秋学期個人研究報告書

自作ゲーム

「玉転がし」「ひよこアタック」について

経営学部マーケティング学科

旭貴朗ゼミナール２年

Oga

目次

1. はじめに 2

2. 「玉転がし」の概要 3

2.1 オブジェクトの作成 3

2.2 重力の追加 3

2.3 反発の設定 4

2.4 カメラ位置の調整 4

3. 「玉転がし」の工夫点 5

3.1 色の変更 5

3.2 シーソーの作成 5

3.3 箱の設置 6

4. 「ひよこアタック」の概要 7

4.1 オブジェクトの作成 7

4.2 オブジェクトの配置 7

4.3 スクリプトの追加 8

4.4 コンポーネントの追加 8

4.5 マテリアルの追加 8

5. 「ひよこアタック」の工夫点 9

5.1 大砲の力 9

5.2 オブジェクトが消えるまでの時間 9

6. 最後に 10

7. 参考文献 11

1. はじめに

昨今スマートフォンなどの機器の普及に伴い、いわゆる Nintendo Switch をはじめとするゲーム機器以外の媒体でもゲームが行われることも増えてきた。2017 年時点で Google Play ストアでは 360 万個、App Store では 210 万個のアプリケーションが公開されている。そんな中、「自分たちが実際にプレイしているゲームなどはどのようにして出来上がっているのか？」という疑問を持った。この疑問を少しでも解決するために、実際に自分自身で簡易的なゲームを作成することにした。

今日、世界にはゲームを作成するためのプログラミング言語やアプリケーションなどがさまざまな種類ある。今回はそのさまざまな種類のゲーム作成アプリケーション中から Unity を用いて作成した。

Unity を用いることには様々な理由がある。その中から 3 つの理由を挙げる。1 つ目の理由は「個人や年商１０万ドル以下の組織は無料で使用することが可能である」ということである。このようなゲーム作成ソフトウェアの中には有料であるものや、課金しないと十分な機能を得ることができないものなどもある。その中で Unity はユーザーの条件付きではあるが無料で十分な機能を利用することができる。プログラミング初心者としては手軽に始めることができる。

２つ目は「様々なプラットフォームに対応している」ということである。現在世界にはゲームを行うことのできる媒体が様々あることを記述したが、それぞれのプラットフォームに対応した形というものが存在する。それらの様々な形に容易に変換することができるのが Unity の特徴の一つである。この特徴のおかげで、様々種類のプラットフォームのユーザーにゲームを届けることが可能となる。

最後３つ目は「アクティブユーザの豊富さ」である。Unity は月に世界で１００万人以上のアクティブユーザーを誇る。アクティブユーザーが多いということは世界中(インターネット上)に情報が豊富だということを示す。そのため行き詰まった際などに参考にできる情報が大量にあるのが魅力だ。以上の 3 つの理由などから今回は Unity というアプリケーションを用いて２種類のゲームを作成した。

次からは、実際に自分が作成した「玉転がし」と「ひよこアタック」の２種類のゲーム（動画のようなもの）の中身や工夫した点についての解説を行っていく。

2.「玉転がし」の概要

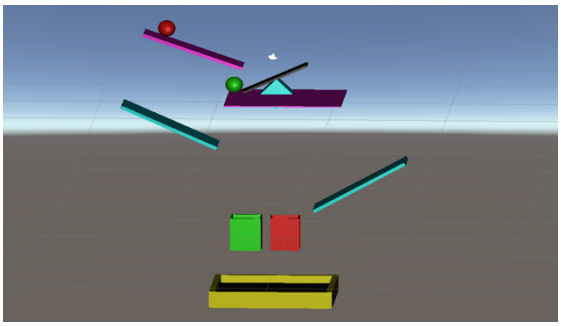

図　1　「玉転がし」の画面

図 2　Hierarchy Window

このゲームはゲームというよりも動画に近いものになる。内容としては、赤い玉と緑色の玉が坂道を下っていき、最終的に四角い筒を通って下に落下するというものである。(図 1 参照) 途中でシーソーのような仕掛けを作ってあるがコレに関しては次の章の工夫点のところで詳しく述べる。

全部で 25 個の Game Object で構成されている。(図 2 参照)　コレらの Game Object　は名前をつけて Hierarchy Window に表示されている。ここの中に、Game View の視点になる Main Camera や光源となる　Directional Light もこちらに含まれる。次から今回のプログラムの制作過程について説明する。

2.1 オブジェクトの作成

まず、床や坂とボールのオブジェクトの作成を行なった。床や坂は 3D Object の Cube を用いることで作成し、ボールは同じく 3D Object の Sphere を用いて作成した。配置や大きさに特にこだわりを持って作ったわけではないが、実際にテストを繰り返して自分の中で一番納得のいく位置に配置した。

2.2 重力の追加

このままだと、プログラムを開始してもボールはその場に止まったままで動画としての意味をなさない。このままでは困るので、重力を働かせるように設定していく。そこで使用するのが Component というものである。この Component はオブジェクトごとに設定することができるものであり、重力だけでなく摩擦や接触判断など他の要素を追加することもできる。Unity ではオブジェクトを追加した際にどの形状に合わせた Collider がアタッチされているがこれも Component の一種だ。なかなか機会はないと思うが最初にアタッチされている Collider から別のものに変更したい場

合は Component の追加によって行うことができる。今回は重力が働くようにするようにしていく。

重力を設定する Component は Rigidbody と呼ばれるものである。この Component は一覧の中の Physics の項目の中に用意されている。この Component を追加するとこのような画面が Inspector Window に追加される。(図 3 参照) Use Gravity にチェックを入れることで重力を有効化することができる。重力に関する細かい設定として、Mass で質量を設定することや、Drag で空気抵抗を設定することができる。

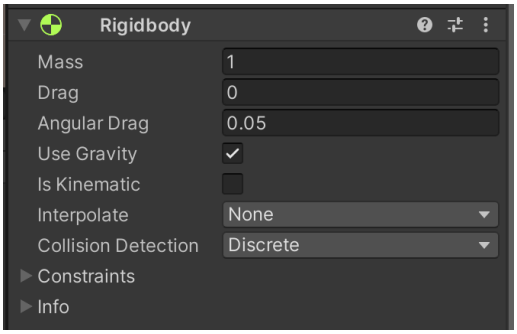

図 3　追加された画面

2.3 反発の設定

ここまでで殆ど動画としては完成に等しいところまで来ることができる。しかし、この状態ではオブジェクト同士が当たった時に跳ね返らずに力吸収されている。現実世界では起こり得ないことだ。これを改善することでより、現実世界で起きている状態に近いものにしていく。そのために Physics Material という Assets を用いて摩擦や反発を設定していく。Physics Material の項目は次の図のようになる。(図 4 参照) 上から、「動的摩擦」「静的摩擦」「反発力」を示す値になっている。この値はボールがコースからはみ出さないための要因にもなるので、テストを繰り返しながら調整する必要がある。

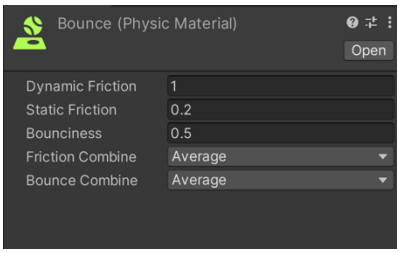

図 4　Physics Material

2.4 カメラ位置の調整

最後に Game View の視点を調整するため、Main Camera の位置を調整していく。次の章で説明するが、シーソーを作る際に立方体を埋め込んで四角錐のように見せようとしている。そのため高さと角度を調整することで立方体であることがわからないように調整した。実際にカメラの位置でどのように見えるかは以下の図の通りだ。(図 5 と図 6　参照)　これでプログラムのベースは完成である。

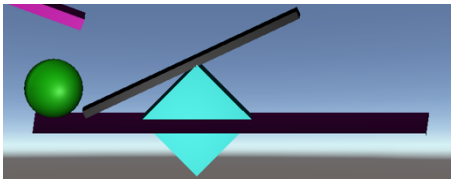

図 5　正面からの図

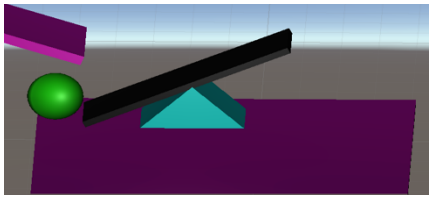

図 6　上からの図

3.「玉転がし」の工夫点

このゲームは大方参考文献にも記載してあるが、「Unity 2020 入門」という本に記載してあるプログラムに則ってして作成した。その中でも、自身で工夫した部分がいくつかあるので紹介する。工夫した点は 3 つあり、「色」「シーソー」「箱」である。

3.1 色の変更

まず、1 つ目の色について説明する。参考文献ではボールに赤色をつけているだけであるが、自分のプログラムでは 6 色を用いてボールと床に色をつけた。Material という Asset を作成して、それをオブジェクトに設定することで色をつけている。具体的な方法をここから説明していく。Project Window から Asset 追加のメニューを開き Material を選択する。次に Inspector Window にある Albedo という項目で具体的な色を定めていく。Albedo を開くと次のような画面が開く。(図 7 参照)この画面で RGB の値を変更して自分の好きな色を設定していく。この画面では Metallic の値を変更させることで素材の金属感を出すことも可能だ。こちらの機能は今回のプログラミングでは活用していない。

ここまでで作成した Material をオブジェクトに設定していく。Project Window に表示されている Assets を Hierarchy Window にある設定したいオブジェクトにドラッグ&ドロップすることで色をつけることができる。

3.2 シーソーの作成

今回、自分のプログラミングでは参考文献には載っていない仕掛けとして「シーソー」を作成した。3D Object の Cube を 2 つ用いることで今回は作成した。本来は四角錐を用いて作成したかったのだが、作成方法がわからなかった為、立方体を用いた。次に具体的な作成手順を説明する。

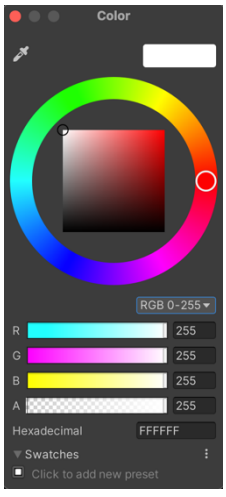

図 7　Albedo の画面

まず、シーソーの板となる部分を作成した。これは特別難しいことはしていない。ボールの直径を 1.5 の値に設定していたので、落ちないように幅を少し大きめの 2.0 で設定した。

次に支点となる部分の作成である。先ほど記述したように、錐の作成方法がわからなかった。その為、今回は立方体を床に埋め込むことで四角錐にする形をとった。(次頁の図 8 参照) 立方体の大きさは板の幅に合わせて一辺 2 の値で作成した。

最後に、ここまでに作成した 2 つのオブジェクトをシーソーの機能を果たすように配置する。まず、支点となる立方体を四角錐のような形にするために回転させた。綺麗な錐体にするために、45 度回転させ、半分床に埋まるように配置した。板は緑のボールをシーソーの動きによって押し出す形にしたかった為、静止時は接さないが、稼働時に接するギリギリのところで調整を行なった。(図 9 参照)

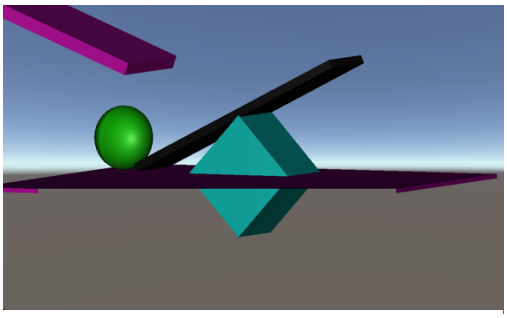

図 8 作成したシーソー

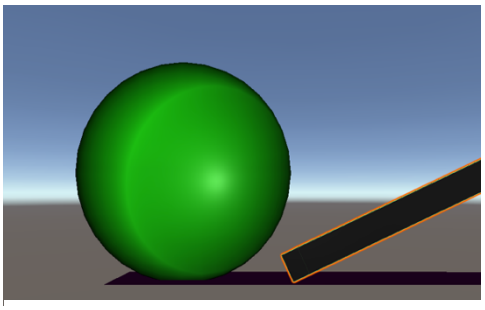

図 9　ボールと板の位置関係

3.3 箱の設置

この箱も、参考文献には載っていないが、自分自身で追加したものである。箱の機能としてはボールが坂を全て下っていって、最後の床に落下する前にそれぞれの色の箱を通過するというものである。当初は 2 つのボールを同じ箱に通すことも検討したが、最終的にはそれぞれ異なる箱を通過するように作成した。

次に作る工程を説明する。この箱は 3D Object の Cube を 4 つ組み合わせることで作成している。箱の大きさは特にこだわりはないが、小さすぎると存在感がないため比較的大きめに作成した。穴の大きさはボールの直径 1.5 より多少大きめの 2~2.5 で作成してある。箱とボールのサイズをピッタリにした方が、綺麗になったのかも知れないが跳ね返りなどの関係で微調整が難しかった。最後に箱の位置を調整して完成した。(図 10 参照)

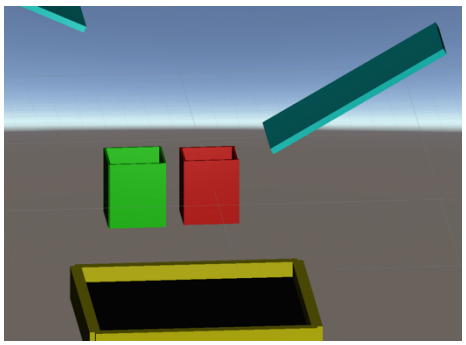

図 10　設置した箱

4.「ひよこアタック」の概要

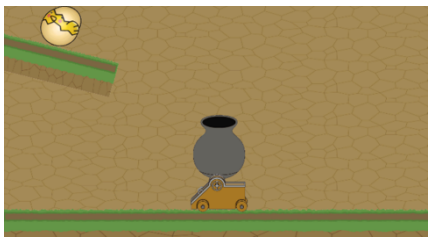

図 11　ひよこアタックの全体図

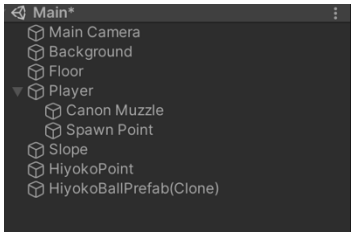

図 12　Hierarchy Window

先程までの「玉転がし」と異なり、こちらの「ひよこアタック」は動画というよりもよりゲームに近づけることができたものである。坂を下ってきたひよこをターゲットにして、大砲を操作して砲弾を当てるというものだ。そうは言っても、スコア表示をしたり、クリアがあったりするものではないのでゲームと呼ぶにはまだまだ遠いものだとは思う。

このプログラムは 9 個の常駐のオブジェクトと 2 種類の Prefab で構成されており、Hierarchy Window は上記の図のようになっている。(図 12 参照) この図を見ると Canon Muzzle と Spawn Point の 2 つが 1 段下がったところに表示されていることがわかること思う。これは、この 2 つのオブジェクトが Player というオブジェクトに対して親子関係に属しているからである。

4.1 オブジェクトの作成

今回のこのゲームでは、自分自身でオブジェクトは作成しておらず、参考文献の書籍がサイトにて提供しているサンプル素材から借りて作成している。素材のデータは様々な素材が 1 シートになっていたため、分割する必要があった。その為 Sprite Editor を使用した。この Editor では Automatic を選択することによってコンピュータが自動的に境界部分を判別して切り分けてくれる。これをすることによって、1 枚のシートから 6 個の素材にすることができた。

4.2 オブジェクトの配置

配置は、基本的に参考文献に載っているものと変わりなく配置した。その過程で、前述した親子化という工程があった。この親子化をすることによって子関係のオブジェクトが親オブジェクトの動きに追従させることや、位置を親オブジェクト基準に設定することができるようになる。今回は前回と異なり 2D のため普通の配置作業に Order in Layer という項目が増える。これはオブジェクトの重なり順を指

定するためのものであり。これを指定しない場合、開くたびに順番が変化して思い通りの表示にならない可能性があるので注意が必要だ。

4.3 スクリプトの追加

このプログラムは「玉転がし」とは異なり、プレイヤーが自分で操作する必要がある。そのため C#というプログラミング言語を用いたスクリプトを設定することで、操作可能にする必要がある。自分の手でプログラミングを描いて設定することもできるのだが、複雑かつ難しいため参考文献の書籍のサイトからダウンロードして利用した。先程、プレーヤーが操作できるようにスクリプトを用いると書いたが、それ以外でも「オブジェクトを一定時間で消去する」や「一定間隔ごとにひよこ玉を生成する」などの自動的に行われて欲しいものもスクリプトを用いることで指定することができる。

4.4 コンポーネントの追加

今回も前回と同様に、このままでは世界の物理法則に反するような動きをしてしまう。そのためコンポーネントを用いて重力の設定などをする必要がある。前回と異なる部分なのは、2D のプログラム専用のコンポーネントを用いないといけないところである。中身は 3D のものと殆ど変わりがないので説明は省略する。ひよこは楕円形をしているが、それに完全に一致する Collider はないため普通の円形の Collider を用いて設定している。(図 15 参照)

図 13　ひよこへの Collider

4.5 マテリアルの追加

現在の状態だとひよこ玉は反発せずに吸収されてしまう。これでは挙動に対する違和感がすごいため、反発するように Material を追加する。Physics Material 2D を追加すると Inspector Window に以下のような画像が追加される。(図 14 参照) Friction は摩擦係数を、Bounciness は反発力を表している。この値を調整することでひよこを反発させることができる。摩擦力は参考文献と同等だが、反発力は 1 だと跳ねすぎたため、0.2 にしてある。

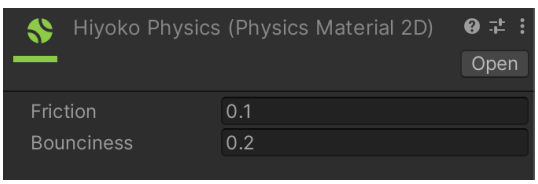

図 14　表示される Inspector Window

5.「ひよこアタック」の工夫点

このゲームは前述した通り、スクリプトなどを用いて作成したため複雑な構成になっており、「玉転がし」のように自分自身の工夫を行うことが難しかった。そのためしっかりとした工夫はできていないが、その中でも多少の工夫を行なった。「大砲の力」「オブジェクトが消えるまでの時間」の 2 つである。

5.1 大砲の力

これはスクリプト内で設定することができる値であるのだが、オブジェクトに一度アタッチするとこの値は、Unity 内でも変更することができるようになる。(図 14 参照)参考文献ではパワーの値は 1000 に設定されているが、自分のプログラミングでは 800 にした。最初の段階では自分も 1000 にしてはいたのだが、実際にゲームの動作を確認するとひよこが飛びすぎて画面外からなかなか帰ってこないことが多かった。弱くしすぎると当たってもひよこがあまり飛ばないのでそこの調節が難しかったが、テストを繰り返し自分の中で納得のいく力に調整した。

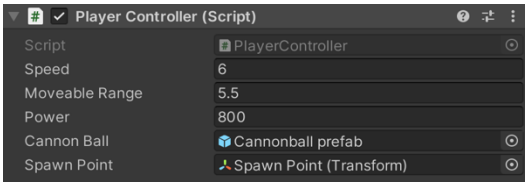

図 15　調整できる画面

5.2 オブジェクトが消えるまでの時間

これは、大砲やひよこがゲーム上に発生してから消滅するまでの時間を調整したということである。この値は、先程の大砲の力と同様に本来、スクリプト内で調整する値なのだが、アタッチすることによって Unity の Inspector Window にて調整することができる。(図 15 参照)大砲の消滅までの時間は特に変更させていないが、ひよこの消滅までの時間を自分自身で調節した。参考文献ではひよこ消滅までの時間の値は 10 に設定されているが、これではひよこが溜まりすぎて画面上に一杯になってしまうことがあった。その為、値を 6 に設定した。大砲の力の際と同様にテストを繰り返しながら自分の中での最適な値を設定した。

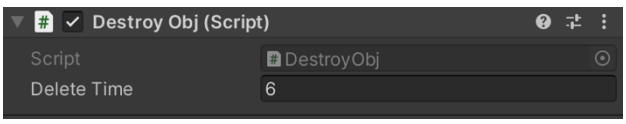

図 16　Inspector Window

6. 最後に

今回はUnityを使って作成した2種類のプログラミングについての説明をしてきた。どちらも基礎的なものであるので作成自体は難しくなかったように感じた。ただ、まだまだゲームとしては不十分な部分が様々ある。例えば、オープニング画面がないことや、得点のカウントなどがないことなどが挙げられる。これらのことは、今回作成する際にも調べることなどをしたが、難しかったため断念した。次回ゲームを作る機会があったら、そのようなところにも挑戦をしてみたいなと思う。

ゲーム作成を通して 2 つ学んだことがある。1つ目は「ゲームの基盤を作ることはできても、操作可能な状態にすることなどが難しい」ということである。実際作成を開始した時には、ステージを作成することが難しくなかったので、工夫などを行うことも簡単なのだろうと思っていた。しかし、ステージを作成したのちにプログラミング言語を用いてスクリプトを作成する段階になったときに壁に当たってしまった。工夫をしようとネットや本をもとに情報を集めても、なかなか思い通りに動かないことが多かった。ここの壁を乗り越えるためにはもう少し専門的に学んだり情報を集めたりする必要があるのかなと思う。

2つ目は1つ目と矛盾しているようなことになってしまうが、「案外簡単に作成することができる」ということだ。ゲームというとハードルが高いと感じていたが、Unityというソフトウェアを用いることで簡易的なものではあるが作成することができた。先程書いた通り、複雑なものは難しいが単純なんものであれば簡単に作成できる。だから、今までゲームの作成などに対してハードルの高さを感じている人にも挑戦をしてもらいたいなと思った。

今回ゲームの作成を通して、ゲームの複雑さと手軽さという正反対の 2 点を感じることができた。それによって日ごろ行なっているスマホや Nintendo Switch などの複雑な動きを有するゲームがいかに様々なプログラミングによって構成されているかということを知ることができた。今後、ゲームを行う際にはそのようなことについても多少なりとも意識しながらプレイすることになっていくのではないかと思っている。

7. 参考文献

荒川巧也・浅野祐一(2020)『Unity 2020 入門』　SB クリエイティブ株式会社

SB Creative ホームページ(こちらでスクリプトや素材を借りました)
https://www.sbcr.jp/product/4815606640/

Unity User Manual (2020-01-22 最終アクセス)
https://docs.unity3d.com/ja/2020.1/Manual/UnityManual.html