Unityはじめるよ
～基礎２～

統合開発環境を内蔵したゲームエンジン
http://japan.unity3d.com/

※いろんな職業の方が見る資料なので説明を簡単にしてある部分があります。正確には本来の意味と違いますが上記理由のためです。ご了承ください。
この資料内の一部の画像、一部の文章はUnity公式サイトから引用しています。

本日勉強する内容

Unityの基本構造

- Unityの世界はGameObjectだらけ
- Unityでスクリプティング
  スクリプト実行順の話
  コルーチンの話
  体験
  Transformアクセスの話

■Unityの世界は GameObjectだらけ

キャラなどのオブジェクト、カメラ、ライト、サウンドなど、
すべて**GameObject**である。
GameObjectに**コンポーネント**追加することで、
色んな振る舞いをするようになる。

コンポーネントとは
ゲームを作成するのに便利な機能を持った部品(機能)のこと

GameObjectにはデフォルトで**Transform**というコンポーネントが追加されている。
Trasnformは、位置、回転、拡大率を持つコンポーネントである。
ちなみにTransformコンポーネントは削除することはできない。

コンポーネント

ドラゴン3Dモデル

GameObject

位置、回転、拡大縮小

コンポーネント

火を吐かせる
スクリプト

コンポーネント

アニメーション
データ

コンポーネント

空を飛ばす
スクリプト

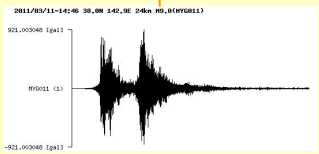
コンポーネント

ドラゴンマテリアル
（テクスチャ＆質感）

コンポーネント

物理演算

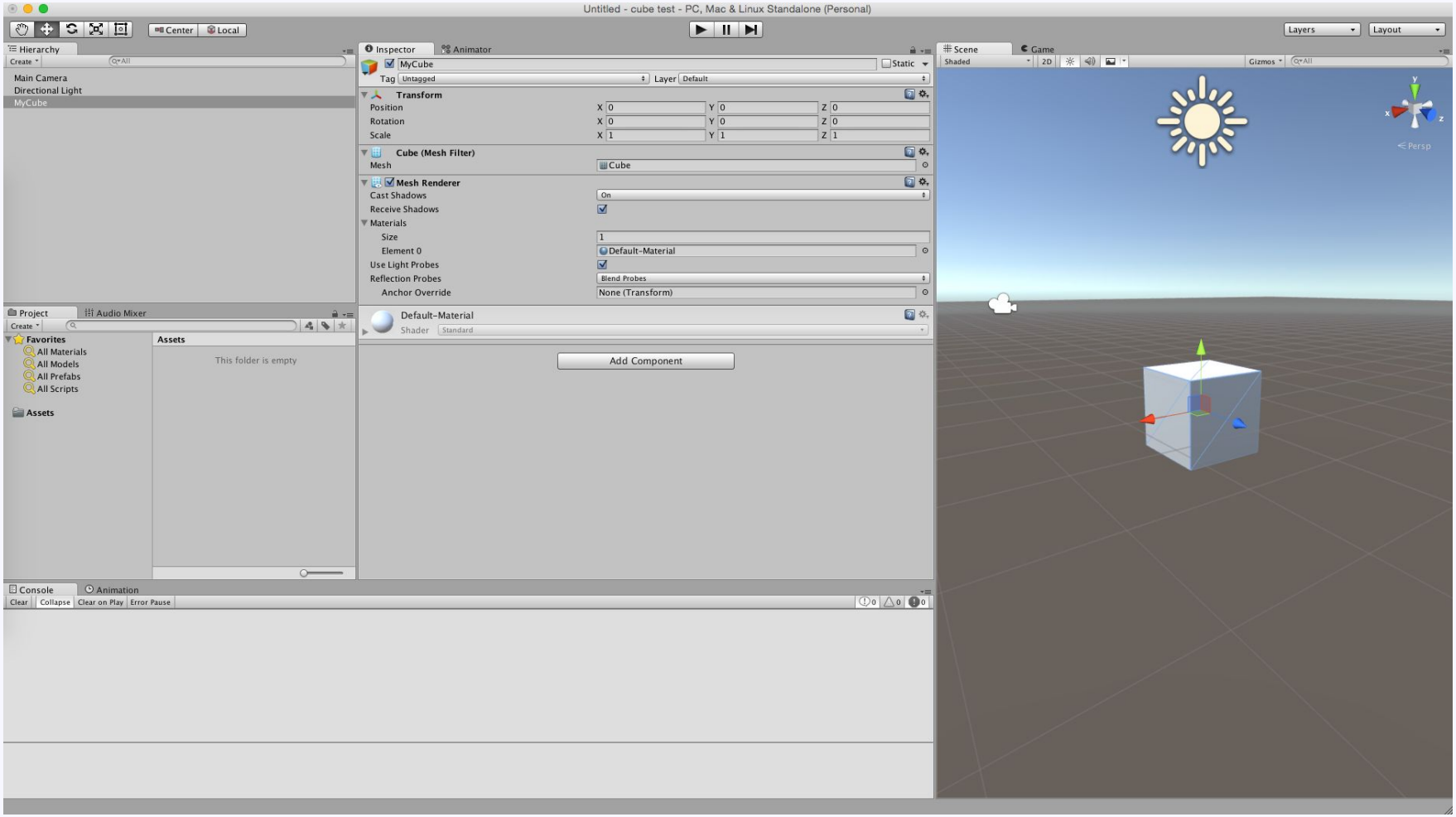
コンポーネント



雄たけびサウンド

「GameObjectにコンポーネントを追加してみよう」

空のGameObjectにコンポーネントを追加し
Cubeを描画できるようにする



手順

1、空のGameObjectをシーンに追加
メニュー > GameObject > CreateEmpty　※Positionを(0,0,0)にしておく

2、Mesh情報を追加する（※Meshとはポリゴンの頂点データのこと）
インスペクタービュー > Add Componentボタン > Mesh > MeshFilter >
追加したコンポーネントのMesh > Cube

3、描画するためのレンダラを追加
インスペクタービュー > Add Componentボタン > Mesh > MeshRenderer
Materals > Element0 > 適当なマテリアルを選択
MeshRendererの設定項目
・Cast Shadows 影の影響を与えるか。自分の影を相手に表示するかどうか
・Receive Shadows 影の影響を受けるか。相手の影と自分の影を、自分に表示するかどうか
・Materials Meshの質感情報
・Use Light Probes ライトプローブの影響を受けるか
・Reflection Probes リフレクションプローブの影響を受けるか

4、当たり判定を追加
インスペクタービュー > Add Componentボタン > Physics > BoxCollider

※ライトプローブとは簡易ライトのこと。動的なGameObjectに通常のライトをたくさん利用すると、描画負荷は相当高いものとなってしまう。
ライトプローブを使うと、似たような表現が低い描画負荷で実現できる。
静的なGameObjectの場合は、ライトマップを使うと良い。（ライトマップは事前に影をテクスチャに焼き込んでおく方法）

※リフレクションプローブとは反射表現手法の一つ。低い描画負荷で自然な反射を表現できる

■Unityでスクリプティング

スクリプトを使ってGameObjectを制御することで、
表現の幅が大きく広がる。

スクリプトもコンポーネントの一種である。
ビルド時に中間コードを生成するので、
C#、JavaScript、Booのどの言語で書いても実行速度は同じ。

※Unityは.NET Frameworkの互換環境である「Mono」上で動作する
　Unity5からIL2CPPという技術でMonoの中間言語の「IL」を「C++」に変換することで、ネイティブ速度で実行できるようになった

注意点
・ファイル名とクラス名を揃える必要がある
・スクリプトの実行順に気を付ける

スクリプトの実行順の話

スクリプトはコンポーネントの一つと話したように、
色んなGameObjectにスクリプトを追加することができる。
どんな順番でスクリプトを実行するかは、

Edit > Project Settings > Script Execution Order

から指定することができる。
実行順を指定しているスクリプトの中では、数字が小さいスクリプトほど先に実行される。
ただし、同じスクリプトを複数のGameObjectに追加してる場合の実行順は未確定となる。
また、実行順を指定してないスクリプトの実行順番も未確定となる。

Unity公式ドキュメントのスクリプト実行順設定に関するページ
http://docs-jp.unity3d.com/Documentation/Components/class-ScriptExecution.html

こんな感じでスクリプトがいっぱいあるとするじゃんね

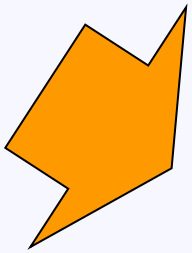
スクリプト２

スクリプト１

スクリプトB

スクリプト３

スクリプトA



そうすると、実行順を指定した箇所に関しては、指定した順番で実行してくれる。
指定してないやつは相変わらず。

スクリプトB

ここだけ
実行順番を指定

スクリプト１

スクリプト２

スクリプト３

スクリプトA

Unityが自動で呼び出す関数

ライフサイクルやイベント通知など、MonoBehaviourを継承したクラスがUnityから呼ばれる関数の表。

特定のタイミングで呼ばれる関数

関数名	呼ばれるタイミング
最初のシーンロード	
Awake	常に Start 関数の前およびプレハブのインスタンス化直後
OnEnable	オブジェクトを有効化した直後
最初のフレームのアップデート前	
Start	最初のフレームのアップデート前
フレーム間	
OnApplicationPause	フレームの終わり、ポーズが検知されるとき
オブジェクト破棄時	
OnDestroy	オブジェクトの生存期間の最後のフレームのフレーム更新の後
終了時	
OnApplicationQuit	アプリケーション終了前
OnDisable	動作が無効になるタイミング

毎フレーム呼ばれる関数

アップデート順		
関数名	呼ばれるタイミング	1フレームで呼ばれる回数
FixedUpdate	信頼できるタイマーで呼び出される	フレームレートとは独立して動作 つまり1フレームに 複数回呼ばれることもある
Update	毎フレーム呼ばれる	1回
LateUpdate	Update後に呼ばれる	1回

レンダリング		
関数名	呼ばれるタイミング	1フレームで呼ばれる回数
OnPreCull	カメラがシーンを間引く前 つまりカリングが発生する直前	1回
OnBecameVisible OnBecameInvisible	オブジェクトがカメラに対して表示 または非表示になる際	1回
OnWillRenderObject	オブジェクトが表示される際 各カメラに対して1回づつ呼ばれる	1回
OnPreRender	カメラがシーンのレンダリングを開始する前	1回
OnRenderObject	すべてのシーンレンダリング終了後	1回
OnPostRender	カメラがシーンのレンダリングを終了した後	1回
OnRenderImage (Pro only)	画面レンダリングが完了し 画面画像の処理が可能になった後	1回
OnGUI	イベントに応じて、フレームごとに複数回	複数回
OnDrawGizmos	可視化のためにシーンビュー内での ギズモの描画	1回

コルーチンの話

コルーチンとは処理をメインループ以外の場所で行わせる仕組みである。

コルーチンを呼ぶと、以後、毎フレーム（または指定秒数）自動的に処理を行ってくれる。
なので、一定期間のみ動かす処理や、一定の秒数ごとに動かしたい処理に有効である。

一定期間のみ動かす処理…フェード処理など
一定の秒数毎に動かしたい処理…敵を探索など

「スクリプトでGameObjectを制御してみよう」

前回作った「玉が転がるプロジェクト」の玉を制御する

やりたいこと

地面から落ちた玉を初期位置に戻して、永遠が玉を転がす。

手順

1、玉を選択 > インスペクタービュー > AddComponent > NewScript(C Sharp) > 名前を
BallManagerにする

2、AssetsにBallManagerのC#ファイルをダブルクリックでエディタ起動
デフォルトで、StartとUpdate関数が記述してある。

TIPS:よく使うオーバーライド関数紹介
Awake スクリプトが起動された直後に1回だけ呼ばれる関数
Start Updateが呼び出される前に一回だけ呼ばれる関数
Update 毎フレーム呼ばれる関数
LateUpdate Updateが実行された後に呼ばれる関数
FixedUpdate　フレームレートには関係なく一定(デフォルト1/100秒)の時間毎に呼ばれる関数

3、Update関数の中に、ボールのY座標が-10以下になったら、開始地点に戻すという処理を追加する。

玉の座標にスクリプトからアクセスするには？
スクリプトが追加されているGameObjectにアクセスするには「gameObject」を使う
位置や姿勢情報を扱うTransformコンポーネントにアクセするには「transform」を使う
Transformコンポーネントの位置情報にアクセスするには「Vector3型のposition」を使う
つまり、位置情報にアクセスするには、
　　gameObject.transform.position
とすればよい。
y座標にアクセスするなら、
　　gameObject.transform.position.y
となる。

サンプルプログラム

```
using UnityEngine;
using System.Collections;

public class BallManager : MonoBehaviour {

		private Vector3 mStartPos;

		// Use this for initialization
		void Start () {
			// 開始時の座標を記録
			mStartPos = gameObject.transform.position;
		}

		// Update is called once per frame
		void Update () {
			// y座標が-10より小さくなったら、開始時点の座標に戻す
			if (gameObject.transform.position.y < -10) {
				gameObject.transform.position = mStartPos;
			}
		}
}
```

Transformのアクセスの話

スクリプトから位置や回転、スケールを制御する場合、方法が複数存在する。
それぞれ違う意味を持っているので注意が必要。

例えば回転を制御する場合、

transform.eulerAngles	オイラー角としての角度
transform.localEulerAngles	親の Transform オブジェクトから見た相対的なオイラー角としての回転値
transform.rotation	Quaternion として保存されるワールド空間での Transform の回転
transform.Rotate	Z 軸で eulerAngles.z 度回転、X 軸で eulerAngles.x 度回転、Y 軸で eulerAngles.y 度回転します (順番は説明した順)
transform.RotateAround	ワールド座標の point を中心とした軸(axis)で angle 度回転させます

これだけの制御方法がある

詳細はUnity公式マニュアルを参照
http://docs.unity3d.com/jp/current/ScriptReference/Transform.html

ご清聴ありがとうございました