2-1 macOSにUnityをインストールしよう

ここでは、macOSにUnity HubおよびUnity本体をインストールする手順を説明していきます。

2-1-1 Unity Hubのインストール

本書ではUnity Hubを利用します。Unity HubはUnity本体ではなく、Unityのバージョンやプロジェクトを管理するためのツールです。

複数のゲームを開発している場合、ゲームごとに利用するUnityのバージョンが異なることがよくあります。またすでにゲームをリリースしていると、あとからUnityバージョンを変更するのが難しくなります。

Unity Hubを使用すると、複数のUnityバージョンの管理が非常に楽になりますので、とても重宝するツールです。

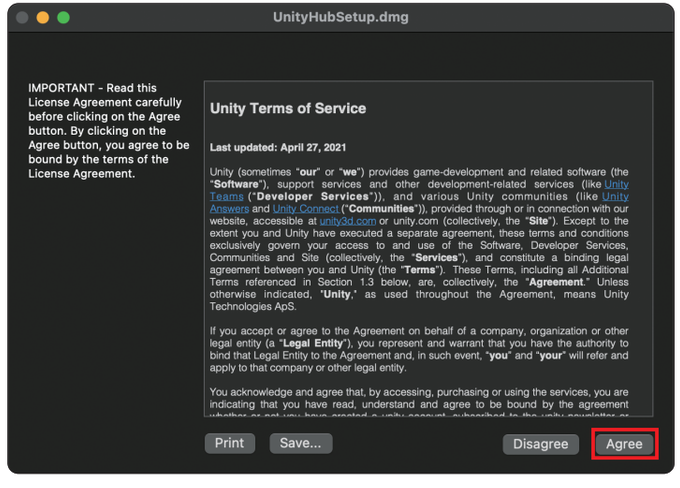
それでは、Unity Hubをダウンロード・インストールしていきましょう。

まずUnityダウンロードページ（https://unity3d.com/jp/get-unity/download）にアクセスし、「Unity Hubをダウンロード」をクリックします。Webサイトのダウンロード許可を聞かれた場合は、「許可」を選択してください。

図2.1 Unity Hubのダウンロード

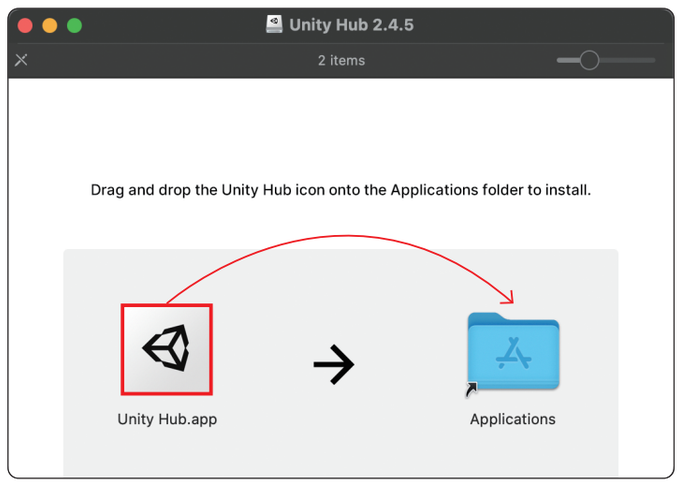
ダウンロードが完了したら、「UnityHubSetup.dmg」をダブルクリックします。英語で記述された利用規約が表示されますので、「Agree」をクリックします。

図2.2 Unity Hub利用規約



左側のUnity Hubアイコンを右側の「Applications」にドラッグします。Applicationsをダブルクリックし、中に「Unity Hub」があればインストールは完了です。

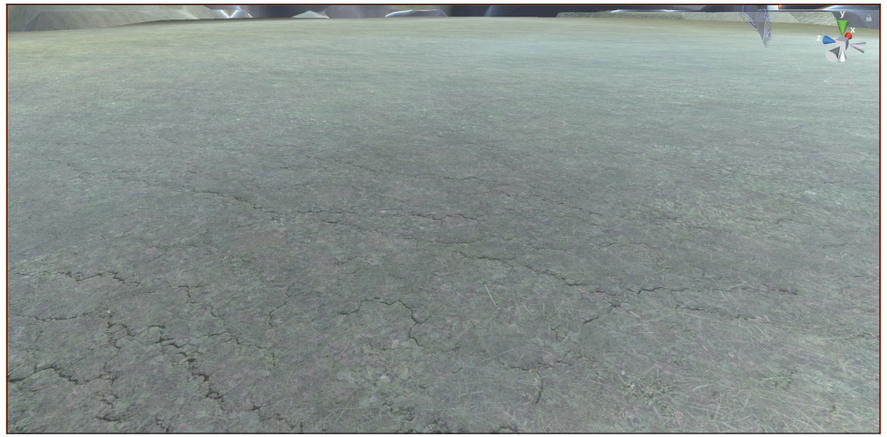
図2.3 Unity Hubのインストール



先ほど追加したテクスチャ（「GrassUV02」「GroundCracked01」「GroundDryLeaves01」「GroundStones01」）のTerrain Layerにも同じ手順でNormal Mapを設定します。これでペイントの準備が整いました。

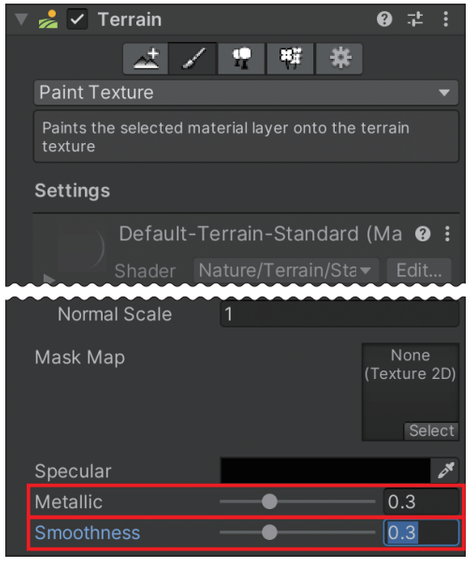
登録したテクスチャから好きなものを選んで、Terrainに色付けしてみましょう。Normal Mapを設定しましたので、ズームしてみると少し凹凸のあるリアルな見た目になっていることが確認できます。

図5.27 ペイント後のTerrain



また、Terrain LayerのMetaricやSmoothnessの値を変更すると、光の反射が変わって、遠くから見たときの様子がかなり変化します。興味のある方はこちらも試してみるとよいでしょう。

図5.28 Terrain LayerのMetaricとSmoothness



5-3 木や草を配置しよう

5-2では舞台となる地面を作成してきました。次にこの地面の上に木や草を配置して、よりゲームの舞台っぽくしていきましょう。

5-3-1 木を植える

Terrainでは木を植えることもできます。

Terrainコンポーネントの左から3番目の（Paint Trees）を選択します。「Edit Trees...」ボタンをクリックして「Add Tree」を選択します。Add Treeダイアログが開きますので、ここに木のPrefab（設計図）を指定します。

図5.29 Paint Trees

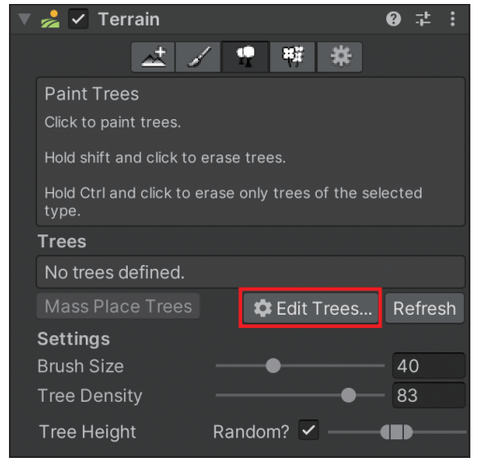
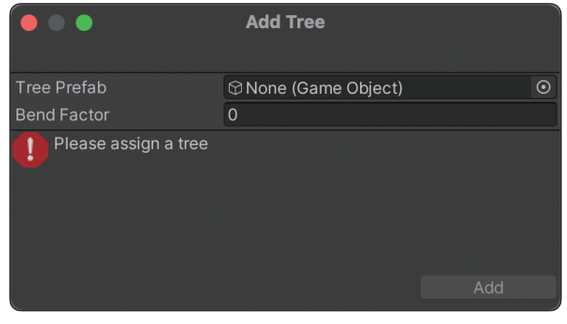


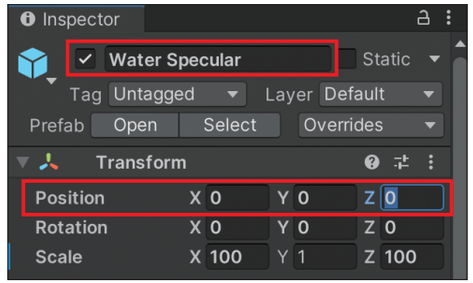
図5.30 Add Treeダイアログ



水面を配置したら、位置を調整します。Hierarchyウインドウから「Water Specular」を選択し、Transformコンポーネントの Positionで Xを「0」、Yを「0」、Zを「0」に変更します。

水面は初期状態でTerrainと同じ大きさになっていますが、もし「大海原に浮かぶ島」のような世界にしたい場合は、ScaleのXとYを（初期値の「100」よりも）大きな値に変更してください。

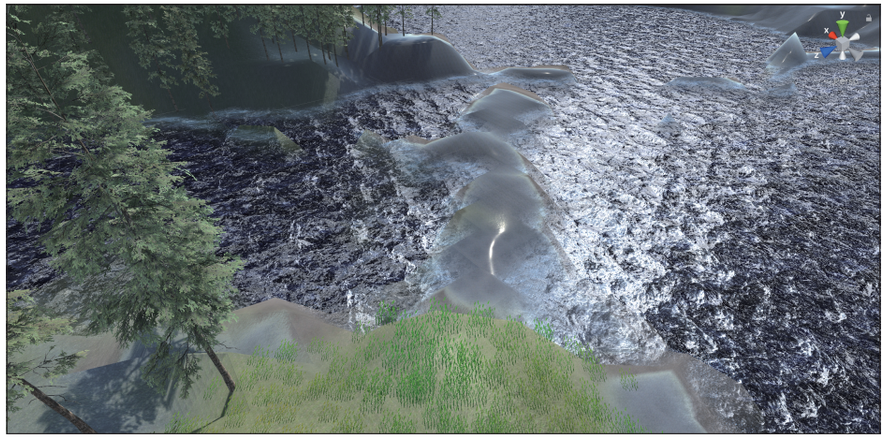
図5.44 水面のサイズ調整



これで水面の配置は完了です。カメラを水面に寄せてみると、とても美しく描画されている様子が確認できます。

ちなみに、山の上に存在する湖など、場所によって水位を変えたい場合は、その都度水面を配置してください。

図5.45 水面のズーム



5-4-2 風を吹かせる

風が吹いて木や草が揺らいでいるとよりリアリティが増します。

今回使用する針葉樹のAssetには、独自の処理が組み込まれており、何も設定しなくても風で揺らぐ効果を備えていますが、一般的にはTerrainの木はWind Zoneを使って任意の強さの風で揺らしてあげます。

Hierarchyウインドウで右クリックし、「3D Object」→「Wind Zone」を選択すると、SceneビューにWind Zoneが追加されます。

InspectorウインドウのWind Zoneで設定可能なプロパティは表5.3の通りです。

図5.46 Wind Zoneの追加

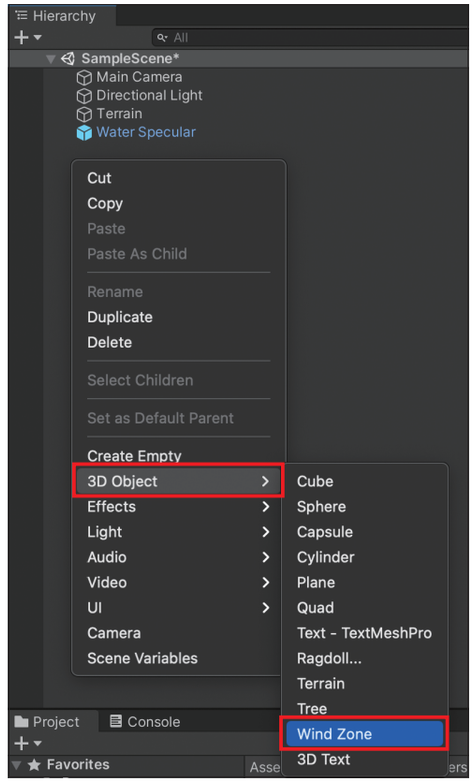


表5.3 Wind Zoneの主なプロパティ

プロパティ	説明
Mode	シーン全体に影響を及ぼすDirectionalと、範囲内だけに影響を及ぼすSphericalの2種類がある
Main	風の強さを設定する
Turbulence	乱気流を設定する。数値を大きくするとよりランダム性を与えることができる
Pulse Magnitude	Wind Zoneの風は周期的に強さが変化するが、その変化の強さを設定する
Pulse Frequency	風が変化する頻度を設定する

また、草の揺らぎについては、Terrainコンポーネントの「Terrain Settings」（一番右のアイコン）にある「Wind Settings for Grass (On Terrain Data)」で設定することが可能です。

図5.47 Wind Settings for Grass (On Terrain Data)

```
        Y軸に対して、1秒間に-12度回転させる
        transform.Rotate(new Vector3(0, -12) * Time.deltaTime);
    }
}
```

スクリプトで継承するMonoBehaviourは、ゲームオブジェクトの座標・回転・スケールを制御できるtransformフィールドを持っています。tranform.Rotate()は、Vector3を指定し、X・Y・Z各軸に対してゲームオブジェクトを回転させるメソッドですので、これを使って物体を回転させます。

ただ、Update()メソッドは毎フレーム呼ばれるメソッドで、通常は1秒間に数十回ほど実行されるため注意が必要です。今回の場合、tranform.Rotate()に対して単純にnewVector3(0,12)を渡すだけだと、1フレームあたり12度という超高速回転になってしまいます。

そのため、前のフレームからの経過時間（秒）を取得するTime.deltaTimeを掛けることで、1秒間に12度回転するようにしています。Time.deltaTimeはオブジェクトを移動・回転する際によく使用しますので、覚えておきましょう。

完成したスクリプトをDirectional Lightにアタッチしましょう。Hierarchyウインドウで「Directional Light」を選択し、Inspectorウインドウの「Add Component」ボタンをクリックして「RoundLight」で検索します。

RoundLightスクリプトが表示されますので、クリックしてアタッチします（スクリプトをInspectorウインドウにドラッグ＆ドロップしてもOKです）。

図5.65 スクリプトのアタッチ

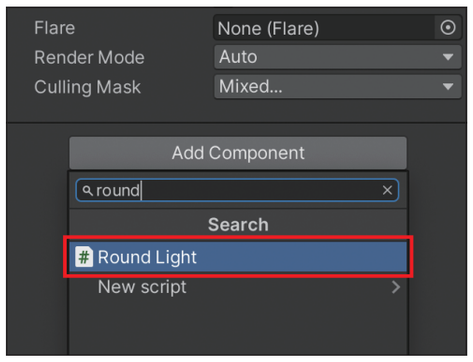
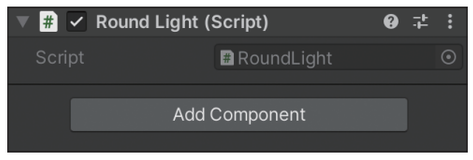


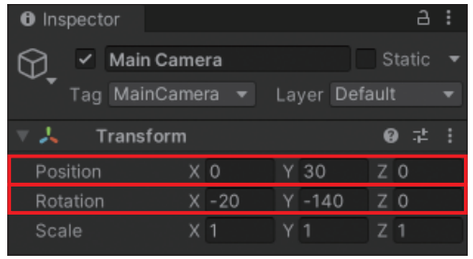
図5.66 スクリプトがアタッチされた



次に日の出の方向がよく見えるように、カメラの位置と向きを調整しておきましょう。

Hierarchyウインドウで Main Cameraを選択して、Transformコンポーネントの Positionで Xを「0」、Yを「30」、Zを「0」に変更し、Rotationで Xを「-20」、Yを「-140」、Zを「0」に変更します。

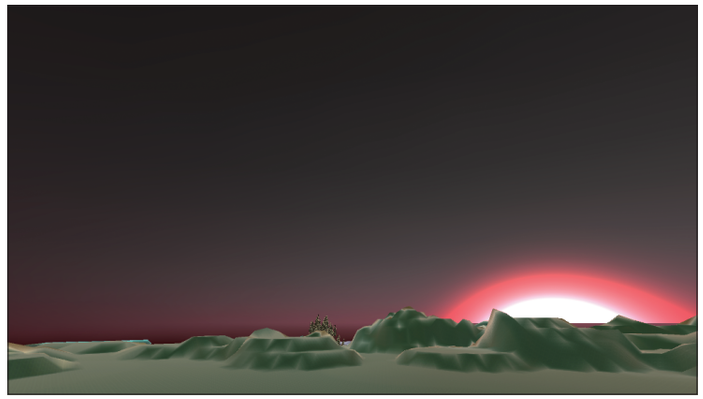
図5.67 カメラの調整

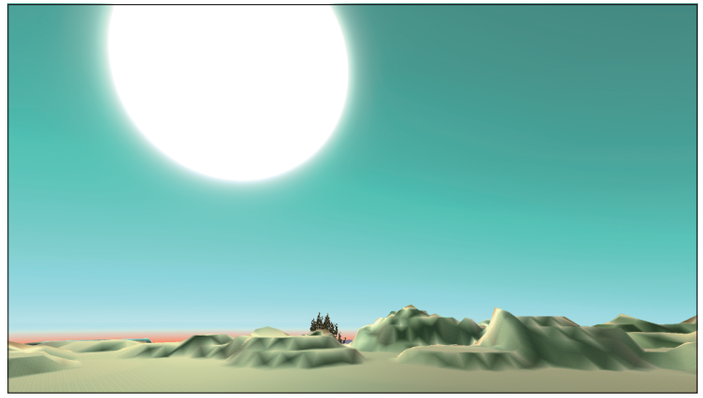


準備ができたらゲームを実行してみましょう。

昼夜以外に朝焼けの太陽も表現され、なかなかリアルですね。

図5.68 動く太陽





6-3 カメラがキャラクターを追いかけるようにしよう

次はキャラクターの動きに合わせてカメラが動くようにします。カメラを動かすスクリプトを自分で書く方法もありますが、Cinemachineを利用すれば、カメラをかんたんに制御できます。

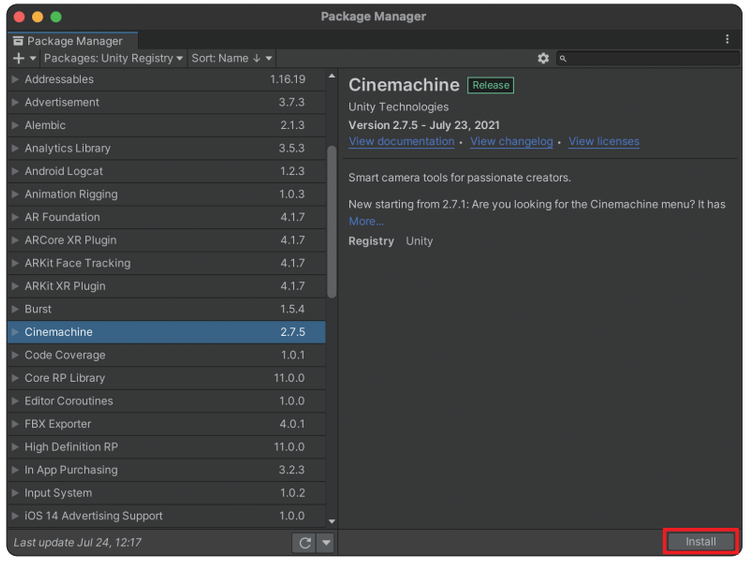
6-3-1 Cinemachineのインポート

Unity公式パッケージのCinemachineを使うと、カメラでキャラクターを追いかけたり、複数のカメラを自動で切り替えるなど、カメラに対してさまざまな制御が行うことが可能です。

Cinemachineをインポートするには、「Window」→「Package Manager」を選択してPackage Managerを開き、ウインドウ左側にあるプルダウンで「Unity Registory」を選択します。

Unity公式パッケージの一覧が表示されますので、「Cinemachine」を選択して「Install」ボタンをクリックします。

図6.13 Cinemachineのインストール



6-3-2 キャラクターを追尾するカメラを配置する

Cinemachineをインポートすると、GameObjectの作成メニューにCinemachineの項目が追加され、CinemaChineで制御する各種カメラを作成可能になります。

Hierarchyウインドウで右クリックし、「Cinemachine」→「Virtual Camera」を選択してカメラを作成しましょう。

Hierarchyウインドウに「CM vcam1」というカメラが生成されていますので、これを選択します。

図6.14 Virtual Cameraの作成

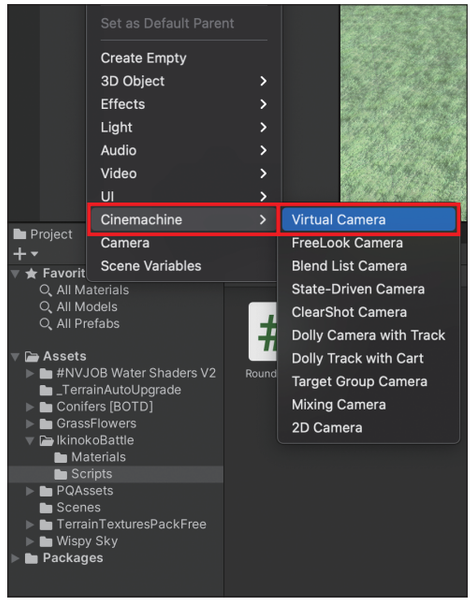
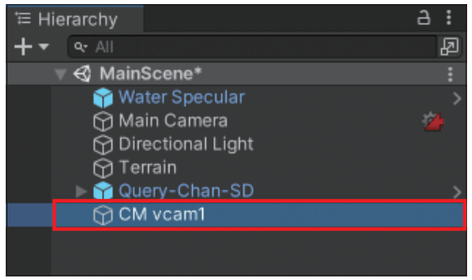


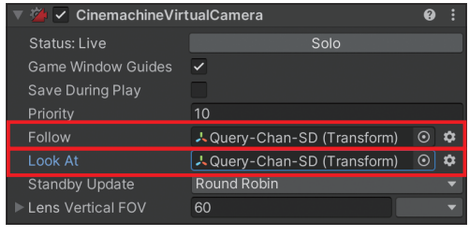
図6.15 生成されたVirtual Camera



Inspectorウインドウからキャラクターを追尾するための設定を行います。

CinemachineVirtualCameraコンポーネントのFollowに「追跡対象のゲームオブジェクト」を、Look Atに「注目対象のゲームオブジェクト」を指定します。今回はどちらも「Query-Chan-SD」としたいので、Hierarchyウインドウから「Query-Chan-SD」をドラッグ＆ドロップしてください。

図6.16 CinemachineVirtualCameraコンポーネントの設定（その1）



敵モデルの配置

敵モデルとして、7-1-1でインポートしたLevel 1 Monter Packに入っている緑色のスライムを使ってみます。

Projectウインドウで「Level 1 Monster Pack」－「Prefabs」－「Slime」フォルダを開いて、「Slime_Green」をSceneビューにドラッグ＆ドロップします。場所はクエリちゃんの近くに配置しておきます。

デフォルトではSlime_Greenのサイズが小さすぎるため、HierarchyウインドウでSlime_Greenの下にある「RIG」を選択し、InspectorウインドウのTransformコンポーネントのScaleで、Xを「30」、Yを「30」、Zを「30」に設定を変更します。

HierarchyウインドウではSlime_Greenではなく、子オブジェクトであるRIGのScaleを変更する点に注意してください。

もし親オブジェクトであるSlime_GreenのScaleを変更してしまうと、以降の手順で作成するすべての子オブジェクトの位置と大きさにズレが生じます。

図7.6 敵キャラクターのPrefab

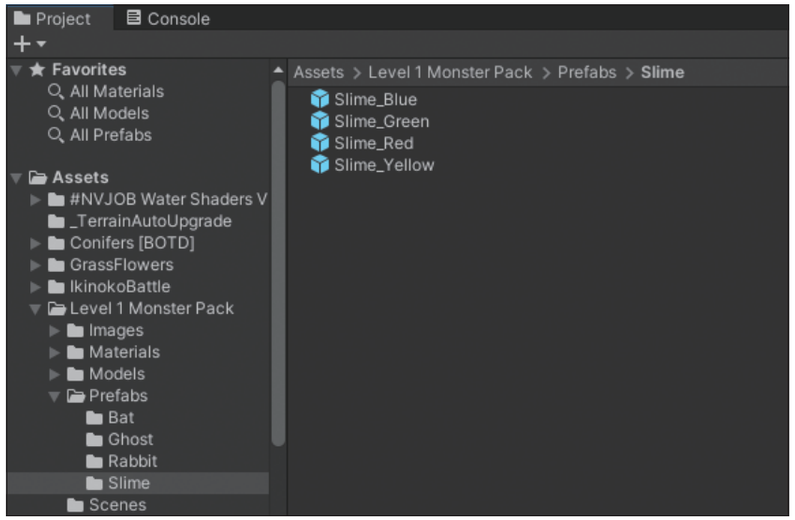
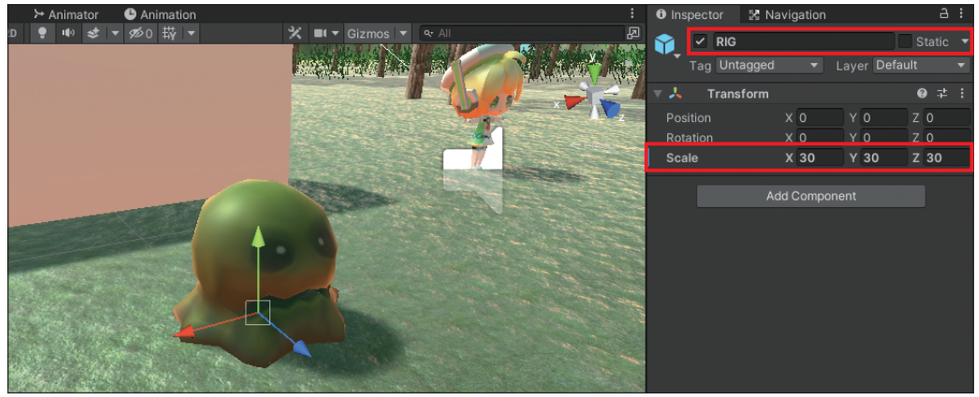
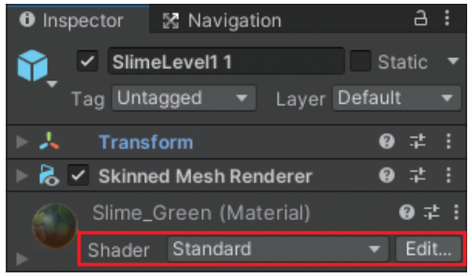


図7.7 敵キャラクターの配置とサイズ調整



また、モバイル用のShaderでは、影が表示されなくなっていますので、クエリちゃん(6-1-4参照)と同様に、Hierarchyウインドウで「Slime_Green」－「MESH」－「SlimeLevel1」を選択し、InspectorウインドウのSlime_Green(Material)でShaderを「Standard」に変更しておきます。

図7.8 Shaderの変更



NavMeshAgentのアタッチ

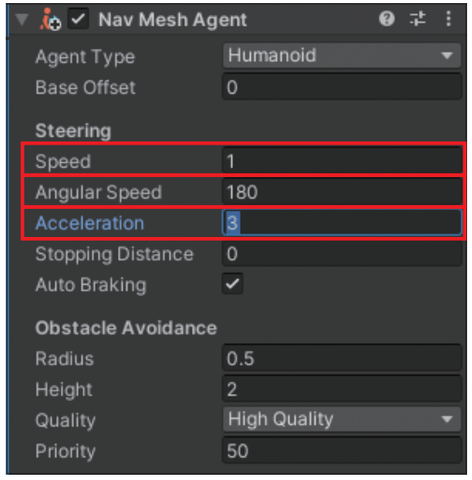
NavMesh上でゲームオブジェクトを動かすためには、ゲームオブジェクトにNav Mesh Agentをアタッチする必要があります。

Hierarchyウインドウで「Slime_Green」を選択し、Inspectorウインドウで「Add Component」ボタンをクリックして検索ボックスで「NavMesh」と入力し、Nav Mesh Agentコンポーネントを追加します。

Nav Mesh Agentコンポーネントの Steeringでは、対象キャラクターの移動速度などを設定できます。

スライムはクエリちゃんよりも移動を遅くしたいので、Speed(移動速度)を「1」、

図7.9 Nav Mesh Agentコンポーネント



7-4 敵キャラクターに攻撃させてみよう

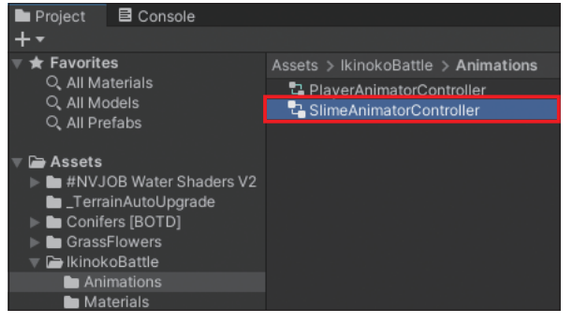
次は敵キャラクターが攻撃してくるようにしてみましょう。

7-4-1 アニメーションの設定

今回使用しているスライムの3Dモデルには、攻撃や被ダメージ時ののけぞりなど、各種アニメーションが同梱されています。Animator Controllerを作成して、スクリプトからアニメーションを制御できるようにしましょう。

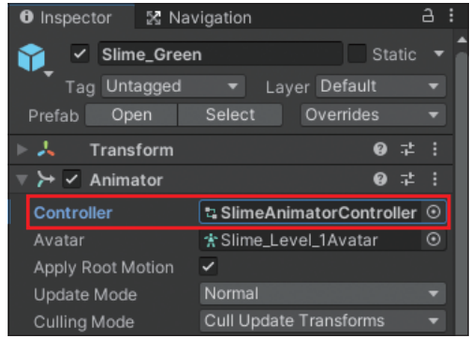
Projectウインドウの「IkinokoBattle」ー「Animations」フォルダを右クリックし、「Create」→「Animator Controller」を選択して、Animator Controllerを作成します。名前は「SlimeAnimatorController」とします。

図7.21 AnimatorControllerの作成



Hierarchyウインドウで「Slime_Green」を選択し、Inspectorウインドウの Animatorコンポーネントにある Controllerに、作成した「SlimeAnimatorController」をドロップします。

図7.22 Animationコンポーネントの設定



クエリちゃんのときと同様の手順（6-6-3参照）で、SlimeAnimatorControllerに「Idle」と「Move」のEntityを作成し、Transitionでつなぎます。「Idle」と「Move」間のConditionsも、クエリちゃんと同じくMoveSpeedを使って設定しておきましょう。

IdleのInspectorウインドウのMotionを「slime_idle」に変更し、MoveのMotionを「slime_move」に変更します。Transitionでつなぐ部分については、6-6-3と同様に設定してください。

図7.23 Idle Entityの設定

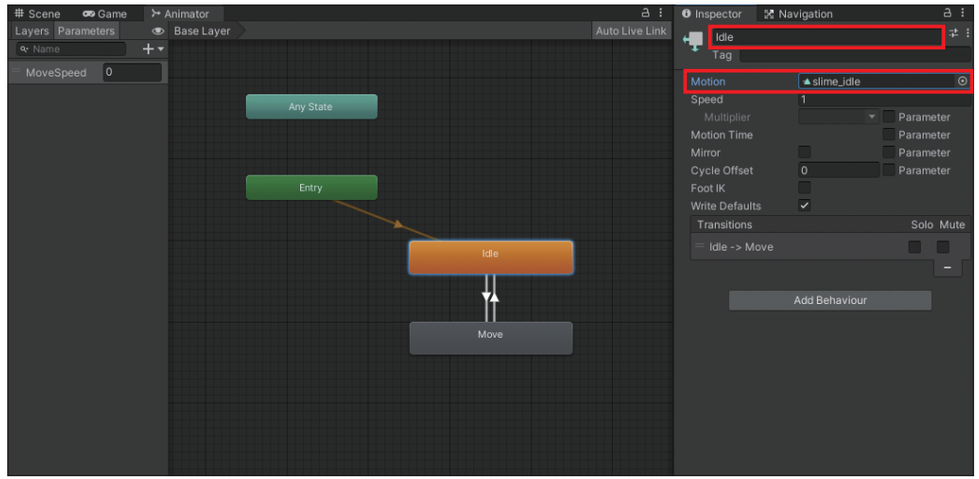
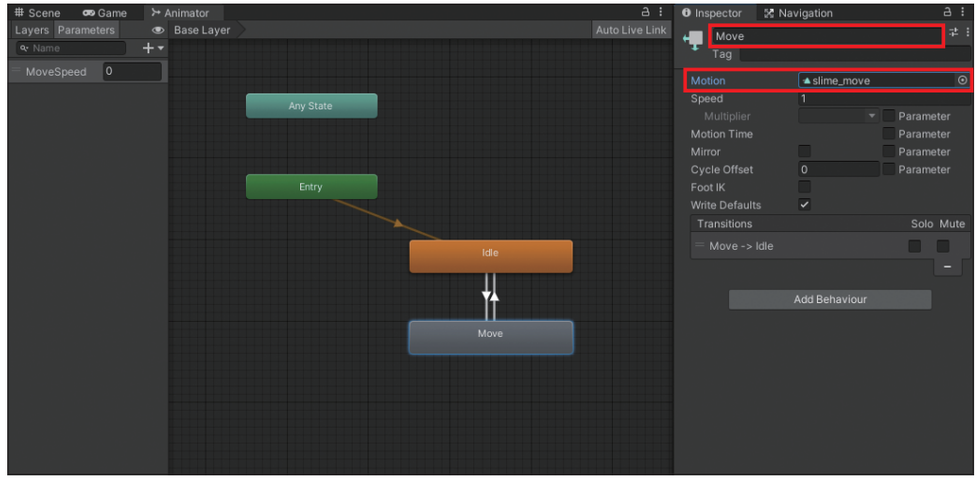


図7.24 Move Entityの設定



PausePanelが最前面に表示された状態ではItemsDialogが見えづらくなるため、Sceneビューでは、PausePanelが表示されないようにします。

Hierarchyウインドウで PausePanelの左側の空きスペースをクリックし、目アイコンを斜線が入ったマークにすると、そのゲームオブジェクトはSceneビューでは表示されなくなります。

図8.55 ItemsDialogの設定

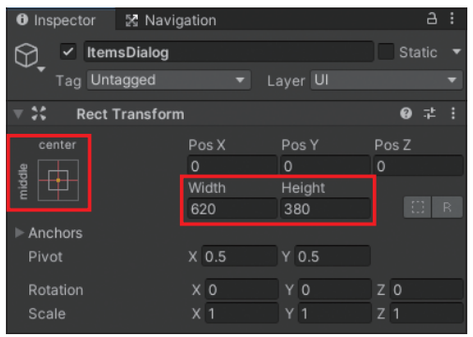
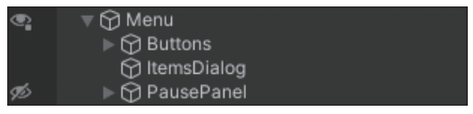


図8.56 ゲームオブジェクトをSceneビューで表示させないようにする



Hierarchyウインドウで「ItemsDialog」を選択し、Imageコンポーネントの Colorを選択してR・G・B・Aの値をすべて「255」(不透明の白)に変更します。

Inspectorウインドウで下にある「Add Component」ボタンをクリックし、Grid Layout Groupコンポーネントをアタッチします。

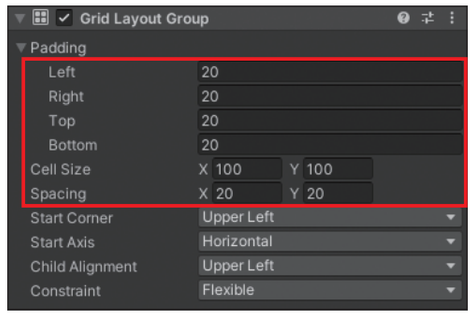
Grid Layout Groupコンポーネントは、グリッドレイアウトを実現するもので、アイコンやボタンが規則正しく並ぶようなUIを作成する際に向いています。

Grid Layout Groupコンポーネントの Paddingを展開し、表8.5のように設定を変更します。

表8.5 Paddingの設定

項目		設定値
Left		20
Right		20
Top		20
Bottom		20
Cell Size	X	100
	Y	100
Spacing	X	20
	Y	20

図8.57 Grid Layout Groupコンポーネント



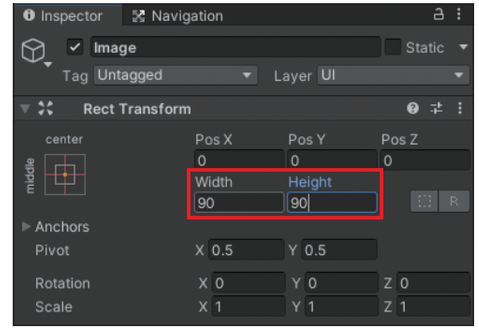
次にアイテムのアイコンと個数が表示されるアイテムボタンを準備します。

HierarchyウインドウのItemsDialogを選択して右クリックし、「UI」→「Button」を選択してButtonを作成し、名前を「ItemButton」とします。ボタンのサイズはGrid Layout Groupで自動調整を行うため、そのままでかまいません。

作成した「ItemButton」を選択して右クリックし、「UI」→「Image」を選択してImageを作成します。名前は「Image」のままでかまいません。

作成した「Image」を選択し、Inspectorウインドウの Rect Tranformコンポーネントで Widthを「90」、Heightを「90」に変更します。

図8.58 アイコンを表示する部分の設定



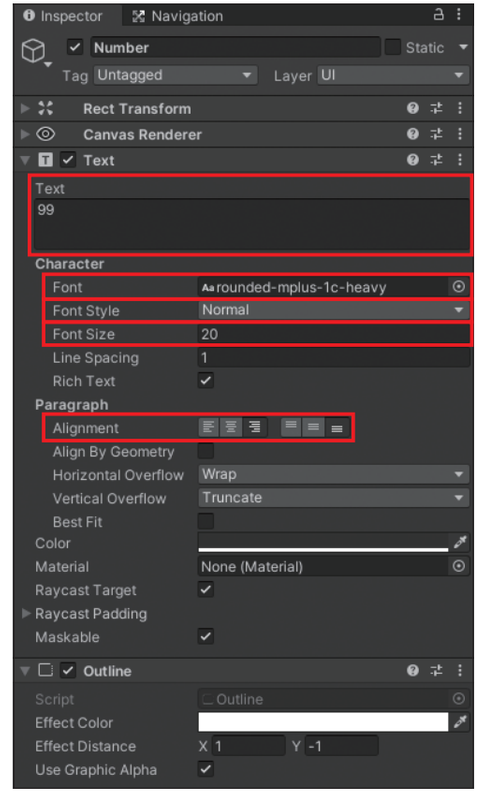
HierarchyウインドウのItemButtonを選択して、「UI」→「Text」を選択してTextを作成し、名前を「Number」に変更します。また並び順をImageの下に移動しておきます。

作成したNumberを選択し、Inspectorウインドウの Textコンポーネントで表8.6のように設定を変更します。

表8.6 Textコンポーネントの設定

項目1	項目2	設定値
Text	Text	99
Character	Font	rounded-mplus-1c-heavy
	Font Size	20
Paragraph	Alignment	(右寄せ)、(下付け)

図8.59 個数を表示する部分の設定

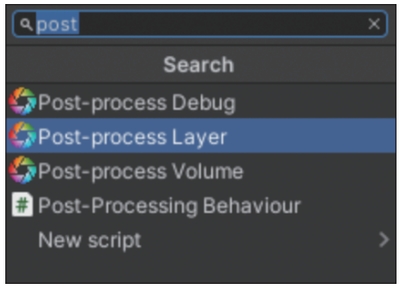


9-3-2 カメラの準備

Post Processingを使ってカメラにエフェクトをかける準備を行います。

Projectビューで「MainScene」を選択して、Hierarchyウインドウの「Main Camera」を選択し、Inspectorウインドウで「Post-process Layer」コンポーネントを追加します。

図9.26 Post Process Layerコンポーネントを追加



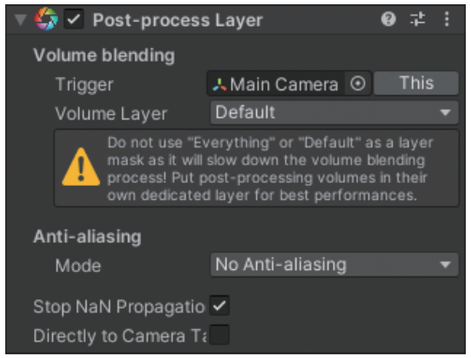
エフェクトの設定

Post-process Layerコンポーネントの Volume Layerのプロパティで指定したレイヤーに対してエフェクトがかかります。

エフェクトをかける処理は負荷が高いため、EverythingやDefaultを選択すると「処理が重くなるのでやめた方が良いよ」というニュアンスの警告が出ます。

今回は「Default」を選択して進めますが、パフォーマンスを考慮する場合はレイヤーを絞って適用するようにしましょう。

図9.27 「処理が重くなる」というメッセージ



アンチエイリアスの設定

Post-process Layerコンポーネントの Anti-aliasingでは、アンチエイリアスの設定が可能です。アンチエイリアスとは、描画物のフチのギザギザ（ジャギーと呼ばれます）を滑らかにする処理です。

これだけでもかなり見た目が変わりますので、Inspctorウインドウの Anti-aliasingの Modeで「Fast ApproximateAnti-aliasing (FXAA)」を選択し、見た目の変化を比べて見ましょう。

図9.28 アンチエイリアスの設定

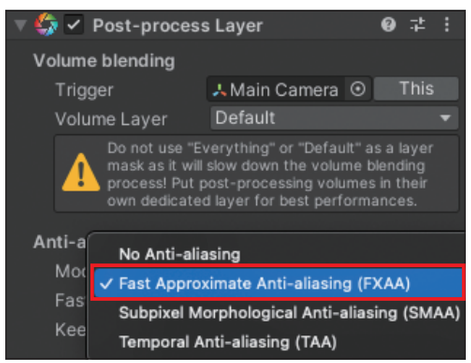
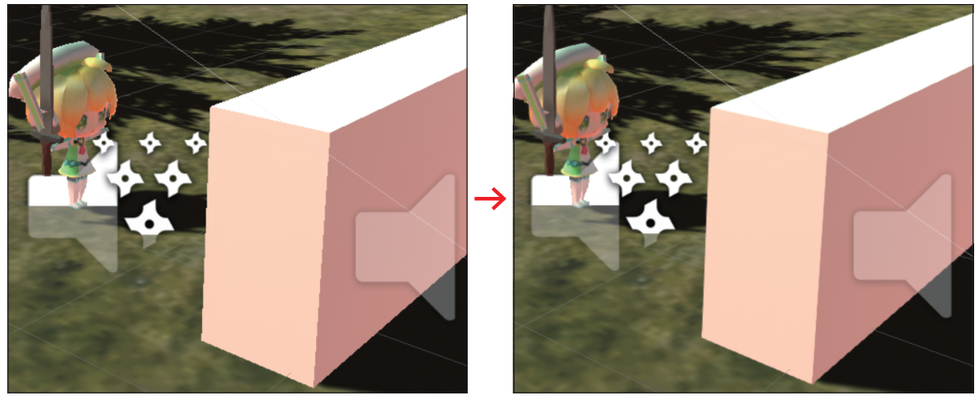


図9.29 アンチエイリアス設定前と設定後



新規カメラプロファイルの作成

続いて、カメラに対してどのようなエフェクトをどの程度かけるかを設定するプロファイルを作成します。

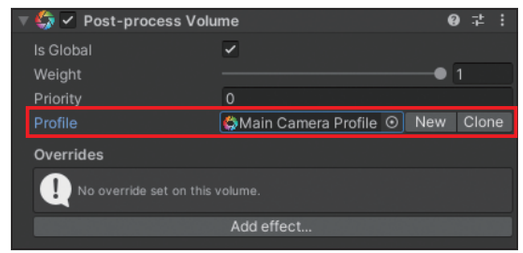
Hierarchyウインドウで「Main Camera」を選択し、Inspectorウインドウで「Post-process Volume」コンポーネントをアタッチします。

初期状態ではPost-process Volumeコンポーネントの Is Globalが OFFになっています。この状態だと、Post-process Volumeがアタッチされているゲームオブジェクトが持つColliderの範囲内にカメラが入ったときのみエフェクトが適用されます。

たとえば、「水中に入ったときに視界が青くぼやける」といった演出をする場合などには Is GlobalをOFFにしておくと良いでしょう。今回は常にエフェクトを適用したいので、Is Globalを「ON」にしておきましょう。

続いてProfileの横にある「New」ボタンをクリックすると、新規にMain Camera Profileが作成されます。

図9.30 Post-process Volumeコンポーネントの設定



12-1 Boltについて知ろう

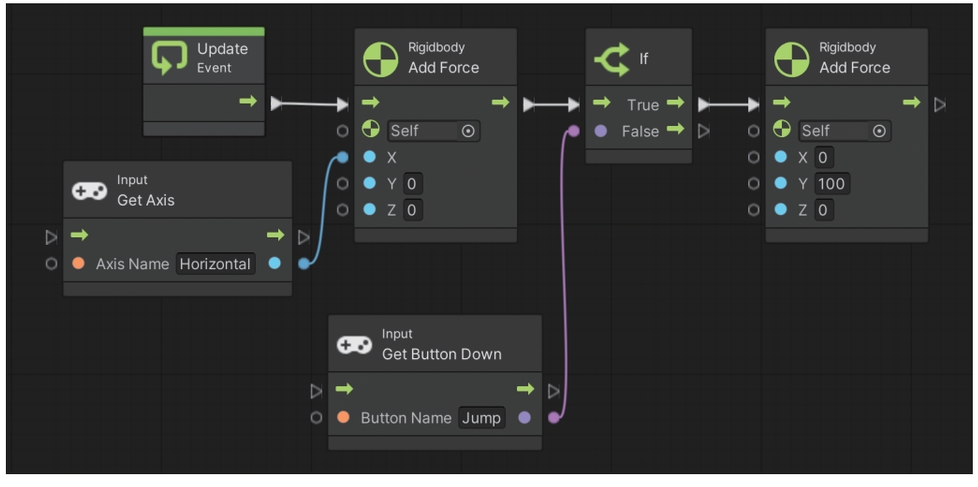
ここでは、ビジュアルスクリプティングとは何かについて解説し、Boltの導入手順や基本知識についてまとめています。

12-1-1 ビジュアルスクリプティングとは

これまでUnityでスクリプトを記述するには、C#でプログラミングするのが基本でした。これはプログラミングが苦手な方からすると高いハードルを感じてしまうものでした。

ビジュアルスクリプティングでは、プログラムを書く代わりに画面上で部品を配置していき、それを繋ぎ合わせてスクリプトを作成します。目に見える形で組み立てていくため、プログラミングの知識が浅くても、処理の流れが把握しやすいという特徴があります。

図12.1 Boltでのビジュアルスクリプティングの例



Bolt（https://unity.com/ja/products/unity-visual-scripting）は、元々Asset Storeで販売されていた有料のビジュアルスクリプティングアセットでした。2020年5月にUnityによって買収

され、現在は誰でも無料で利用可能になっています。

なお、Boltでビジュアルスクリプティングをする場合でも、プログラムの基礎知識（変数やif文など）およびUnityの基礎知識（ライフサイクルやコンポーネントなど）は必要です。本書でもChapter 3で一通り解説していますので、プログラミング未経験者の方は、あらかじめ目を通しておくようにしてください。

12-1-2 Boltの導入方法

Boltは、Unity 2021ではデフォルトで組み込まれているため、特別な設定を行わなくても利用可能です。

それ以前のバージョンを利用している方は、Asset Store（https://assetstore.unity.com/packages/tools/visual-scripting/bolt-163802?locale=ja-JP）からダウンロード・インストールを行ってください（5-1-3参照）。

図12.2 Asset StoreのBoltページ