これは、タイルアセットで設定されたコライダータイプに基づいて、各タイルにコライダーを追加します。

Composite Collider 2D を追加してコライダーを 1 つにまとめると、ジオメトリを **Outline** に設定した場合の衝突動作が滑らかになります。タイルは個々のタイルとしてではなく、1 つの連続した地形のように動作します。この設定により、パフォーマンスも若干向上します。しかし、ゲーム実行時にタイルを追加したり削除したりすることを計画している場合は、各タイルにコライダーを維持したほうがよいでしょう。

Collider 2D コンポーネントの「**Used by Composite**」オプションにチェックを入れ、Rigidbody 2D タイプを **static** に設定して、落下しないようにしてください。

様々なパレットツールを見てみましょう（キーボードショートカットはカッコ内に記載）。

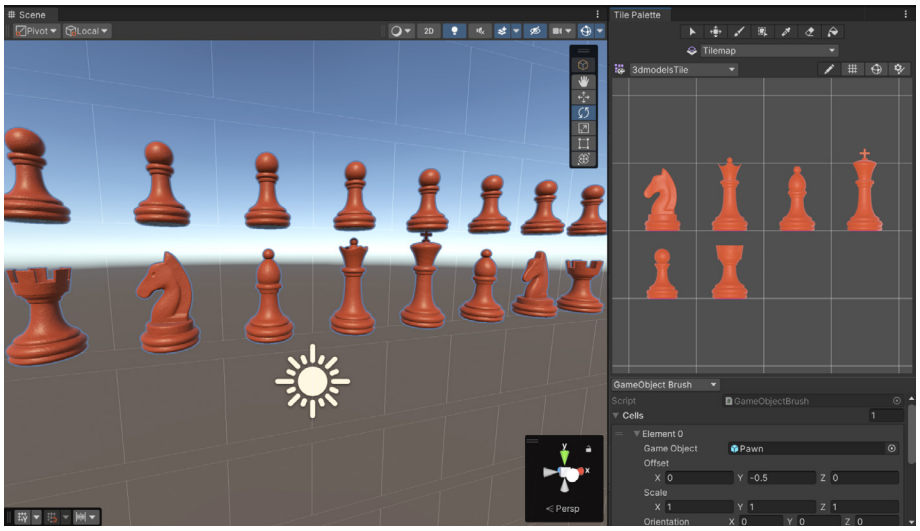
Tile Palette ツール

1. **Selection（S）：**クリックしてタイルを 1 つ選択するか、ドラッグして長方形の領域内のタイルを選択する
2. **Move（M）：**選択したタイルを移動する
3. **Brush（B）：**選択したタイルとブラシで、アクティブタイルマップ（アクティブタイルマップのドロップダウンから 1 つ選択）にペイントする
4. **Fill Selection（U）：**ドラッグして、選択したタイルで長方形の領域を埋める
5. **Tile Sampler（I）：**タイルマップからタイルを選択し、アクティブに設定してペイントする
6. **Eraser（D）：**タイルマップからタイルを削除する
7. **Fill (G)：**タイルで領域を塗り埋める（領域は他のタイルで囲まれている必要がある）

2D Tilemap Extras

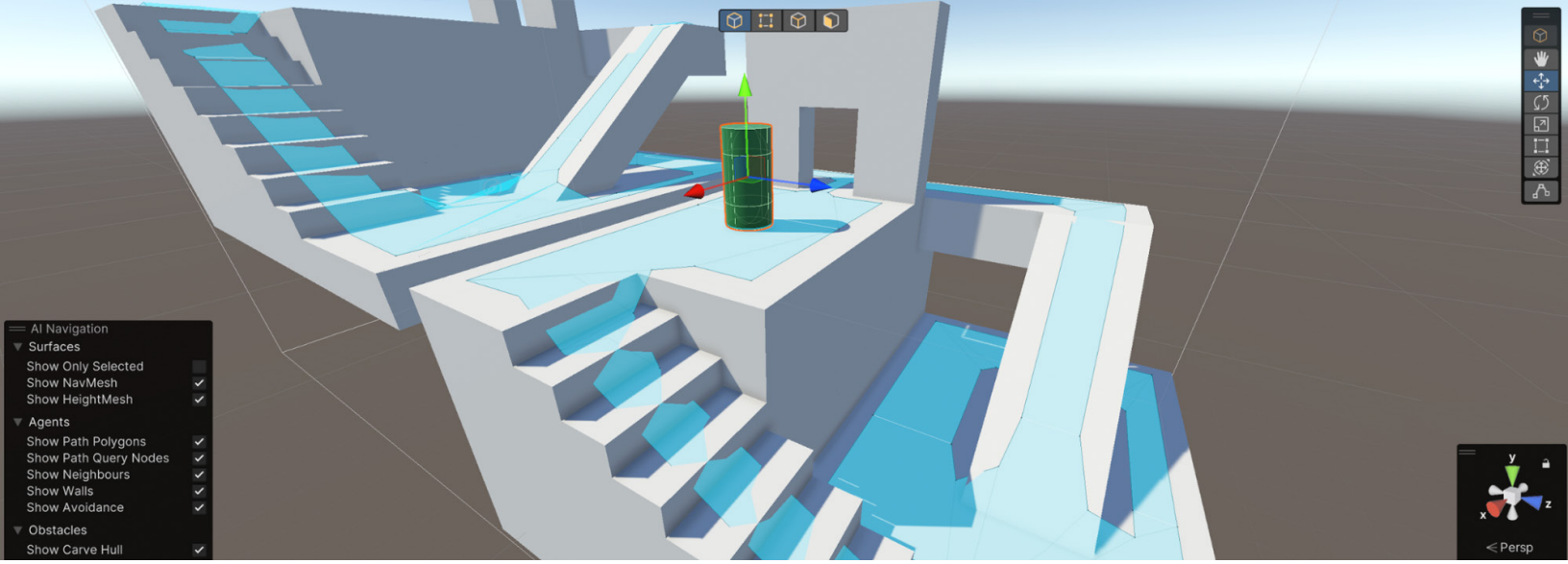
追加パッケージ、2D Tilemap Extras には、ルールタイル、アニメーションタイルなどのタイルマップの新機能へのアクセスを可能にする追加スクリプトが含まれています。

レベルデザイナーにとって特筆すべき機能の 1 つは、ゲームオブジェクトのサポートです。これにより、3D モデルのような任意のプレハブで構成されるタイルパレットを作成し、2D Tilemap グリッドを使用してシーンにゲームオブジェクトを配置することができます。ゲームがグリッドパターンに従う必要がある場合、これは強力なレベルデザインツールになります。



2D タイルマップを使用してシーン内のゲームオブジェクトを配置する様子

ゲームオブジェクトを使ってタイルパレットを作成するには、ドロップリストの「**Default Brush**」オプションを「**GameObject Brush**」に変更し、**「Cells」>「Element 0」>「GameObject」フィールド**にプレハブをドラッグします。ブラシを使用して、このプレハブをタイルパレットに追加します。完了したら、「GameObject」フィールドを別のプレハブ用に変更し、タイルパレットに追加します。タイルパレットに追加したいゲームオブジェクトの数だけ、このプロセスを繰り返します。

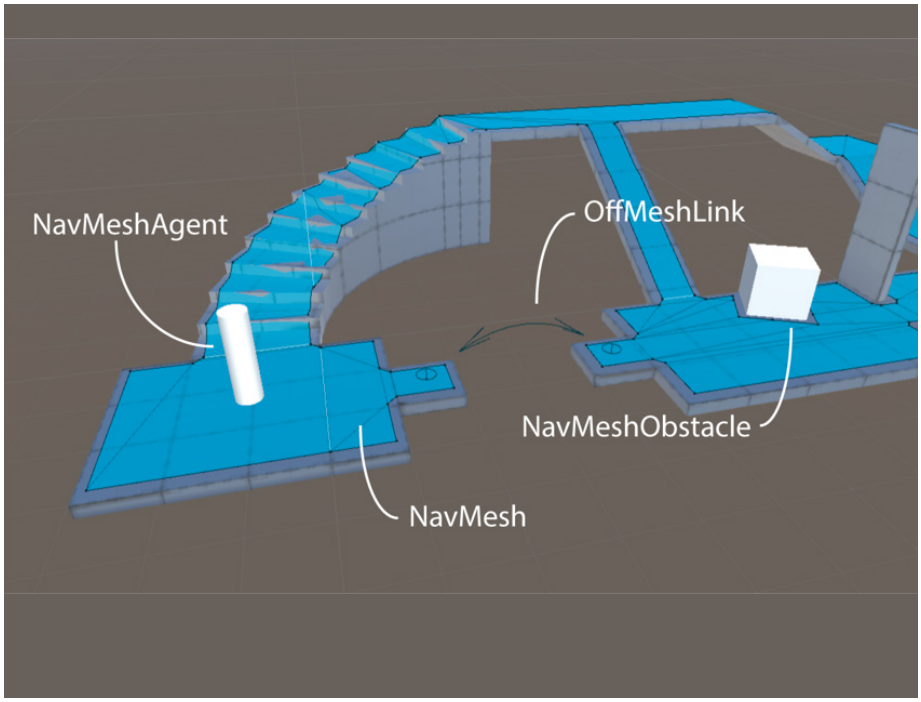


AI Navigation を使用した、キャラクターまたは NPC の簡単な経路検索

AI Navigation による経路検索

目的にたどり着くために、プレイヤーや他の NPC は、ワールドを移動する必要があるはずです。ゲームプレイメカニクスの他に、キャラクターが自らワールドを探検する様子を見ることで、プレイヤーがゲームに没入しやすくなります。

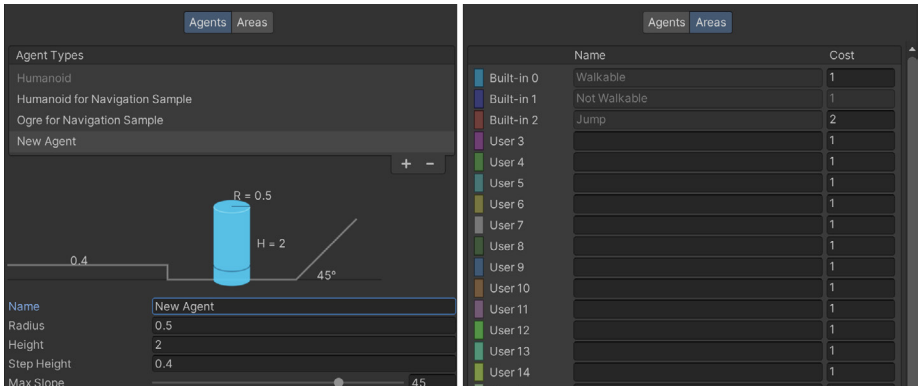
AI Navigation パッケージ（旧 NavMesh）を使用すると、シーンジオメトリから自動的に作成されるナビゲーションメッシュを使用して、ゲームワールドを自動で移動できるキャラクター（AI Navigation のコンテキストではエージェントと呼ばれる）を作成できます。



AI Navigation のキーコンポーネント：NavMesh Agent、NavMesh Surface、NavMesh Obstacle、NavMesh Link

「Navigation」ウィンドウ：エージェントとエリア

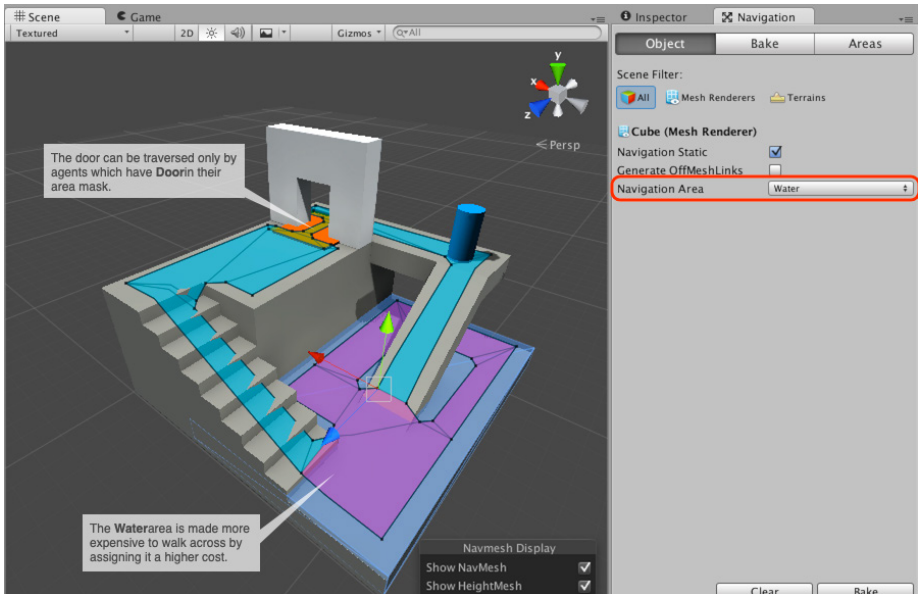
トップメニューの**「Window」>「AI」>「Navigation」**では、マップ上を動き回るエージェントやキャラクターのタイプと、エージェントの優先パスとなるエリアを設定できます。



プロジェクトのエージェントとエリアを定義する様子

エージェントタイプはキャラクターの全体的な体積を定義し、キャラクターコライダーと同様に円柱形で表現されます。**Step height**（ステップの高さ）はエージェントが跨げる垂直方向の最大距離を定義し、**Max slope**（最大傾斜）はその名の通り、エージェントが登れる傾斜の上限を設定します。また、エージェントは 1 つのメッシュ面から別のメッシュ面にジャンプでき、例えばキャラクターが低い地面に落下したり、別のエリアにジャンプしたりすることを可能にします。**Drop height**（落下の高さ）と**Jump Distance**（ジャンプの距離）はそのパラメーターを定義します。

エージェントタイプの設定は、「**Bake**」ボタンを使用して、NavMesh Surface コンポーネントで移動可能なエリアを準備する際、AI ナビゲーションアルゴリズムによって使用されます。

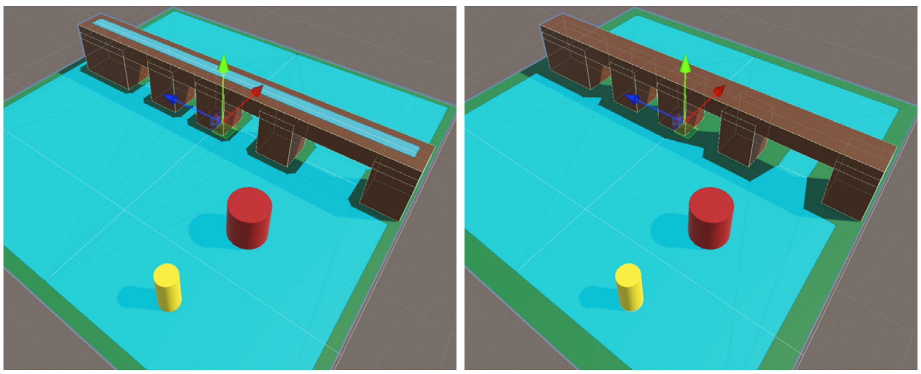


Unity Documentation に記載されている例。コストの高いエリアが紫色で示されている。

エリアは、特定のエリアを横切って歩く難易度を定義し、経路検索中は低コストのエリアが優先されます。さらに、各ナビメッシュエージェントは、エージェントが移動可能なエリアを指定するのに使用できるエリアマスクを持っています。

NavMeshModifier Volume コンポーネントを使用して、変更可能な立方体エリアを作成し、エリアタイプを定義します。特定のエリアタイプのメッシュオブジェクトを作成する NavMeshModifier コンポーネント（override area を有効にしたもの）か、2 つの場所の間に移動可能なリンクを作成する NavMeshLink を使用することで、リンクにエリアタイプに対応したコストを持たせられます。異なるエリアタイプは、異なるプリビジュアライゼーションカラーで表示され、これはこのメニューから設定可能です。

AI Navigation で移動可能なパスを作成する 3 つのステップ



左の画像では、黄色で示されたエージェントタイプ用にナビメッシュ表面が生成されています。歩行可能な道（青い部分）によって、橋の各柱の下が歩けるようになっていることに注目してください。左側では、赤色のエージェントタイプ用にナビメッシュ表面が生成されています。サイズの大きさを考慮して、歩行可能な道は、柱の間の空間が広い場合しか通れないようになっています。

1. **ナビメッシュ表面の作成：**エージェントが移動可能な表面を定義します。**「GameObject」>「AI」>「NavMesh Surface」**から簡単に行えます。Bake を選択し、デフォルトパラメーターに基づいてナビゲーションデータを生成します。NavMeshSurface コンポーネントの主な設定は以下の通りです。

   a.**Agent Type**：表面を移動するエージェントタイプごとに 1 つの NavMeshSurface を作成します。各エージェントは、設定に基づいて事前にベイクされた異なるデータを使用します。エージェントタイプに変更を加えた場合は、データを再度ベイクする必要があります。

   b.**Default Area**：生成される移動可能な表面のデフォルトのエリアタイプ。前述のように、ここで選択したデフォルトのタイプをオーバーライドするモディファイアがあります。

   c.**Generate Links**：有効にすると、ナビゲーションリンクが生成され、エージェントパラメーターの **Jump Distance** と **Drop Height** に基づいて、キャラクターがジャンプまたは落下できるようになります。

d.**Use geometry**：メッシュの情報または物理演算コライダーのデータを使用して表面を生成できます。

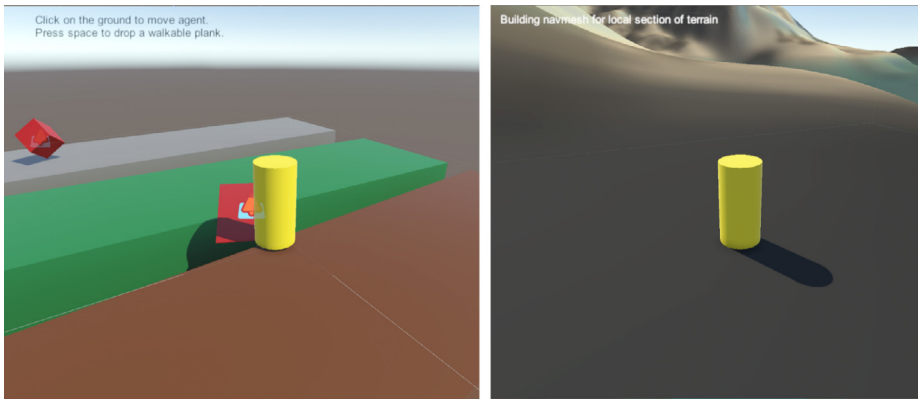
e.**Object collection**：表面生成に使用するオブジェクトやレイヤータイプをフィルターできます。デフォルトでは、シーン内のすべてのメッシュを使用します。特定のオブジェクトのメッシュを除外したい場合、**NavMesh Obstacle** または **NavMesh Modifier**（オブジェクト除外）コンポーネントを追加できます。

f.**Advanced**：ナビメッシュデータに必要な精度を微調整するオプションです。

g.**Bake**：この設定とエージェントタイプ設定に基づきデータを処理します。一方、Clear **はデータを削除します。**

2. キャラクターを表すゲームオブジェクトを作成します。上の例に従って、エージェントタイプのサイズに対応する円柱を作成し、NavMesh Agent コンポーネントを割り当てます。エージェントがナビメッシュ表面にマッチするタイプに設定されていることを確認してください。

3. ナビゲーションをテストします。AI Navigation サンプルに含まれる **ClickToMove** コンポーネントをゲームオブジェクトに追加して、クリックした場所に移動させることができます。また、AI Navigation を使用して、NPC をある地点から別の地点に移動させる **NavigationGoals** の動作も確認できます。



Package Manager から AI Navigation のサンプルをダウンロードしてインポートしましょう。

AI Navigation は、ProBuilder と Terrain Tools との互換性があり、ビジュアライゼーションツールで表面の異なる移動可能エリアやエージェントタイプに関連するその他の情報を視覚化することもできます。

TileWorldCreator 3 はアセットストアから購入可能

レベルデザインツールのクリックリファレンス

以下のリストでは、前のセクションで紹介したツールと Unity Asset Store からの提案（カテゴリ別）を掲載しています。特定のゲームジャンルに特化したソリューションについては、Unity Asset Store のこのセクションを参照してください。

Unity ツール	説明	入手場所
3D モデリング、プロトタイピング		
ProBuilder	Unity 内で直接ジオメトリをビルド、編集、テクスチャ適用する	Package Manager
Polybrush	メッシュペインティング、スカルプティング、ジオスキャッタリング	Package Manager
Splines	スプラインパスとチューブ状の 3D メッシュを作成する	Package Manager
UModeler	ローポリ 3D モデリングと環境のプロトタイピング	Unity Asset Store

SabreCGS	空間領域構成法を使用して複雑なレベルを構築する	Unity Asset Store または GitHub
ハイトマップに基づく地形生成		
Terrain + Terrain Tools	地形タイルで大規模なランドスケープを作成する。追加パッケージをインストールするとさらに多くのブラシとツールを入手できる	Package Manager
Gaia Pro	地形と環境生成ツール	Unity Asset Store
Atlas Terrain Editor	破壊できないプロシージャル地形を生成	Unity Asset Store
グリッド 、タイルベースのデザイン		
2D Tilemap + Tilemap Extras	グリッド上に配置されたタイルアセットを使用して 2D レベルを作成。六角形や等角タイルもサポートし、ルールタイルやアニメーションタイルなどの特殊なタイルタイプも含まれる	Package Manager
IntelliMap Pro	AI 生成された 2D のタイルマップ	Unity Asset Store
TileWorld Creator	3D タイルマップを連続的にまたは手動で作成する	Unity Asset Store
Tessera Pro	タイルプレハブを基に 3D レベルを生成する	Unity Asset Store
プレハブの整理、配置、スナップ		
Octave3D	プレハブスナップとペイントツールのコレクション	Unity Asset Store
SmartBuilder	アセット整理と生産性	Unity Asset Store
Placer 2	エディター内またはランタイムでオブジェクトをスポーン	Unity Asset Store
Grabbit	エディター内にて物理ベースでオブジェクトを配置	Unity Asset Store

追加のレベルデザインリソース

レベルデザインのトピックに特化した YouTube チャンネルは数多くあります。いくつかのおすすめチャンネルとして、**Game Maker’ s Toolkit**、**Steve Lee (Level and Game Design)** そして **Level Design Lobby** が挙げられます。

レベルデザイナーに有益な Unity ユーザーフォーラムには、**World Building Prefabs Unity** や **Cinemachine Unity** などがあります。

最後に、**Unity best practices hub** では、アーティスト、デザイナー、プログラマーのための Unity 上級者向け eBook の他、多くの貴重なリソースが入手できます。

Unity クリエイター向けのプロフェッショナルトレーニング

Unity プロフェッショナルトレーニングでは、Unity での生産性を高め、共同作業を効率的に行うためのスキルと知識を習得できます。あらゆる業界およびスキルレベルのプロフェッショナル向けに設計された豊富なトレーニングカタログを、複数の配信形式で利用できます。

すべての教材は、経験豊富なデザイナー講師と当社のエンジニアおよびプロダクトチームの協力のもと作成されています。つまり、常に最新の Unity 技術に関する最新のトレーニングを受講できます。

Unity プロフェッショナルトレーニングがどのように皆さんのチームの役に立つかについては、**こちら**をご覧ください。

unity.com