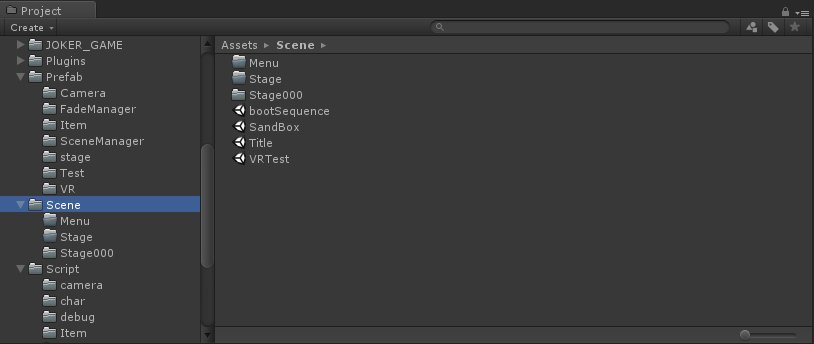
シーンの作り方

1 Assetフォルダにシーンファイルを作る



　AssetフォルダのSceneフォルダにシーンファイルを作成します。

サブフォルダは適時作成します。

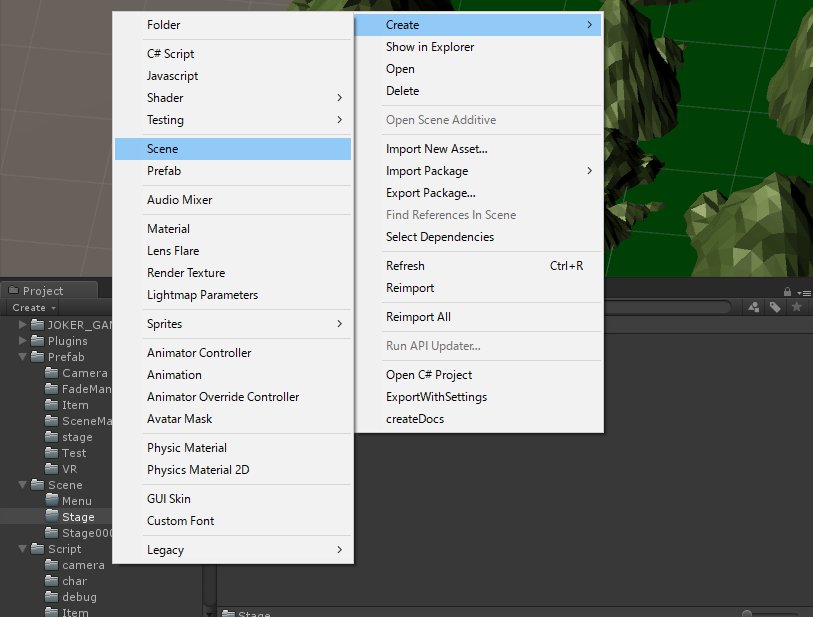
　とりあえず、

各ステージ Stage/

メニュー周り Menu/

を用意してあるので、ステージやメニュー系はサブフォルダ内に作成してください。

２ シーンファイルの作り方



空いてるところで右クリックです

ファイルを作りたい階層まで表示してから

各ファイルやフォルダが表示されているウィンドウにカーソルを持っていって

右クリック -> Create -> Sceneと選択していくとシーンファイルが作成されます。

名前をつけて完了です。

(ファイル名に命名規則は特に設けませんが、見てわかりやすい名前にしてください)

3 シーンに配置するもの

シーンには必ず配置する必要のあるGameObjectがあります。

　・GameSceneManager

・GameFadeManager

・VRManager

・Text

・各シーンのアップデーター

まずは、

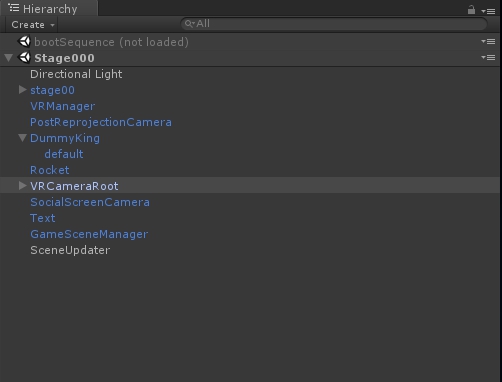
GameSceneManager

GameFadeManager

VRManager

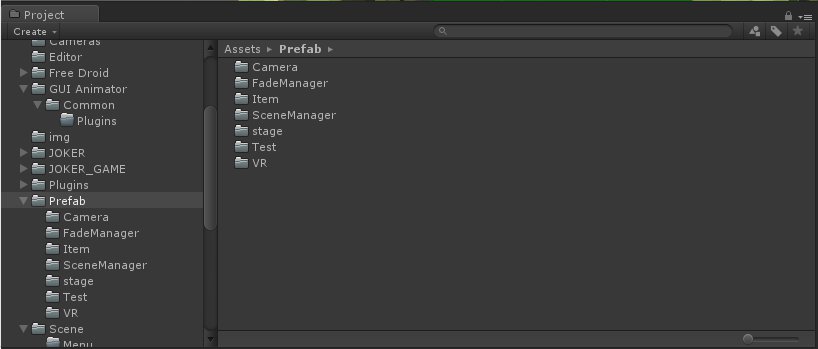
Text

について説明します。



Stage000に配置しているGameObjectです。

ここにはGameFadeManagerがないですね・・・(;’∀’)



GameSceneManager, GameFadeManager,VRManager, Textは

全部Prefab化してあります。

Asset/Prefabにそれぞれ置いてあるので、PrefabウィンドウからHierarchウィンドウにドラック&ドロップすればシーンに配置されます。

(TextオブジェクトはTestフォルダの中です。あとは何となく分かるかと思います)

配置する座標は

X:0 Y:0 Z:0

に配置すればいいと思います。

座標の指定は

座標を設定したいGameObjectを選択すると表示される

Inspectorウィンドウ内の

Transform -> Position -> XYZで指定します。

簡単な各オブジェクトの説明です。

・GameSceneManager

シーンのマネージャーです。

　　今のところ、シーン遷移を制御する機能を持っています。

・GameFadeManager

フェード処理を担当します。

・VRManager

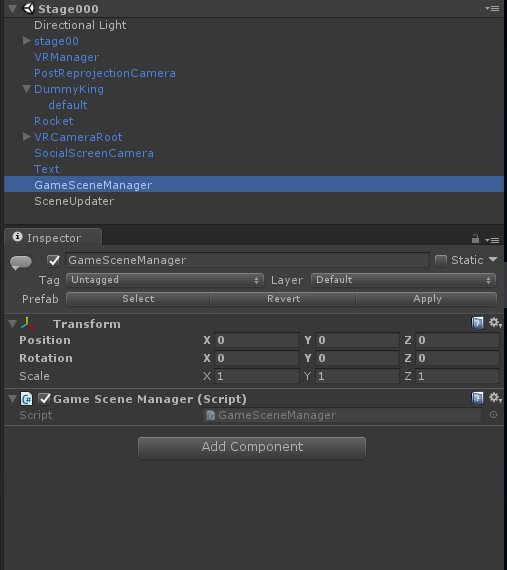
VRの制御を担当します。

(VRモードと普通の画面表示モードを切り替えたりしています)

・Text

デバック表示用のオブジェクトです。

テキスト表示とGUI表示で画面上にデバック表示を行います。



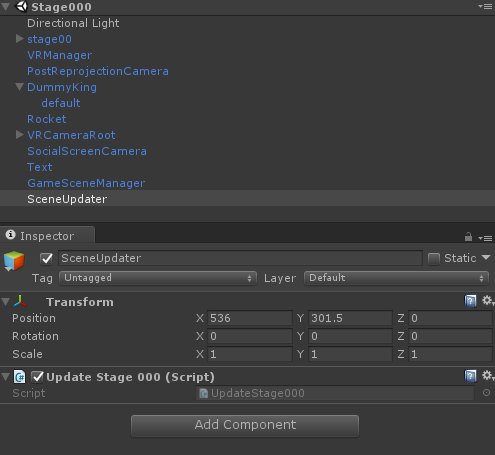
GameSceneManagerのInspectorウィンドウです。

ちなみに、Hierarchyウィンドウで青文字で表示されているオブジェクトはPrefabです。

・各シーンのアップデータについて

　　シーンアップデータは、そのシーン専用の更新処理を担当するオブジェクトです。

　　Stage000ではSceneUpdaterというオブジェクトがシーンアップデータです。



SceneUpdaterにはUpdateStage000というスクリプトだけがアタッチされています。

シーンアップデータにはUpdatexxx(xxxは各シーンの名前)というスクリプトをアタッチします。

スクリプトは

Asset/Script/scene

に作成します。

Asset/Script/scene/templateに

UpdateSceneTemplate.csという雛形スクリプトを用意しました。

このファイルをコピー、リネームして使用します。

　スクリプト作成手順

　・UpdateSceneTemplate.csをコピー、Asset/Script/sceneにペースト

　・コピペしたファイル名をUpdateシーンの名前.csにリネーム

ex:) TitleシーンのシーンアップデータTitleSceneUpdaterというオブジェクトの場合

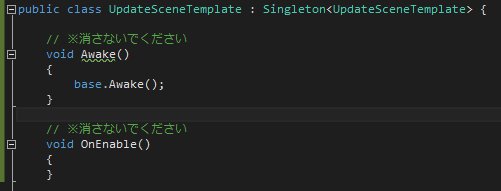
UpdateTitle.csにリネーム

　・リネームしたファイルを開いて クラス名をファイル名と同じに書き換える

Public class UpdateSceneTemplate : Singleton<UpdateSceneTemplate>

を

Public class Updateシーンの名前 : Singleton<Updateシーンの名前>



この2か所を書き換えます

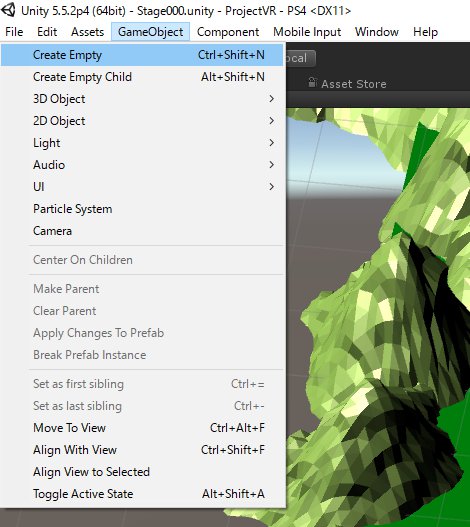
VisualSudioの場合、Ctrl + Shift + Bでビルドをします。

エラーがでないかどうかだけチェックしてください。

MonoDevelopの場合はググってください・・・

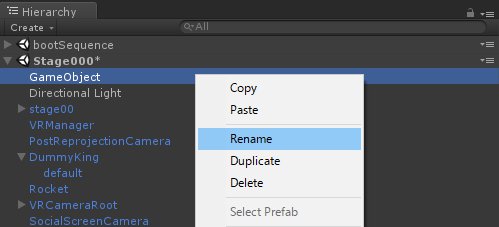
・空のGameObjectの作成

　シーンアップデータとする空のGameObjectを作成します。



メニューからGameObject -> CreateEmptyで空のGameObjectを作成します。

次に名前をUpdateシーンの名前に変更します。



最後に、作成したGameObjectに作成したスクリプトをアタッチします。

アタッチする方法は、

InspectorのAddComponentからか、

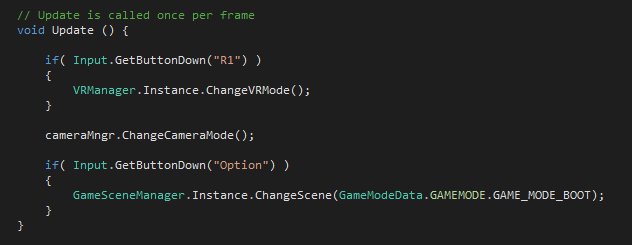
スクリプトファイルをGameObjectにドラック&ドロップです。

シーンアップデータ用のスクリプトで何するの？

　シーン遷移するためのトリガーとなる処理を記述してもらうことになります。

　Update関数の中であるボタンが押されたらシーン遷移する。

　みたいな感じです。

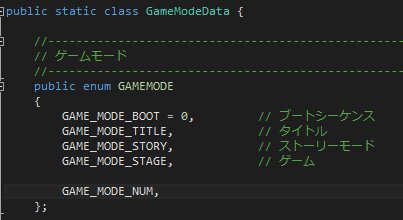


この部分は

「ゲームコントローラーのスタートボタンを押したらBootSequenceシーンに遷移します」

という意味です。

各シーンの定義名はGameModeData.csに定義されています。



ここに追加します

シーンを追加したら、ここにGAME\_MODE\_XXXX(XXXXは任意の名前)として追加してください。

スクリプトでの記述方法は

GameModeData.GAMEMODE.XXXXXX

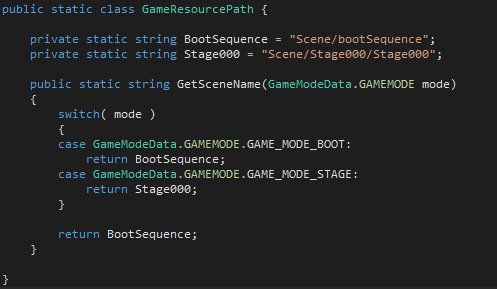
(XXXXXはそれぞれの定義名です。上の画像でいえば

GAME\_MODE\_BOOTや、GAME\_MODE\_TITLEといったものです)

・シーンパスの定義

シーンのパスを定義する方法です。

GameResourcePath.csに定義します。



1 で囲った部分に追加

private static string 変数名 = “シーンのあるパス”

変数名は任意の文字列です。わかりやすい名前を付けます。

　 シーンのあるパスはAssetフォルダ以下のパスを記述します

ファイル拡張子はいりません。ダブルクォーテーションで囲みます。

２　　　　で囲った部分に追加

　　case GameModeData.GAMEMODE.XXXXXX:

return 変数名;

GameModeData.GAMEMODE.XXXXXXはGameModeDataで定義した定義名です。

　　return 変数名の変数名は１で追加したシーンパスです。

・・・・・・・・・・・

・・・・・・・・

・・・・・・

・・・

・・

・

シーンの定義や、シーンパスの定義はかなりややこしいと思うので鈴木の方で対応します。

シーンを追加したら連絡をください。

宜しくお願いします。