3-1 ゲームの設計を考えよう

3章ではこれからの学習のウォーミングアップとして、簡単なゲームを作ってみましょう。最初から、あれもこれも詰め込んだ大規模なゲームを作ろうとすると、途中で挫折してしまうかもしれません。そこで最初は簡単なゲームから作り始めて、少しずつ複雑なゲーム作りに挑戦していきましょう。この章ではオブジェクトの動かし方から学び始めます！

3-1-1 ゲームの企画を作る

簡単といっても、ただ画面に画像が表示されているだけではゲームと言えません。ゲームには最低限、**ユーザの入力によって動くもの**が必要です。そこで3章では、タップで占いができる「占いルーレット」を作ってみましょう。

Fig.3-1が、これから作るゲームのイメージです。画面上に大きなルーレットが表示されており、画面をタップするとルーレットが回転を始め、時間が経つにつれて回転速度が遅くなり最後に停止します。

Fig.3-1 これから作るゲームの画面

3-1-2 ゲームの部品を考える

Fig.3-1のゲームイメージをもとに、ゲームの設計を考えてみましょう。本書でゲーム設計をする際には、次の5ステップで考えていきます。このステップにしたがって設計を考えることで、さまざまなゲームが機械的に設計ができます。

なお、3章で作るゲームは単純なのでステップ③とステップ④は不要です。ここでは軽く確認しておき、この後の章で必要に応じて詳しく説明します。

Step❶ 画面上のオブジェクトをすべて書き出す
Step❷ オブジェクトを動かすためのコントローラスクリプトを決める
Step❸ オブジェクトを自動生成するためのジェネレータスクリプトを決める
Step❹ UIを更新するための監督スクリプトを用意する
Step❺ スクリプトを作る流れを考える

ステップ① 画面上のオブジェクトをすべて書き出す

このステップでは画面上にあるオブジェクトを書き出します。Fig.3-1のゲームイメージを見ながら、ゲームに出てくるオブジェクトを探します。ゲームの画面上には「ルーレット」と「針」がありますね。3章のゲームは単純なのでこの2つしかありませんが、大きなゲームになると、ここで書き出す個数も増えます。

Fig.3-2 画面上のオブジェクトを書き出す

ルーレット　　針

ステップ② オブジェクトを動かすためのコントローラスクリプトを決める

次に、ステップ①で書き出したオブジェクトのなかから、動くオブジェクトを探してみましょう。ここで作るゲームでは、ルーレットは回転するので動くオブジェクトに含めてよさそうです。針は動かないので動くオブジェクトには含めません。

Fig.3-3 動くオブジェクトを書き出す

ルーレット　　針

動くオブジェクトには、オブジェクトの動きを制御する台本が必要です。オブジェクトを動かすための台本を、本書ではコントローラスクリプトと呼びます。今回のゲームでは、動くオブジェクトとして「ルーレット」があるので、「ルーレットの台本（ルーレットコントローラ）」を用意します。

Fig.3-4 コントローラスクリプトとは？

必要なコントローラスクリプト

・ルーレットコントローラ

ステップ③ オブジェクトを自動生成するためのジェネレータスクリプトを決める

このステップでは、ゲームプレイ時に生成されるオブジェクトを探します。敵キャラやステージの足場など、**プレイヤの移動や時間経過によって出現するもの**がこれに当てはまります。ゲームプレイ時にオブジェクトを作成するスクリプトを、本書ではジェネレータスクリプトと呼びます。ジェネレータスクリプトをは**オブジェクトを生成する工場**のようなものです。この章で作成するゲームでは、該当するものはありません。5章で出てくるので、その時に詳しく説明します。

ステップ④ UIを更新するための監督スクリプトを用意する

ゲームのUI（ユーザインターフェイス）を操作したり進行状況を判断したりするために、ゲーム全体を見渡せるスクリプトが必要となります。本書ではこれを監督スクリプトと呼びます。この章で作成するゲームにはUIがなく、ゲームの流れもシンプルなので監督スクリプトは用意しません。

ステップ⑤ スクリプトを作る流れを考える

ステップ④までに書き出した各スクリプトから、どのようにゲームを作っていくかを考えます。基本的には「**コントローラスクリプト**」→「**ジェネレータスクリプト**」→「**監督スクリプト**」の順番で作っていきます（Fig.3-5）。

3章で作る必要があるスクリプトは、ルーレットコントローラだけですね。オブジェクトの配置方法などUnityの基本操作を除けば、「**ルーレットコントローラさえ作成できればゲームが動く**」ということです。なんとなくできる気になってきませんか？

Fig.3-5 スクリプトを作る流れ

それでは、ルーレットコントローラにどのような動きをさせるのかを、簡単に確認しておきましょう。

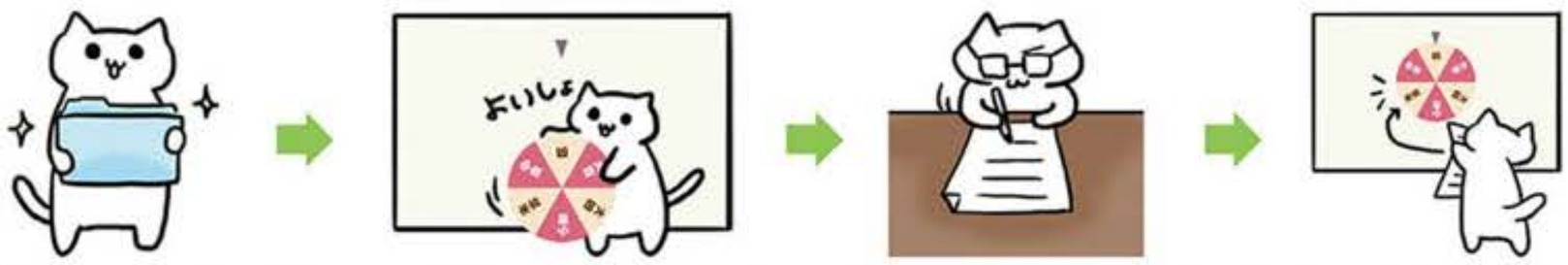
ルーレットコントローラ

ルーレットはタップすると回転を始めて、徐々に減速します。この動作をルーレットコントローラに記述します。具体的な方法はスクリプトに記述する項（130ページ）で考えましょう。

「この段階でしっかり設計をしなきゃ！」と堅苦しく考えてしまうと、ゲームを作り始める前からいやになってしまいます。**大切なのは、「どんなスクリプトを作るのか」「どういう手順で作るのか」の全体像が見渡せることです。**個々のスクリプトについては「こんな動作を実現したい」ぐらいのフワッとした感じで考えておくとよいと思います。

ここまで、理論的なお話ばかりでした、いよいよ次からは、実際に手を動かしながら占いルーレットを作っていきます！　ゲーム作成の手順は、Fig.3-6のようになります。

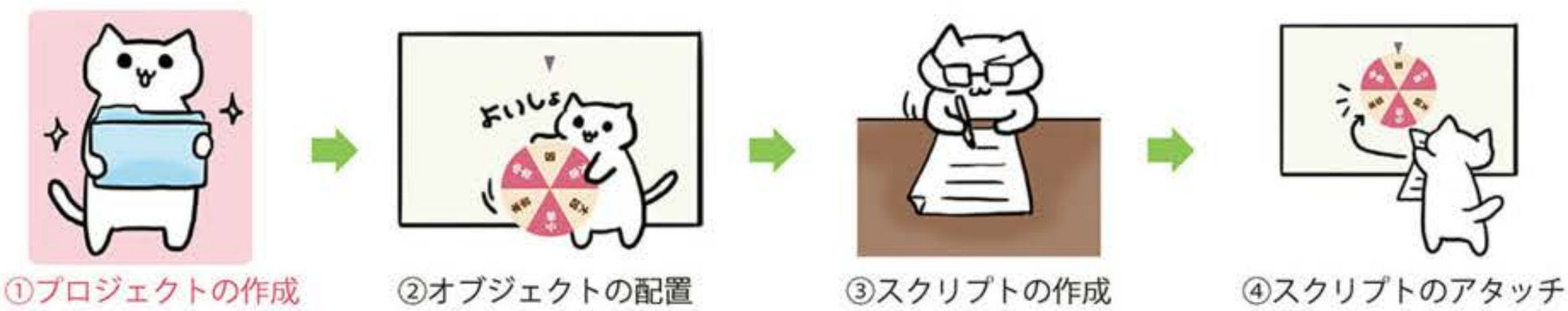
Fig.3-6 ゲームを作る流れ



①プロジェクトの作成　②オブジェクトの配置　③スクリプトの作成　④スクリプトをアタッチ

作ろうとしているゲームが単純なので、5ステップに沿って考えるのは面倒に感じたかもしれません。しかし、作るゲームの規模が大きくなった場合、このステップに沿ってゲーム設計をすることで、設計が後から破綻することが少なくなります。ゲームの規模が小さいうちに、この設計方法に慣れておきましょう！

3-2 プロジェクトとシーンを作成しよう



3-2-1 プロジェクトの作成

まずはプロジェクトを作成します。プロジェクトを作成するには、Unity Hubを起動した時に表示される画面から新規作成をクリックするか、画面上部のツールバーからFile→New Projectを選択します。

Fig.3-7 プロジェクトの作成画面

「新規作成」をクリックするか「New Project」を選択すると、プロジェクトの設定画面に進みます。

プロジェクト名を「Roulette」とし、ここでは2Dゲームを制作するので、テンプレートの項目は「2D」を選択します。画面右下にある青色の作成ボタンをクリックすると、指定したフォルダにプロジェクトが作成され、Unityエディタが起動します。

Fig.3-8 プロジェクトの設定画面

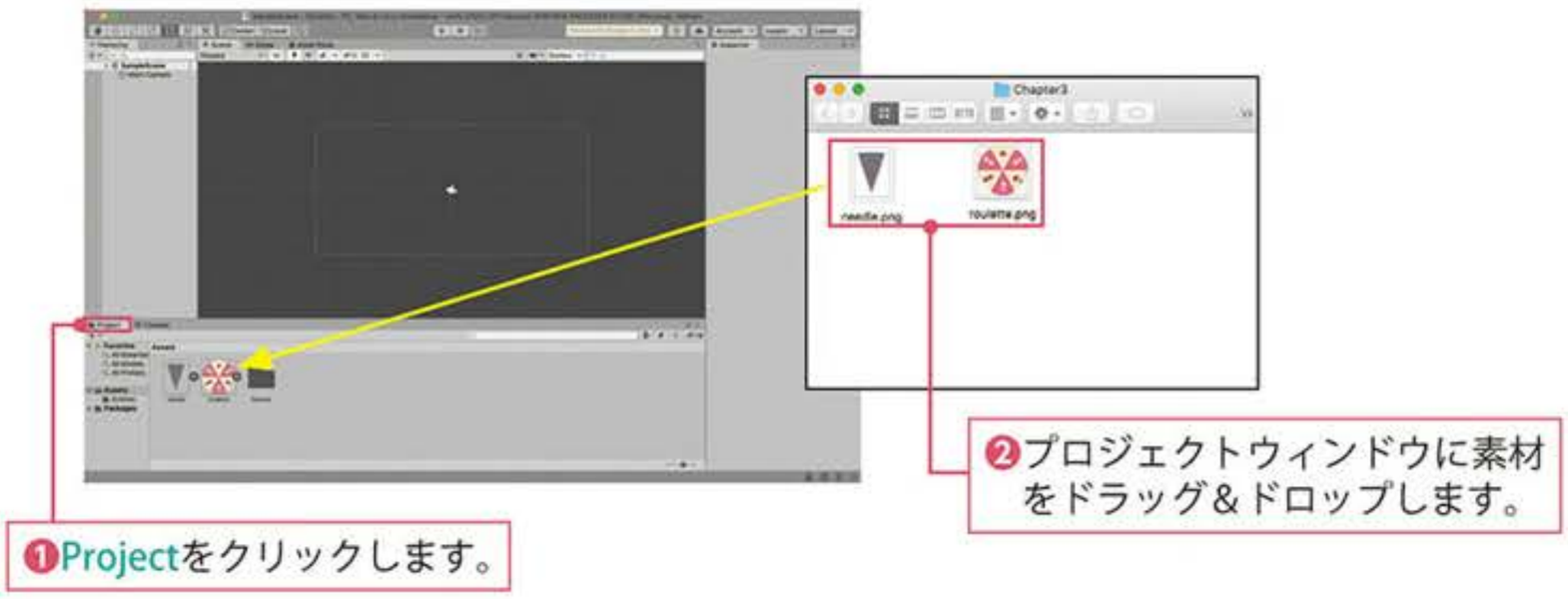
プロジェクトに素材を追加する

Unityエディタが起動したら、今回のゲームで使用する素材をプロジェクトに追加しましょう。本書のサポートページからダウンロードした素材ファイルの「Chapter3」フォルダを開いて、中身の素材をプロジェクトウィンドウにドラッグ＆ドロップしてください（ドラッグ&ドロップする際には、プロジェクトウィンドウの左上にある「Project」タブが選択されていることを確認してください）。

URL 本書のサポートページ

https://isbn2.sbcr.jp/06657/

Fig.3-9 素材を追加する



各ファイルの役割は、以下の通りです。

Table 3-1 使用する素材の形式と役割

ファイル名	形式	役割
needle.png	pngファイル	針の画像
roulette.png	pngファイル	ルーレットの画像

Fig.3-10 使用する素材

needle.png

roulette.png

実行時の描画設定

実行したときのフレームの描画速度をモニターの更新速度に合わせるため、Gameタブをクリックして左上にある画面サイズ設定（Aspect）のドロップダウンリストを開き、VSync（Game view only）にチェックを入れてください。

Fig.3-11 Lightingの設定

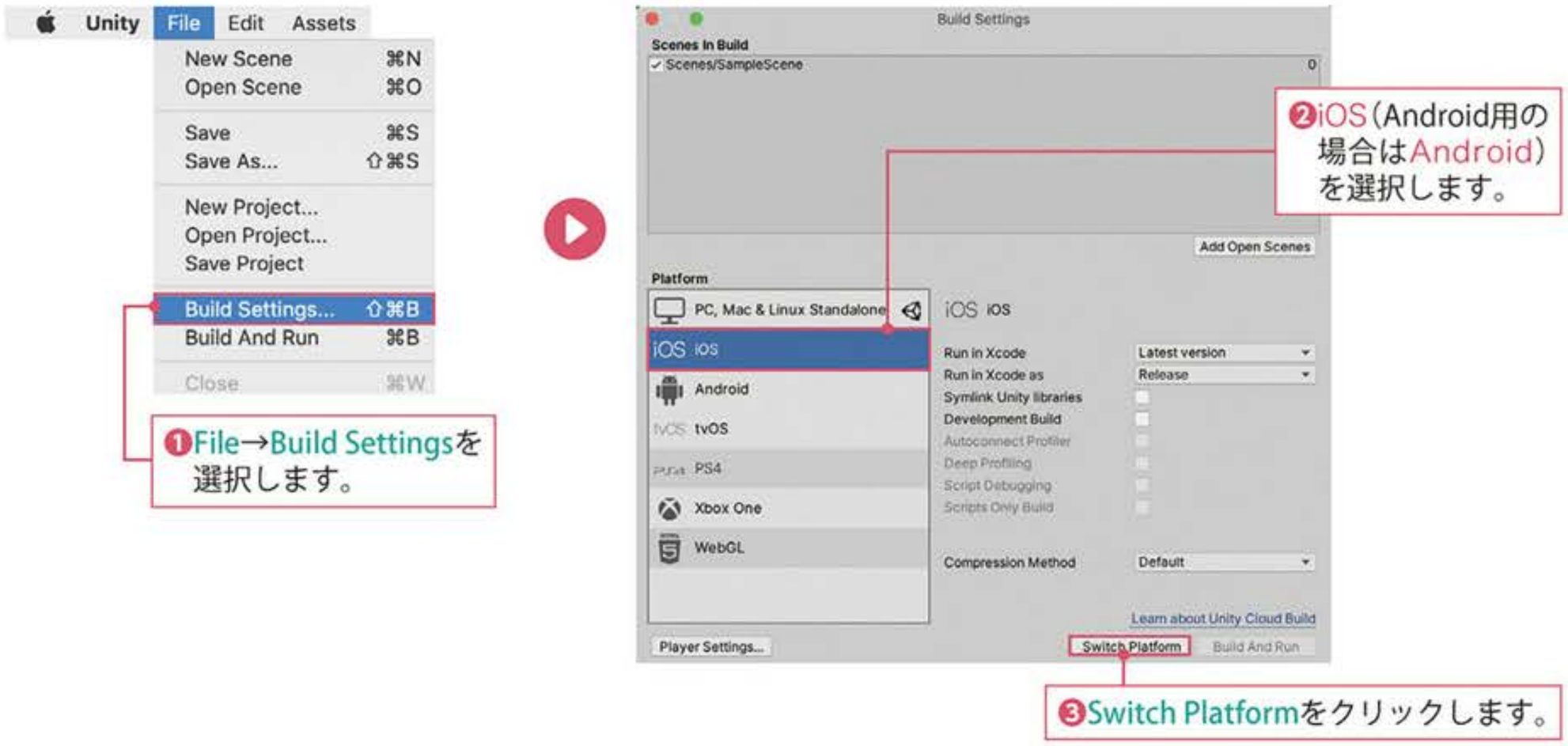
3-2-2 スマートフォン用に設定する

本書では、**スマートフォン（iPhoneやAndroid）で動作するゲーム**を作ることを目標としています。そのための設定を行っていきましょう。

ビルドの設定

まずは、スマートフォン用にビルドするための設定を行いましょう。ツールバーからFile→Build Settingsを選択してください。Build Settingsウィンドウが開くので、Platform欄から「iOS（Android用にビルドする場合はAndroid）」を選択して、Switch Platformボタンをクリックします。

Fig.3-12 ビルド設定を変更する



このようにビルド先を「iOS」あるいは「Android」に指定することで、それぞれのスマートフォン向けにプロジェクトをビルドできるようになります。設定が終わったら、Build Settingsウィンドウは閉じておいて大丈夫です。

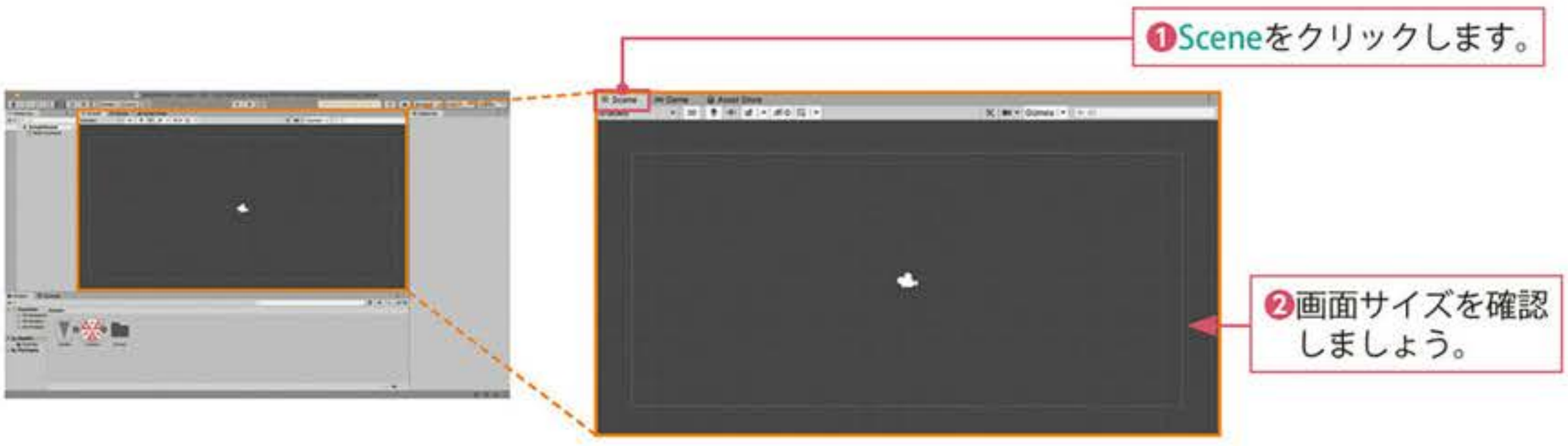
画面サイズの設定

続けて、ゲームの画面サイズを設定しましょう。Gameタブをクリックし、ゲームビュー左上にある画面サイズ設定（Aspect）のドロップダウンリストを開きます。**スマートフォンによって画面のサイズが異なるので、対象となるスマートフォンの画面サイズに合ったものを選択してください。**本書では、「iPhone X/XS 2436×1125 Landscape」を選択します。

Fig.3-13 画面サイズを設定する

Sceneタブをクリックしてシーンビューに戻り、画面サイズが変更されていることを確認します。シーンビュー内で、白の四角で表示されている範囲が、ゲーム画面の範囲です。

Fig.3-14 画面サイズを確認する

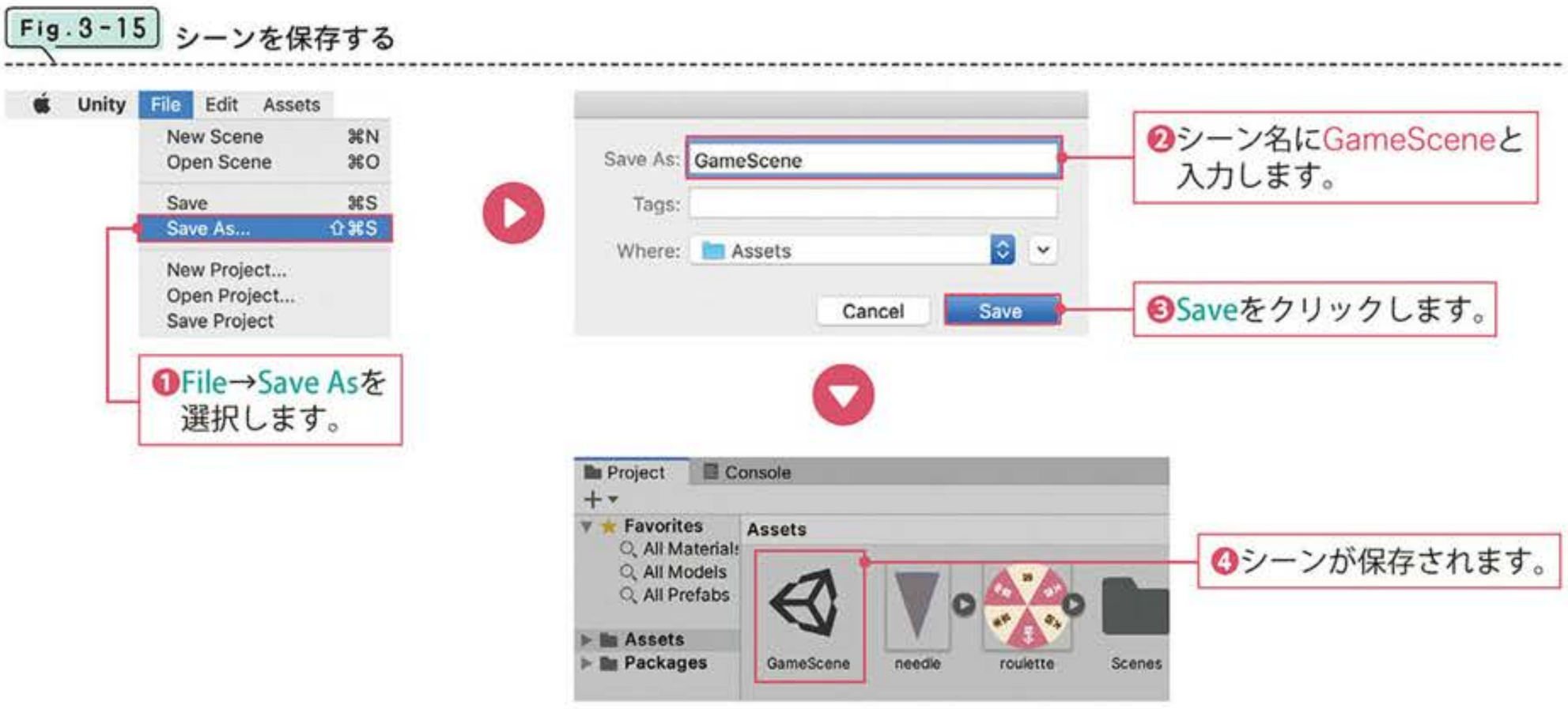


3-2-3 シーンを保存する

シーンを作成します。ツールバーからFile→Save Asを選択すると、シーン保存のウィンドウが表示されます。ここではSave As欄に「GameScene」と入力してSaveボタンをクリックしてください。プロジェクトウィンドウにUnityのアイコンが出現し、「GameScene」という名前でシーンが保存されます。

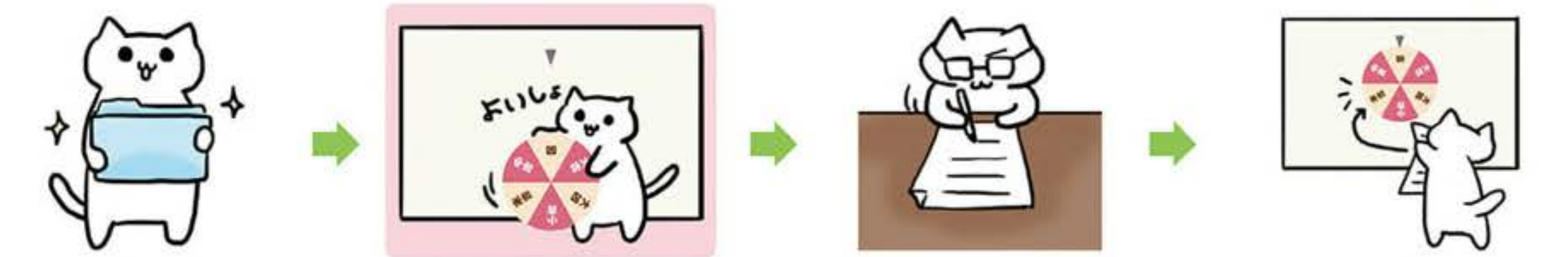
File→Saveメニューで作業中のシーンを保存することができます。ゲームの作成中は、随時セーブを行いながら作業を進めていきましょう。

Fig.3-15 シーンを保存する



ここまででゲームを作るための下準備が整いました。ついに次の節からゲーム本体の作成を開始します。お楽しみに！

3-3 シーンにオブジェクトを配置しよう



①プロジェクトの作成　②オブジェクトの配置　③スクリプトの作成　④スクリプトのアタッチ

3-3-1 ルーレットの配置

シーンビューにルーレットの画像を配置します。先ほどプロジェクトウィンドウに追加した「roulette」をシーンビューにドラッグ＆ドロップしてください。Unityの2Dゲーム用のプロジェクトでは、シーンビューに配置した画像はスプライトと呼ばれます。

シーンビューとヒエラルキーウィンドウのオブジェクトは一対一で対応しているため、ヒエラルキーウィンドウの一覧にも「roulette」と表示されます。

Fig.3-16 ルーレットをシーンに追加する

オブジェクトの位置を調節する

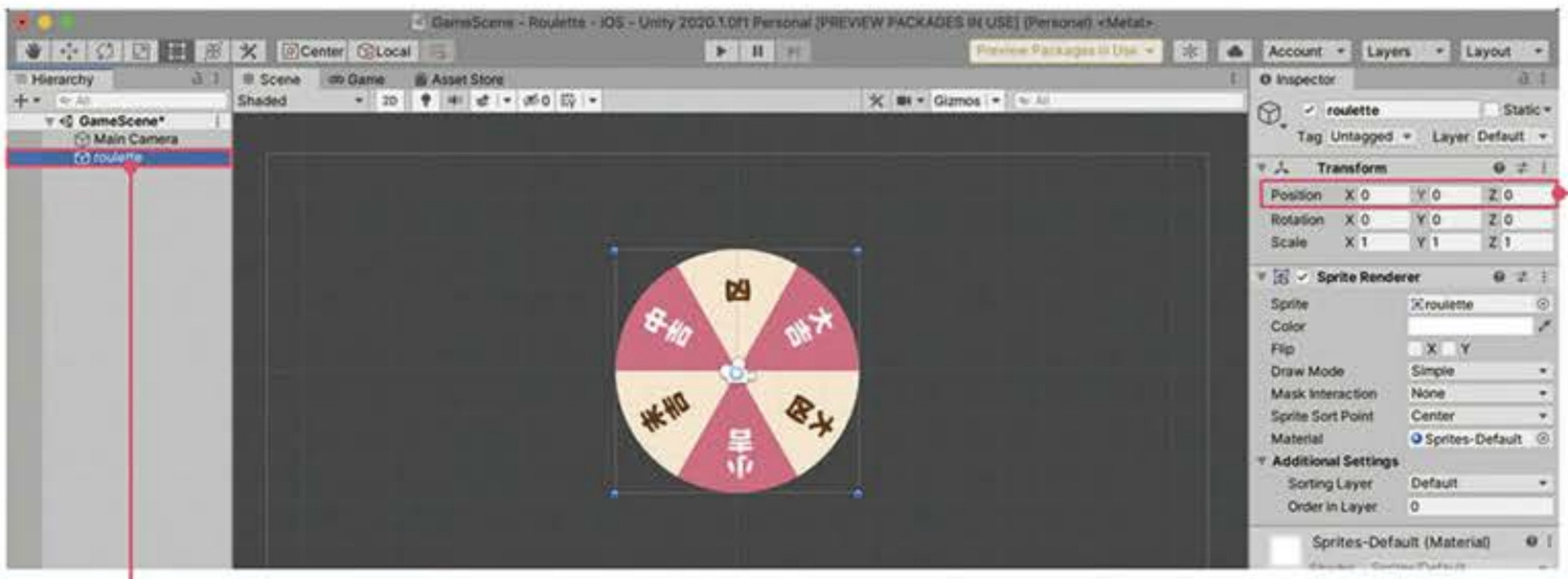
次にルーレットの位置を決めます。1章では画面上部の操作ツールを使ってオブジェクトを移動する方法を紹介しましたが、ここではインスペクターを使って座標を指定する方法を説明します(Fig.3-17)。インスペクターはオブジェクトの詳細情報を編集できるツールです。**座標を特定の位置に合わせたい時は、インスペクターを使う方が便利です。**

Fig.3-17 オブジェクトの移動方法

大まかな移動は操作ツール　　細かな移動はインスペクター

インスペクターを使ってルーレットの座標を指定してみましょう、まず、ヒエラルキーウィンドウでroulletteを選択してください。すると、ルーレットの詳細情報がUnityエディタ右側のインスペクターに表示されるので、Transform項目のPosition欄の「X」「Y」「Z」をそれぞれ「0」に設定します。

Fig.3-18 ルーレットを配置する



❶rouletteを選択します。

❷Positionを0, 0, 0に設定します。

これはルーレットのスプライトのX座標とY座標を、それぞれ「0」の位置に配置するという意味になります（今回は2Dのゲームですので、Z座標の数値は位置に影響しません）。**X座標とY座標を「0」にすることで、シーン（ゲーム画面）の中央にスプライトを配置することができます。**

Tips 座標の方向とカメラの位置

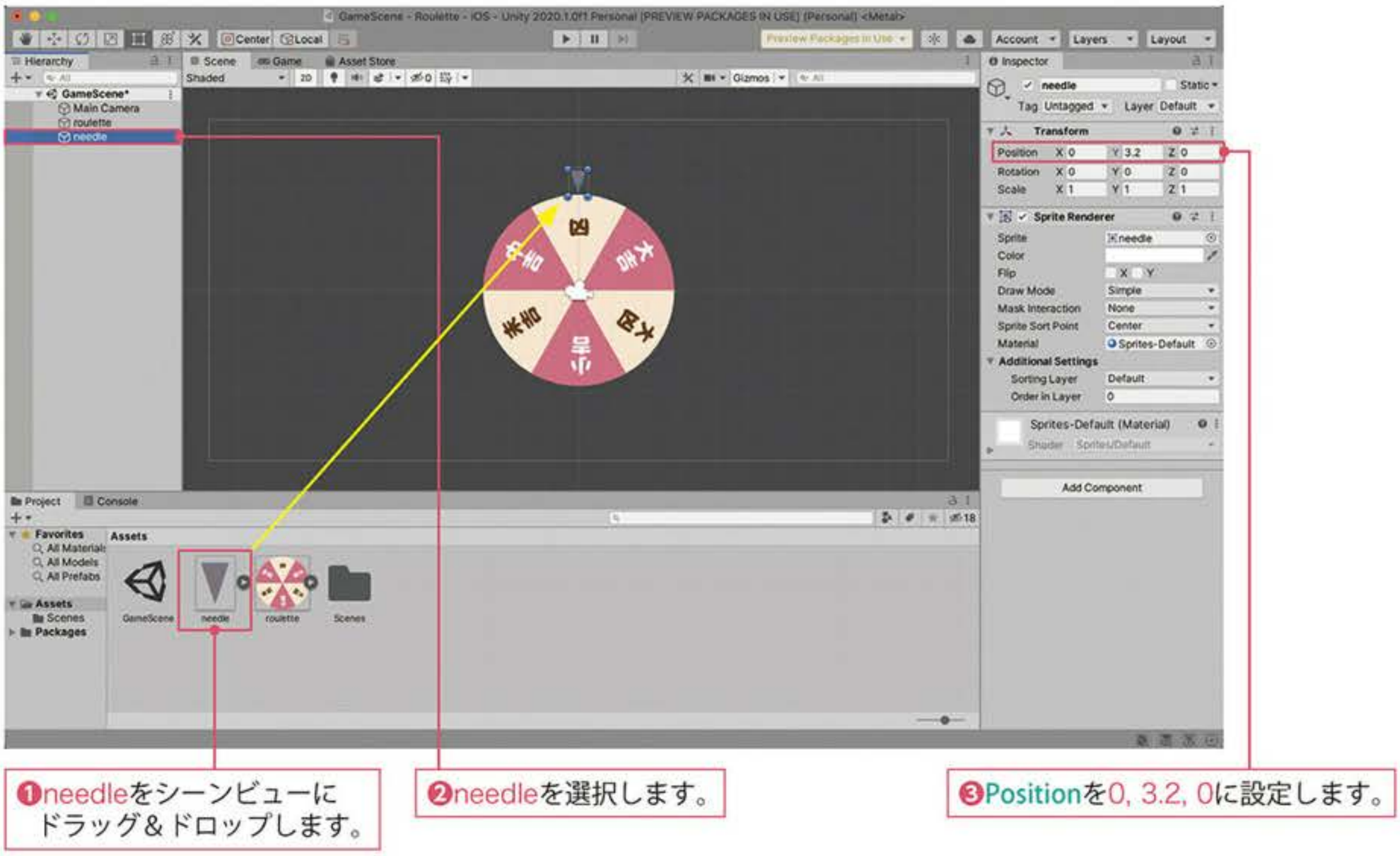
Unityの2Dプロジェクトでは、初期状態で画面の左右方向がX軸、上下方向がY軸、奥行き方向がZ軸になります。2Dゲームの場合、基本的にZ座標の値は「0」にしておきます。これは、シーンを撮影するカメラが「Z = -10」の位置に置かれているためです。もしスプライトのZ座標が「-10」より小さいとカメラに映らなくなってしまい、ゲーム画面にも表示されなくなってしまいます。

3-3-2 針の配置

針の画像をシーンビューに配置します。針の配置手順はルーレットの場合と同様です。針の画像をシーンビューにドラッグ＆ドロップした後に、インスペクターで座標を指定します。

プロジェクトウィンドウから、針の画像「needle」をシーンビューにドラッグ＆ドロップしてください。そして、ヒエラルキーウィンドウでneedleを選択し、インスペクターのTransform項目のPositionを「0, 3.2, 0」に設定します。

Fig.3-19 針を配置する

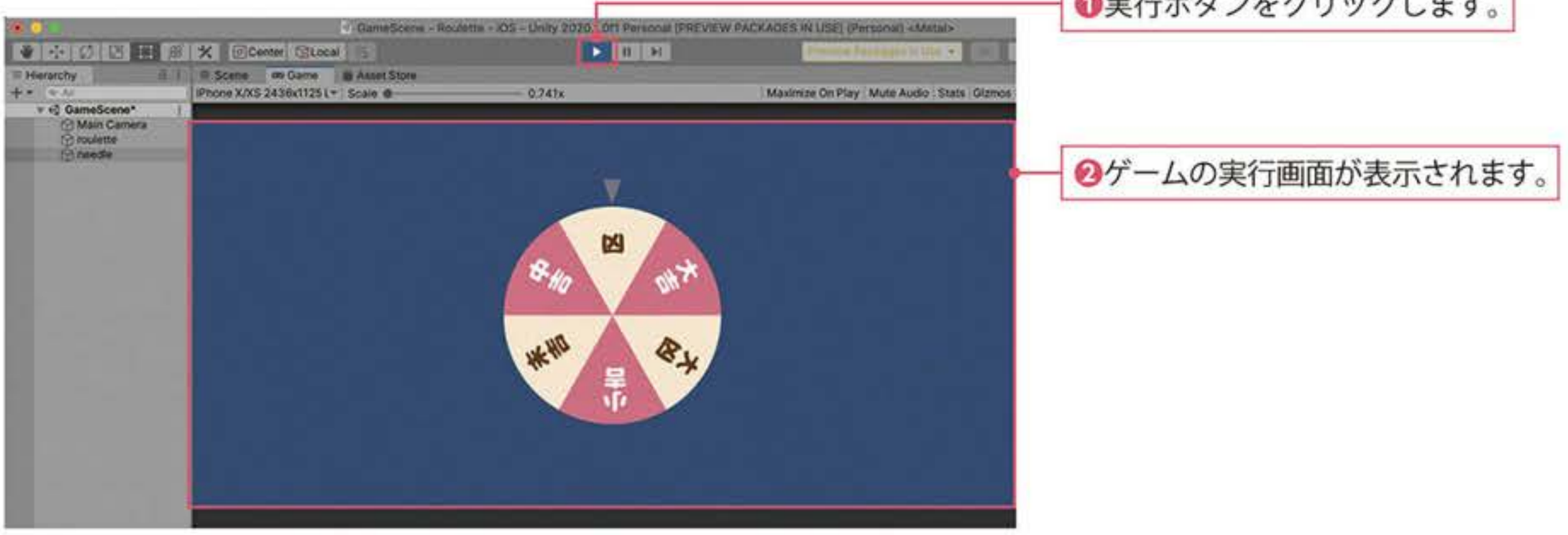


先に配置したルーレットのスプライトの上部分に、針のスプライトを配置することができました。

ここまでできたら、ゲームを実行してみましょう。Unityエディタ上部の実行ボタンをクリックしてください。Unityの画面がゲームビューに切り替わって、ゲーム画面が表示されます。ルーレットと針がちゃんと表示されていますね（Fig.3-20）。ルーレットが拡大して表示される場合は、ゲームビュー上部の「Scale」をスライドして見え方を調整してください。

ついにゲーム作りの第一歩を踏み出しました！ なお、ゲームを終了する場合にはもう一度、実行ボタンをクリックします。

Fig.3-20 ゲームを実行してみる



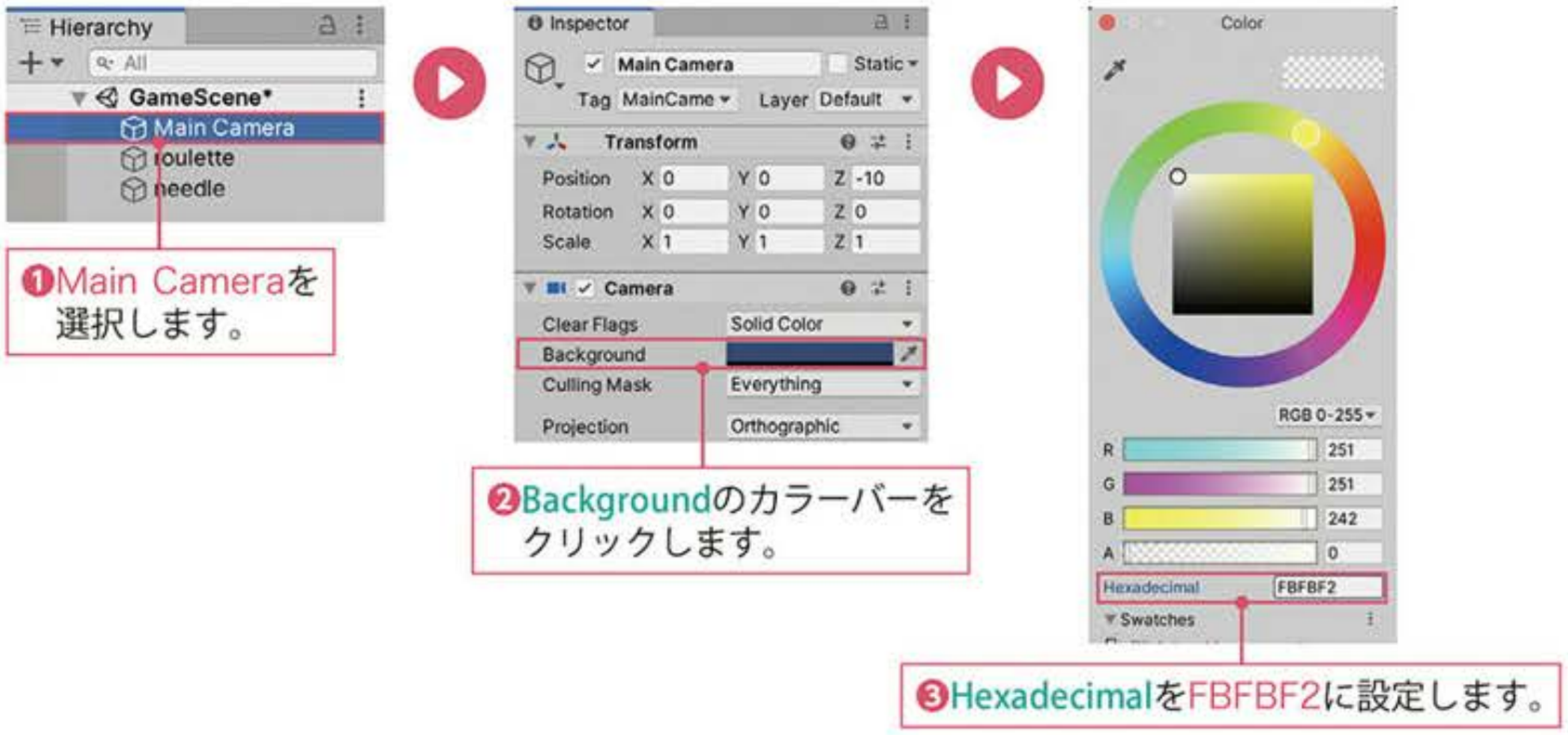
3-3-3 背景色を変更する

背景が青色だと少し印象が強すぎるので、背景色を淡い色にしましょう。**背景色を変えるには、カメラオブジェクトのパラメータを変更します。**

ヒエラルキーウィンドウでMain Cameraを選択し、インスペクターからCamera項目のBackgroundのカラーバーをクリックし、Colorウィンドウを表示します。ここではルーレットの色に合うように、Hexadecimalを「FBFBF2」にしました。

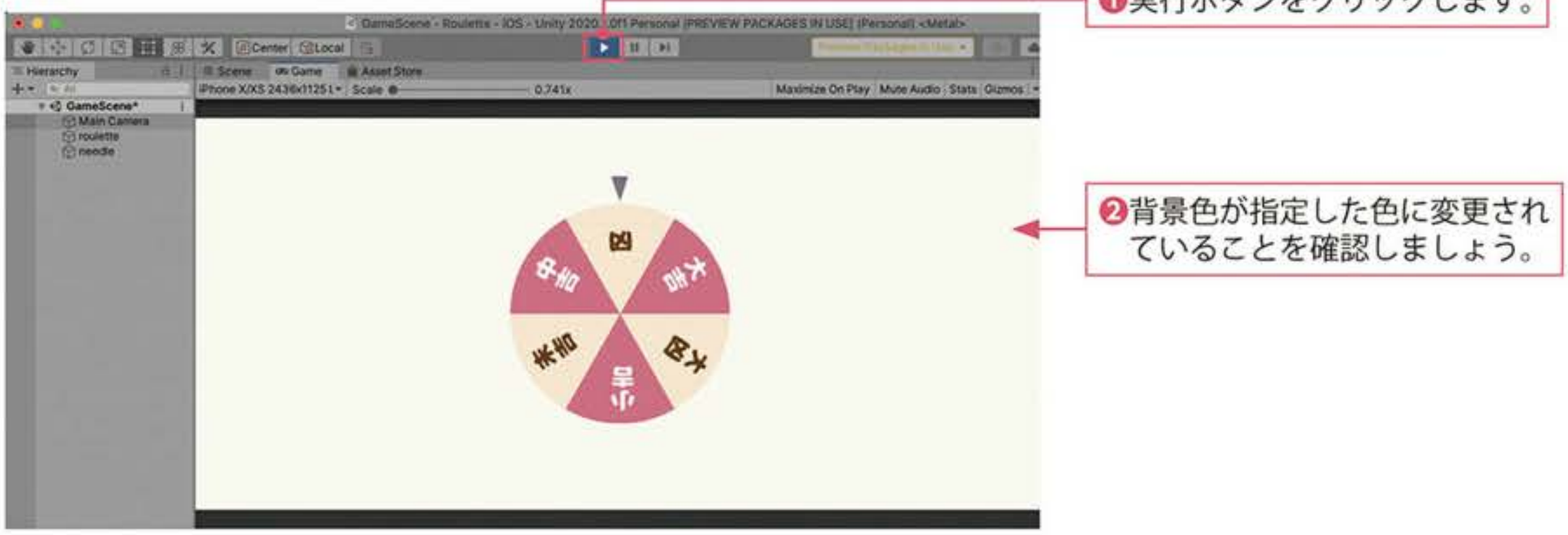
ここで「Hexadecimal」に指定した「FBFBF2」は、色を16進数の数値で表したカラーコードと呼ばれるものです。「000000」が黒、「FFFFFF」が白を表します。その他の色は、その中間の数値で表します。詳しくはWebなどで調べてみてください。

Fig.3-21 背景色を変更する



もう一度ゲームを実行してみて、背景色が指定した色になったことを確認しましょう。

Fig.3-22 背景色が変更されたことを確認する



インスペクターで色を選ぶだけで簡単に背景色を変えることができました！ このように、Unityを使えばスクリプトを書くことなく、さまざまな設定をエディタで視覚的に変更できるのです。

Unityで画像を表示するのは、思ったよりも簡単だったのではないでしょうか。一昔前まではスクリプトを数百行書いてようやく画像が表示されるという状態でした。それが、Unityを使えばスクリプトを1行も書くことなく、ゲーム画面に画像を表示できるのです。まさに夢のようなツールですね。Unityの使い方を覚えて面白いゲームをどんどん作っていきましょう！

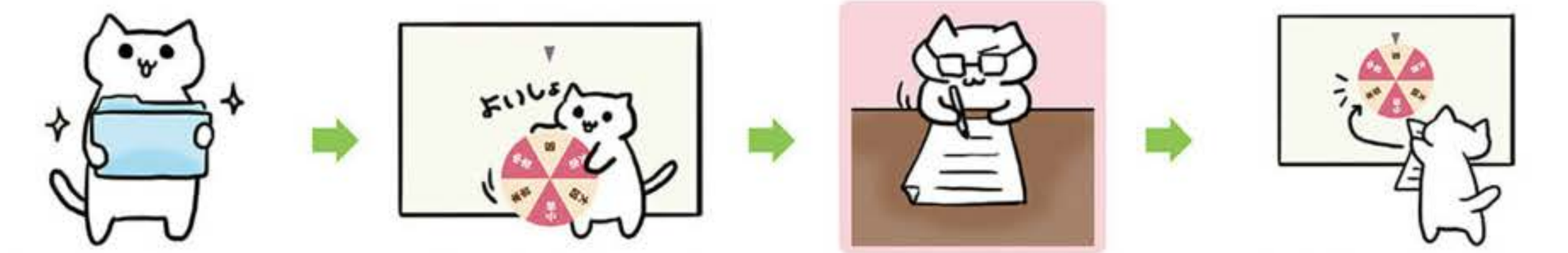
3-3節ではゲームで使う部品を配置しました。次の3-4節ではルーレットを回転させるために、コントローラスクリプトを作成します。

＞Tips＜ デザインには理由がある

「デザイン」と聞くと、「センス」と「直感」で作り出すもの、というイメージがあるようですが、そうではありません。「design」には「設計」という意味があることからもわかるように、デザインはエンジニアリングの分野です。よいデザインには「なぜこの色なのか？」「なぜこのレイアウトなのか？」など、1つひとつの要素に理由があるものです。

背景色ひとつとってみても、「なぜその色を使うのか？」を考えてから色を決めることで、最終的な見栄えが違ってきます。

3-4 スクリプトの作り方を学ぼう

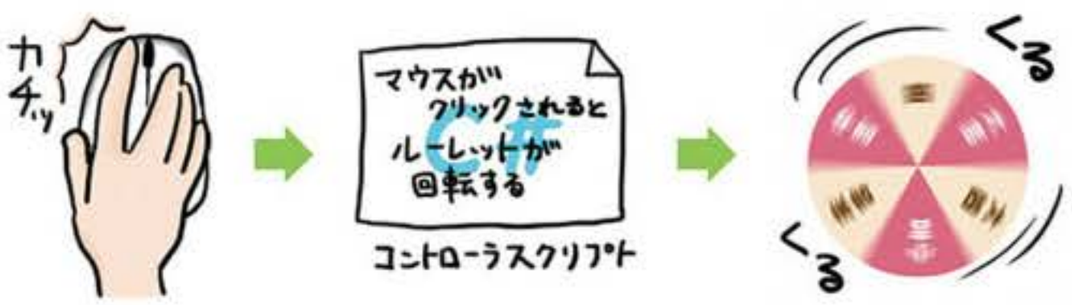


①プロジェクトの作成　②オブジェクトの配置　③スクリプトの作成　④スクリプトのアタッチ

3-4-1 スクリプトの役割

3-4節では**マウスのクリックに応じてルーレットを回転させ、減速して止まる仕組み**を作成します。オブジェクトを動かすためには、オブジェクトの動かし方を書いた「台本」が必要になります。Unityでは、この台本のことを「スクリプト」と呼ぶことは、既に2章で解説しましたね。この章ではルーレットを回すためのコントローラスクリプトを作っていきましょう。

Fig.3-23 台本を作る

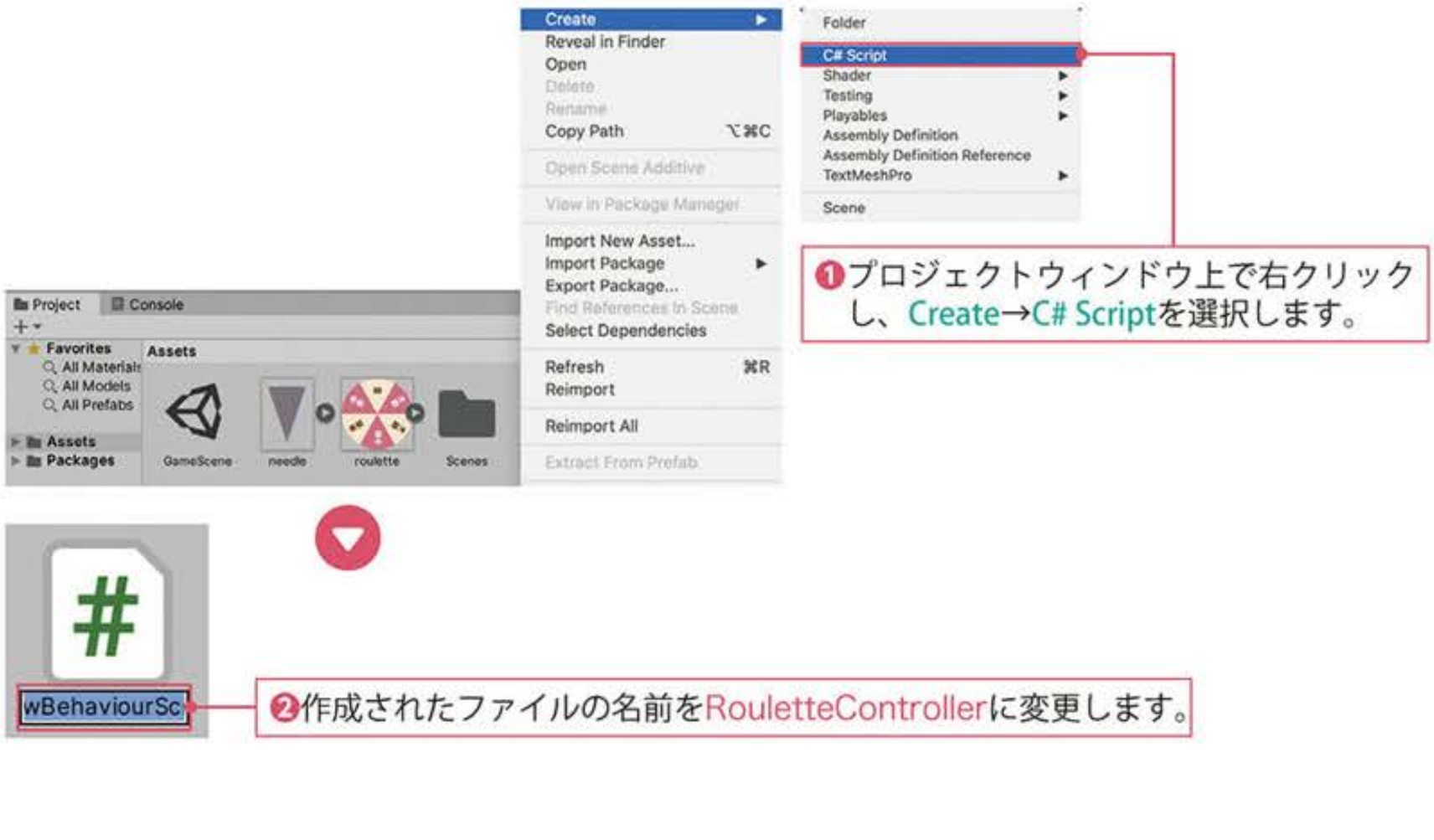


スクリプトを作成するにあたって、まずは「クリックすると一定の速さで回転する」スクリプトを考えます。減速して止まる部分は後で考えましょう。一見難しそうなものでも、**単純な動作に分解すると、思いのほか簡単に実現できるものです。**

3-4-2 ルーレットのスクリプトを作る

プロジェクトウィンドウ内で右クリックし、メニューからCreate→C# Scriptを選択します。ファイル作成直後はファイル名が編集状態になるので、確定する前にファイル名を「RouletteController」に変更して決定します。

Fig.3-24 スクリプトの新規作成



ファイルが作成できたら、ダブルクリックして開いてみましょう。Visual Studioが起動します。開いたファイルに、List 3-1のスクリプトを入力・保存してください。

List 3-1 「クリックすると一定の速さで回転する」スクリプト

```
1  using System.Collections;
2  using System.Collections.Generic;
3  using UnityEngine;
4
5  public class RouletteController : MonoBehaviour
6  {
7      float rotSpeed = 0;  // 回転速度
8
9      void Start()
10     {
11
12     }
13
14     void Update()
15     {
16         // マウスが押されたら回転速度を設定する
17         if (Input.GetMouseButtonDown(0))
18         {
19             this.rotSpeed = 10;
20         }
21
22         // 回転速度ぶん、ルーレットを回転させる
23         transform.Rotate(0, 0, this.rotSpeed);
24     }
25 }
```

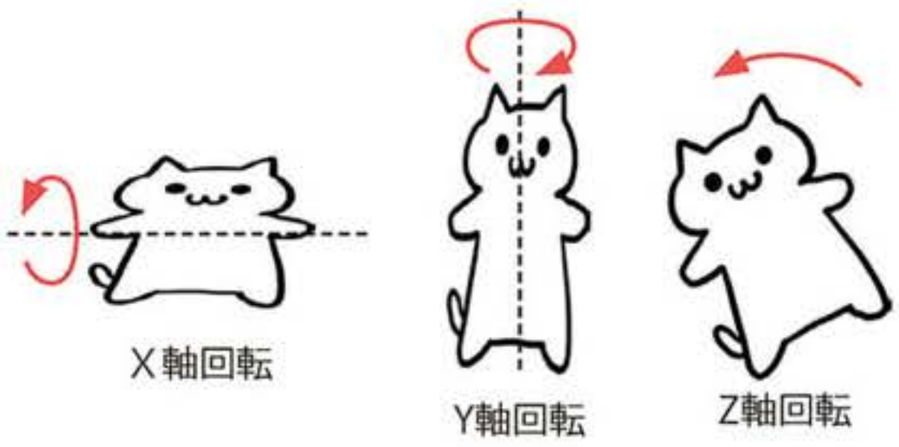
ルーレットコントローラでは、毎フレーム実行されるUpdateメソッドのなかで少しずつルーレットを回転させることで、回転アニメーションを実現しています（フレームについては60ページを確認してください）。

ルーレットを少しずつ回転させるには、Rotateメソッドを使います（23行目）。Rotateの前にある`transform.`については4章で説明するので、ここでは「**Rotateメソッドを使えば、オブジェクトを回転できる**」ことだけ覚えておいてください！

このRotateメソッドは、ゲームオブジェクトを**現在の角度から引数に与えた量だけ回転する**機能を持っています。Rotateメソッドに渡す引数は順番にX軸方向、Y軸方向、Z軸方向を中心とした回転量です。ここではZ軸（画面奥に向かう軸）を中心に回転させたいので第3引数に回転量を指定しています。

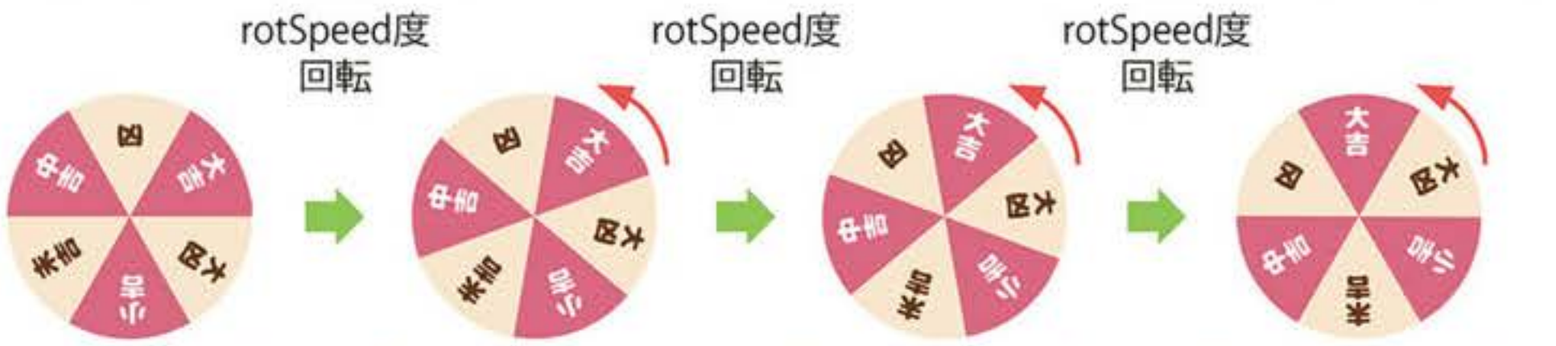
引数に与える回転量がプラスの場合には反時計回り、マイナスの数値を与えた場合は時計回りに回転します。

Fig.3-25 軸ごとの回転の向き



List 3-1では、ルーレットの回転速度をメンバ変数（rotSpeed）で定義しています。Updateメソッドのなかで`Rotate(0, 0, this.rotSpeed);`とすることで、フレームごとにrotSpeedで設定した角度ずつ回転します。

Fig.3-26 Rotateの回転量




```
Rotate(0, 0, this.rotSpeed);
Rotate(0, 0, this.rotSpeed);
Rotate(0, 0, this.rotSpeed);
Rotate(0, 0, this.rotSpeed);
```

マウスをクリックした時に回転し始めるように、**最初は変数「rotSpeed」の値を「0」にしておき（7行目）、マウスをクリックした時に回転量を「10」に設定**しています（17〜20行目）。

Fig.3-27 「rotSpeed」で回転速度を調整する

マウスがクリックされたことを検知するため、Input.GetMouseButtonDownメソッドを使っています（17行目）。このメソッドは、マウスがクリックされた瞬間に一度だけ「true」を返します（trueは「真」を意味する値です。64ページ）。引数が「0」なら左クリック、「1」なら右クリック、「2」なら中ボタンクリックを検知します。

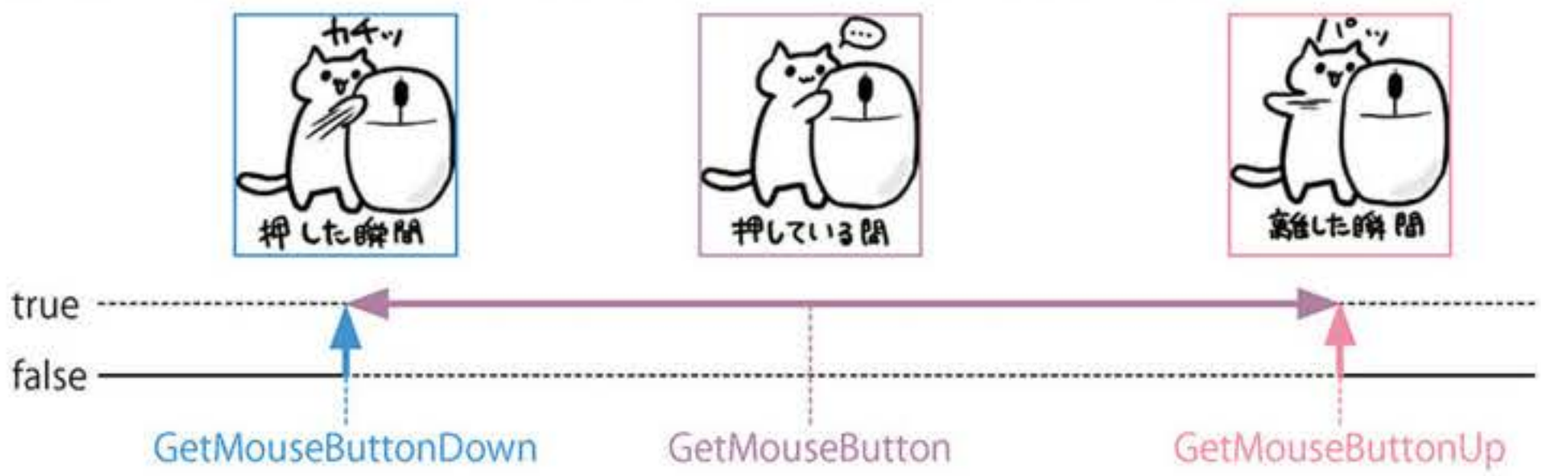
if文（77ページ）でInput.GetMouseButtonDownメソッドの返り値をチェックし、左クリックされた場合は、「rotSpeed」の値を「10」に設定しています。これにより、マウスがクリックされると、ルーレットは毎フレーム10度の速度で回転し続けるようになります。

以上で、スクリプトの作成と説明はおしまいです。書いたスクリプトでちゃんとルーレットが回転するのか不安だと思うので、さっそく実行してルーレットを動かしてみたいところです。でもルーレットを回転させるには、今書いたスクリプトをルーレットのオブジェクトに渡す（アタッチする）必要があります。この方法を次の3-5節で説明します。

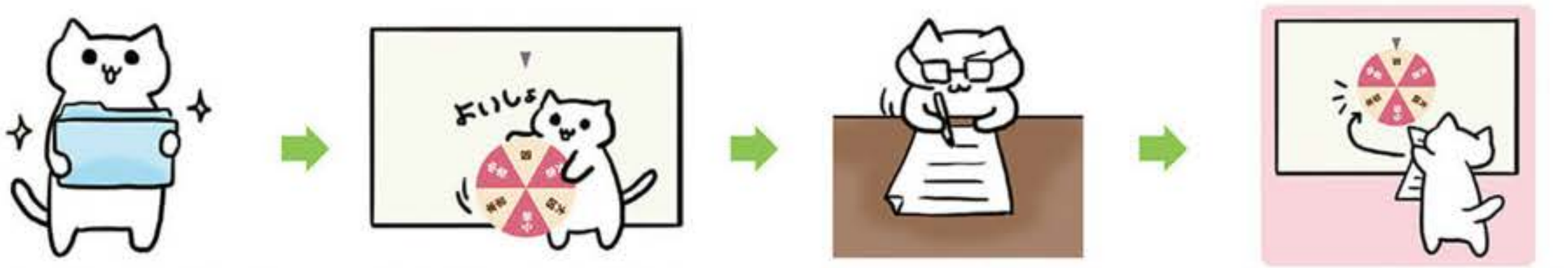
Tips マウスからの入力を取得しよう

マウスに関連するメソッドを2つ紹介しておきます。GetMouseButtonDownがマウスクリックされた瞬間にtrueを返すのに対して、GetMouseButtonUpメソッドはマウスボタンが離された瞬間に一度だけtrueを返します。また、GetMouseButtonメソッドはマウスがクリックされている間は、ずっとtrueを返します。

Fig.3-28 GetMouseButtonの働き



3-5 スクリプトをアタッチして ルーレットを回そう



①プロジェクトの作成　②オブジェクトの配置　③スクリプトの作成　④スクリプトのアタッチ

3-5-1 ルーレットにスクリプトをアタッチする

3-5節では、3-4節で作成したルーレットコントローラを、ルーレットのスプライトにアタッチします。**スクリプトをアタッチすることで、ルーレットはスクリプトの指示通りに動くようになります。**役者さんに台本を渡して、その台本通り動いてもらうイメージです。

Fig.3-29 役者に台本を渡す

スクリプトをルーレットにアタッチするには、Fig.3-30のようにプロジェクトウィンドウにあるスクリプト「RouletteController」を、ヒエラルキーウィンドウの「roulette」オブジェクトにドラッグ＆ドロップします。これで、ルーレットを動かすことができるようになります。

ルーレットのスプライトにコントローラスクリプトがアタッチできた（役者に台本が渡せた）ので、ゲームを実行してみましょう（Fig.3-31）。画面上でクリックするとルーレットが回転するはずです！

Fig.3-30 ルーレットにスクリプトをアタッチする

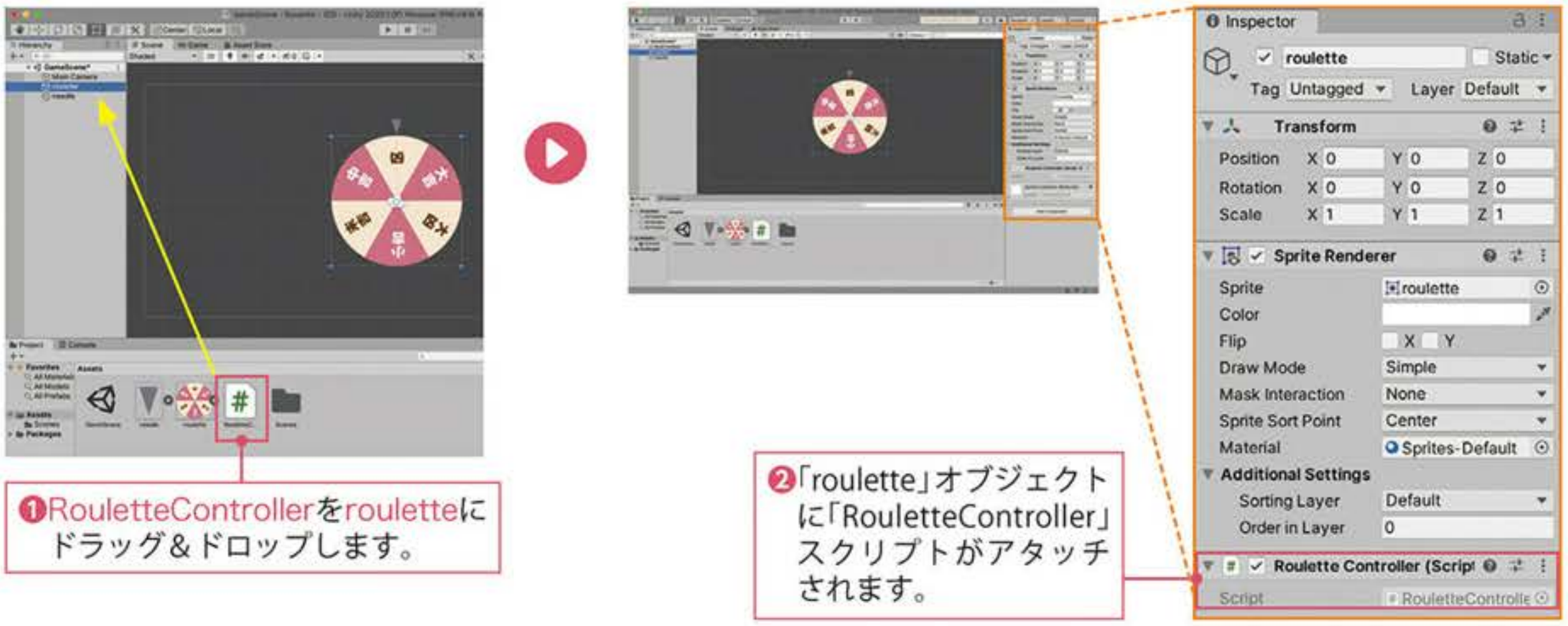
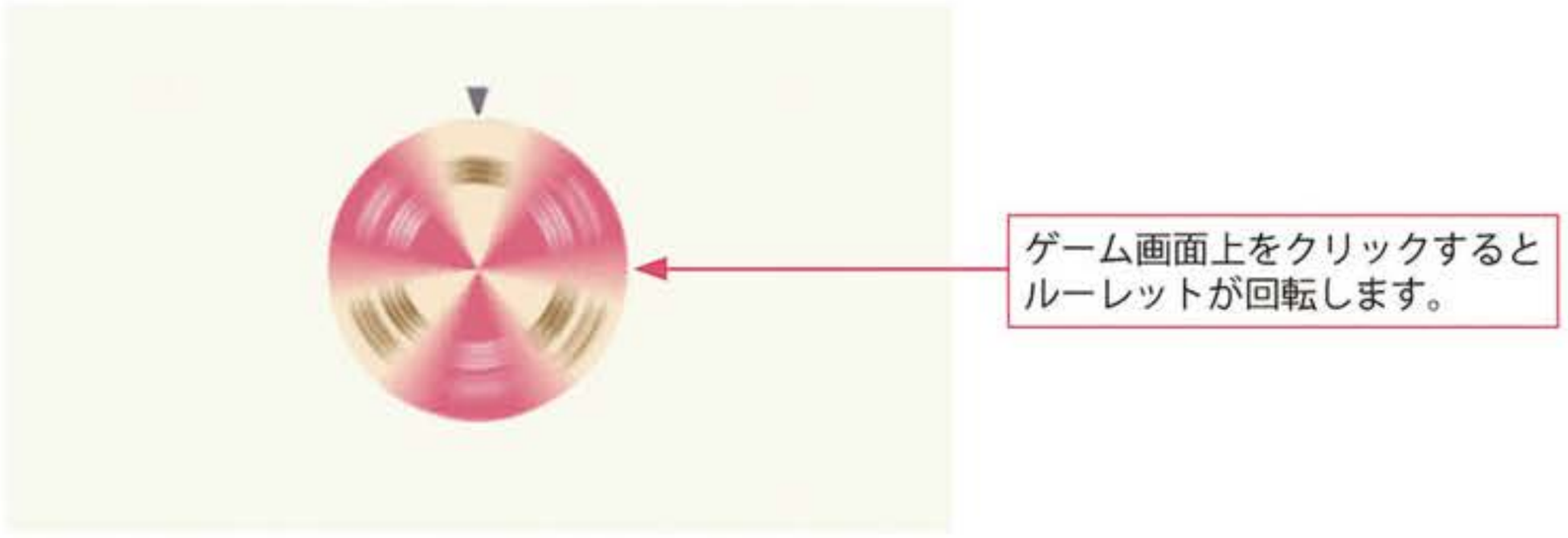


Fig.3-31 ルーレットが回転することを確認する



ここで、ルーレットの動かし方をおさらいしておきましょう。Unityで動くオブジェクトを作る時は、いつでも下記の方法を使うので、しっかりと復習しておいてください。

動くオブジェクトの作り方 重要!

❶シーンビューにオブジェクトを配置します。
❷オブジェクトの動かし方を書いたスクリプトを作成します。
❸作成したスクリプトをオブジェクトにアタッチします。

やってみよう!

ルーレットコントローラ（131ページのList 3-1）の19行目で、「this.rotSpeed」を「10」に設定している部分を「5」に変更してください。変更するとルーレットの回転速度が半分になることを確認してみましょう！

3-6 ルーレットの回転が止まるようにしよう

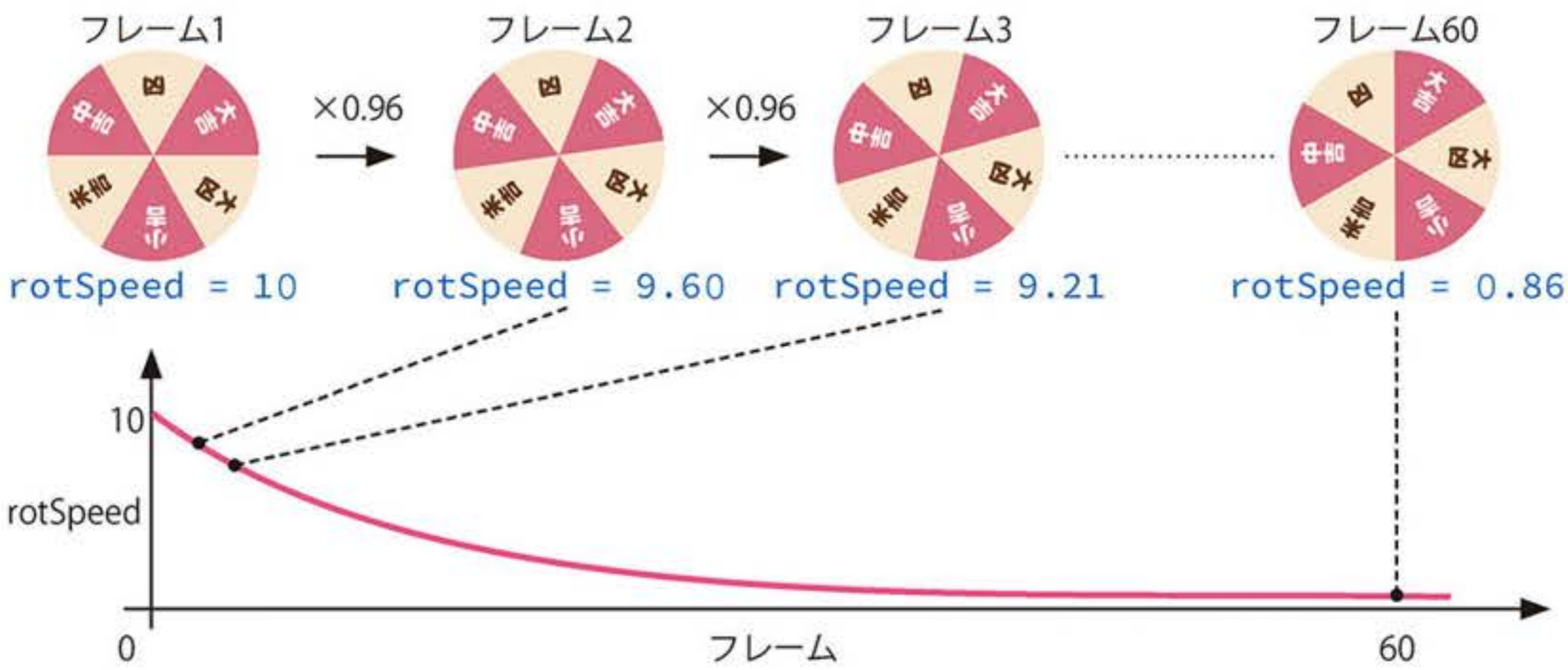
今のままでは、一度マウスをクリックしてルーレットを回転させると、止まることなく回り続けてしまいます。これでは占いになりませんね（笑）。そこで、だんだん回転が遅くなって、最後には停止するようにスクリプトを修正しましょう。

3-6-1 回転速度を遅くする方法を考える

ルーレットの回転速度をだんだん遅くするには、回転速度用のメンバ変数「rotSpeed」の値を少しずつ小さくしていけばよさそうです。ただ、**rotSpeedを少しずつ減算していくだけだと、一定の速さ（線形）で減速するため不自然な動きになってしまいます。**そこで、フレームごとにrotSpeedに減衰係数（例えば0.96）を掛けてみましょう。

この方法の場合、線形ではなく指数関数的に減速するので、自然に減速しているように見せることができます。**減衰係数を変更するだけで減速のスピードを簡単に変えられるので、空気抵抗での減速やバネ振動の減衰など、さまざまな場面で使われています。**

Fig.3-32 減衰係数を使って減速する



3-6-2 ルーレットのスクリプトを修正する

この方法をスクリプトに追記しましょう。プロジェクトウィンドウの「RouletteController」をダブルクリックして開いて、List 3-2のようにスクリプトを追加してください。

List 3-2 ルーレットを減速させる処理を追加する

```
1  using System.Collections;
2  using System.Collections.Generic;
3  using UnityEngine;
4
5  public class RouletteController : MonoBehaviour
6  {
7      float rotSpeed = 0;  // 回転速度
8
9      void Start()
10     {
11
12     }
13
14     void Update()
15     {
16         // マウスが押されたら回転速度を設定する
17         if (Input.GetMouseButtonDown(0))
18         {
19             this.rotSpeed = 10;
20         }
21
22         // 回転速度ぶん、ルーレットを回転させる
23         transform.Rotate(0, 0, this.rotSpeed);
24
25         // ルーレットを減速させる(追加)
26         this.rotSpeed *= 0.96f;
27     }
28 }
```

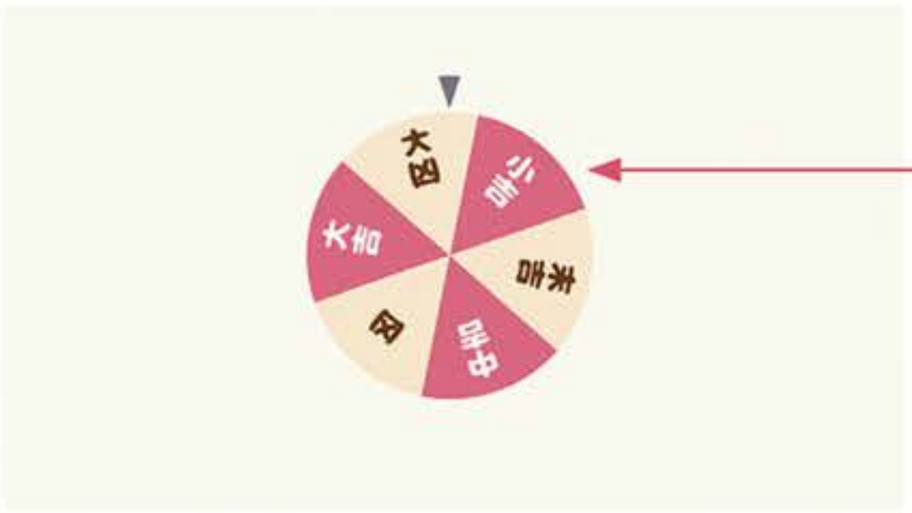
26行目にルーレットを減速させるための処理を追加しています。Updateメソッドのなかでで、rotSpeedに減衰係数(0.96)を掛けることで、Fig.3-32のように、繰り返しrotSpeedが0.96倍されることになります。このスクリプトでは、マウスクリック時にrotSpeedに「10」が代入されます(19行目)。その1フレーム後には0.96倍されて「9.60」、2フレーム後にはさらに0.96倍され「9.216」となり、回転速度が時間とともに減衰していきます。最終的にはrotSpeedは限りなく「0」に近づいていきます。rotSpeedが0になることはありませんが、非常に小さな数のため、見

た目上はルーレットが停止したように見えます。

ルーレットが回転している途中で画面をクリックすれば、再度rotSpeedに「10」が代入されるので、回転速度が最高速度に戻ります。

スクリプトを保存したら再度実行してみてください（先ほどルーレットにスクリプトをアタッチしたので、今回はアタッチの作業は不要です）。ルーレットの回転がだんだん遅くなって止まりましたね！ 1行追加しただけで、いきなり動きが本物っぽくなりました！

Fig.3-33 ルーレットが減速して止まる



やってみよう!

もし減速スピードが気に入らない場合は、減衰係数を変更して、気持ちよい止まり方になるように修正してみてください。簡単に修正＆実行できるのはUnityの強みです。気軽にどんどん修正しましょう！

Tips スクリプトがアタッチできない？

スクリプトにエラーがあると、エラーを消すまではゲームオブジェクトにアタッチできないようになっています。スクリプトをドラッグ＆ドロップしてもアタッチされない時は、Unityエディタの左下にエラーを示すメッセージが出ていないかを確認してみましょう。