#24 エディタ拡張

1. Editor 拡張の利用

Editor 拡張を利用するためには、ディレクトリの準備が必要。

○Asset 下に “Editor” フォルダを作成し、その中にエディタ拡張用のファイルを保存。

○フォルダ内には、複数のエディタ拡張のクラスファイルを保存できる。

1.1. エディタ拡張の基本構造

エディタ拡張用 C#ソースコード、および注意点は下記の通り。

○ “UnityEditor” 名前空間の利用

○ “MonoBehaviour” ではなく “EditroWindow” クラスを継承する。

○ “MenuItem” でエディタ拡張の表示名を設定する。

○ “EditorWindow.GetWindow” にて Window 情報を取得する。

EditorEx.cs

```
 1  using UnityEngine;
 2  using UnityEditor;
 3  using System.Collections;
 4
 5  public class EditorEx : EditorWindow
 6  {
 7      [MenuItem("Window/EditorEx")]
 8      static void Open()
 9      {
10          EditorWindow.GetWindow<EditorEx>( "EditorEx" );
11      }
12
13      void OnGUI()
14      {
15          EditorGUILayout.LabelField( "Open Editor 拡張ウィンドウ" );
16      }
17  }
```

保存後、Unity Editor のメニューから Window>EditorEx を選択することで、設定した Editor 拡張用のメニューが開く。

1.2. 確認：課題

Assets フォルダ内に Editor フォルダーが作成できている。	10 点
UnityEditor 名前空間が追加できている。	10 点
“EditorWindow” クラスを継承している。	10 点
MenuItem を適切に設定し、指示通り Window -> EditorEx がプルダウンメニューに表示されている	10 点
OnGUI()内で EditorGUILayout.LabelField を設定し、表示したエディタ拡張ウィンドウ内にコメントを表示できている。	10 点

2. GUILayout の利用

GUILayout クラスは MonoBehaviour を基底クラスとした class in UnityEngine GUI の自動レイアウトを行うためのインターフェースである。

GUILayout リファレンス
https://docs.unity3d.com/ja/current/ScriptReference/GUILayout.html

BeginArea	固定されたスクリーン領域に GUI コントロールの GUILayout ブロックを開始する
EndArea	BeginArea で開始した GUILayout ブロックを閉じる

BeginHorizontal	水平のコントロールグループを開始する
EndHorizontal	BeginHorizontal で開始したグループを閉じる

BeginScrollView	自動的にレイアウトされるスクロールビューを開始する
EndScrollView	BeginScrollView を呼び出して開始したスクロールビューを閉じる

BeginVertical	垂直のコントロールグループを開始する
EndVertical	BeginVertical で開始したグループを閉じる

Box	自動レイアウトのボックスを作成する
Button	Make a single press button.
ExpandHeight	コントロールの垂直方向の拡張を許可/禁止するオプション
ExpandWidth	コントロールの水平方向の拡張を許可/禁止するオプション
FlexibleSpace	フレキシブルなスペースを挿入する
Height	決められた高さをコントロールに与えるオプション
HorizontalScrollbar	水平のスクロールバー
HorizontalSlider	ユーザーが最小値と最大値の間で値をドラッグで変更できる水平スライダー
Label	自動レイアウトのラベル
MaxHeight	コントロールの高さの最大値を設定するオプション
MaxWidth	コントロールの幅の最大値を設定するオプション
MinHeight	コントロールの高さの最小値を設定するオプション
MinWidth	Option passed to a control to specify a minimum width.
PasswordField	パスワードを入力するフィールドを作成する。
RepeatButton	リピートボタン。ユーザーがボタンをマウスで押している間は true を返す。
SelectionGrid	選択グリッドボタン
Space	現在のレイアウトグループにスペースを挿入する
TextArea	ユーザーが文字列を編集することができる複数行のテキストエリア
TextField	ユーザーが文字列を編集することができるテキストエリア
Toggle	on/off のトグルボタン
Toolbar	ツールバー
VerticalScrollbar	垂直のスクロールバー
VerticalSlider	ユーザーが最小値と最大値の間で値をドラッグで変更できる垂直スライダー
Width	決められた幅をコントロールに与えるオプション
Window	ウィンドウ内のコンテンツが自動でレイアウトされるポップアップウィンドウ

2.1. 代表的な機能

2.2. Label

項目名などの表示

GUILayout.Label(StringText);

2.3. Button/RepeatButton

入力用ボタン、Button は立ち上がりのみ、RepeatButton はボタンを押している間 True

ex)

```
if( GUILayout.Button( "Button" ) )  Debug.Log( "Button!" );
if( GUILayout.RepeatButton( "RepeatButton" ) )  Debug.Log( "RepeatButton!" );
```

2.4. TextArea/TextField

文字入力用フィールド

ex)

```
textArea = GUILayout.TextArea( textArea );
textField = GUILayout.TextField( textField );
```

2.5. HorizontalScrollbar/VerticalScrollbar

スクロールバー

```
private float hScrollvalue = 0.0f;
private float vScrollvalue = 0.0f;
            :
hScrollvalue = GUILayout.HorizontalScrollbar(
              hScrollvalue ,    // value: 初期値
              10.0f,   // size : ボタンサイズ
              0.0f ,    // leftValue 最小値
              100.0f   // rightValue 最大値
);

vScrollvalue = GUILayout.VerticalScrollbar(
              vScrollvalue ,    // value: 初期値
              10.0f,   // size : ボタンサイズ
              0.0f ,    // leftValue 最小値
              100.0f   // rightValue 最大値
);
```

2.6. 確認：課題

各機能に対して適切なラベルが表示されている。	10 点
Button/RepaetButton を押すことで、コンソール上に文字が表示できる。	10 点
TextArea/TextField を作成できている。	10 点
HrizontalScrollbar が指定通り表示できている。 ※指定値	10 点
verticalScrollbar が指示通り表示できている。 ※指定値	10 点

※指定値

初期値 5.0f , ボタンサイズ 5.0f 、 最小値 2.0f 、 最大値 50.0f

2.7. EditorGUILayout の利用

EditorGUILayout クラスは、EditorWindow を基底クラスとした GUI の自動レイアウトを行うためのインターフェースである。

EditorGUILayout リファレンス
https://docs.unity3d.com/ja/current/ScriptReference/EditorGUILayout.html

いくつか、その要素を取り上げる。

2.7.1. Foldout

指定された範囲の要素を、折りたたむことができる。

```
public static bool Foldout(
    bool foldout,           //状態の指定
    string content,         //表示されるラベル
    );
```

利用例

EditorEX02.cs

```
1   using UnityEngine;
2   using UnityEditor;
3
4   public class EditorEX02 : EditorWindow
5   {
6           //開いているか
7           private bool _isOpen = true;
8
9           [MenuItem("Examples/EditorEX02")]
10          // Use this for initialization
11          static void Open()
12          {
13          EditorEX02 window = (EditorEX02)GetWindow(typeof(EditorEX02));
14          }
15
16          private void OnGUI()
17          {
18          //折りたたみ UI を表示、現在開いているかを取得
19          bool isOpen = EditorGUILayout.Foldout(_isOpen, "折りたたみ");
20
21                  //状態の保存
22                  _isOpen = isOpen;
23
24          //開いている時は GUI 追加
25          if (isOpen)
26          {
27                  EditorGUILayout.LabelField("    表示しています。");
28          }
29          }
30  }
```

2.7.2. ColorField

色選択

Color colorField = Color.white;

colorField = **EditorGUILayout.ColorField**("ColorField", colorField);

2.7.3. Vector2/3/4 Field

Vector 要素入力フィールド。

```
//選択したオブジェクトの位置を表示
if(Selection.activeTransform)
                    EditorGUILayout.Vector3Field("Position", Selection.activeTransform.position);
else
                    EditorGUILayout.LabelField("Select a GameObject");

Repaint();
```

2.7.4. 確認：課題

エディタメニューExamples 以下にメニューがあり、選択することで EditorWindow が開く。	10 点
ColorField で色選択 UI が表示できる。	10 点
ColorField で選択した色の RGBA 成分を表示できる。	10 点
Vector3 入力フィールドを表示できる。	10 点
上記の内容を折りたたみ表示できる	10 点

3. Editor 拡張ウィンドウでの情報取得

エディター拡張ウィンドウで、情報の取得をしたい！

→ EditorGUILayout/GUILayout にて適切な Field を利用する。

文字、数字	TextField
Object	ObjectField

3.1. 数値の入力

TextField にて文字列として取得→int 型に変換

○受け取る変数の定義

```
private  int  num;
```

○値の取得

```
num= int.Parse(     // int への変換
            //テキスト情報として入力
            EditorGUILayout.TextField("数値", num.ToString())
);
```

3.2. Object の取得

ObjectField にて取得

○受け取る変数の定義

```
public  GameObject  target;
```

○値の取得

```
    target
 = EditorGUILayout.ObjectField( "目標",  target, typeof(GameObject), true) as  GameObject;
```

3.3. Object の生成

○Instantiate にて生成

○利用例

```
GameObject obj  = Instantiate(prefab, pos, Quaternion.identity)  as  GameObject;
```

"prefab" オブジェクトを "pos" の位置に生成 、Quaternion.identity は "回転させない"してい

3.4. アクティブな Object の取得

Scene ビュー上でクリックしたオブジェクトの取得

"Selection.GameObject" で取得できる。