Unityを使用したリズムゲームソフト制作

川下　直輝・川上　真心
久保　友人・簗澤　輝彦

１．研究概要

Unity を使用して実際にリズムゲームを開発し，どこまで実際に開発されているゲーム(Osu Mania など)に近づけることができるのかを試すとともに，開発時に苦労する点を知る。

２．研究の具体的内容

(1)使用機材・ソフトウェア

- Unity(ゲームエンジン)
- Mono develop(IDE)
- Visual Studio 2017(IDE)
- SONER LE (DAW)
- Clip Studio Paint Pro(ペイント)
- Live2D Cubism Editor 3.0

(2)使用機材・ソフトウェアの説明

(a)Unity(ゲームエンジン)

2D,3D ゲームを開発できるゲームエンジンで，開発環境や実行できる環境を提供してくれるプラットフォームである。無料バージョンでも十分利用できるのでお財布にも優しい(図１)。

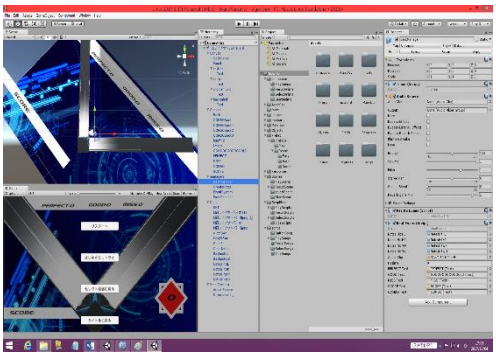

図１ Unity

(b)Mono develop (IDE)

Unity が無料で提供する統合開発環境（IDE）の１つである。Visual Studio 2017 よりも立ち上がりが早く，動作も軽いのでこちらを中心に，プログラミングを行った(図２)。

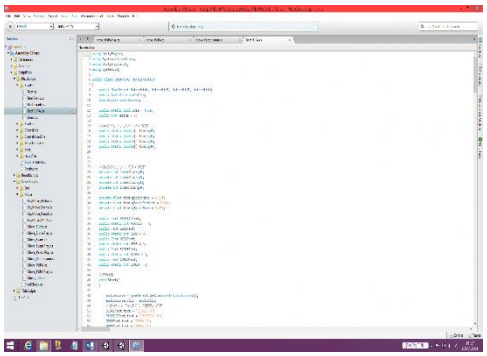

図２ Mono develop

(c)Visual Studio 2017 (IDE)

Mono Developと同様にUnityのプログラムを記述する際に使用する統合開発環境（IDE）の１つである。

Mono Developよりもコード補完が優秀でビルド時のエラー文も分かりやすく，開発を始めた当初はこちらのIDEを使用した。しかし，スクリプトが増え，コードを徐々に長くすると，Visual Studio 2017の起動に時間がかかるとともに，余計にコードを補完されてしまうので，今回のゲーム開発にはMono Developを中心に使用した。

Unity2017でVisual Studio 2015を使用するとビルドができないことがあるのでUnity2017を使用すると

きはVisual Studio 2017を使用するようにしたほうが良い(図３)。

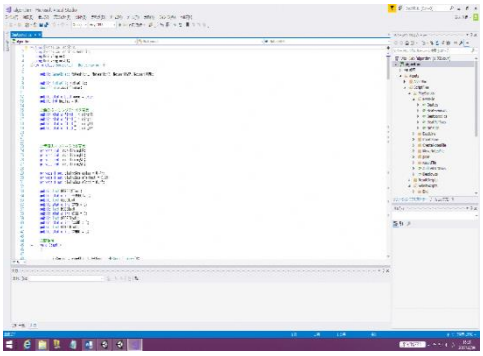

図３ Visual Studio 2017

(d)SONER LE(DAW)

リズムゲームを開発するにあたって必要になってくる譜面データを制作するためにこのソフトを使用した。DAW とは Digital Audio Workstation の略で，実際に楽器を持っていなくてもコンピュータを使用して音楽作成をできるソフトである。今回はこのソフトを使用して曲を読み込み，BPM を合わせて別のトラックにピアノロールを MIDI データのノーツと見立てて，リズムに合わせて配置していき，そのあと C#のスクリプトの配列にタイミングを入力して，譜面データを作成した。オーディオインタフェースを買うと付属してくる(図４)。

図４ SONAR LE

(e)Clip Studio Paint Pro(ペイント)

このソフトでは，ゲームに使用するレーンやノーツ，他にも難易度のイラストや，タイトルのイラスト制作に利用した。１か月間は無料でフル機能使用でき，ペンタブレットを買うと付属していることもある(図５)。

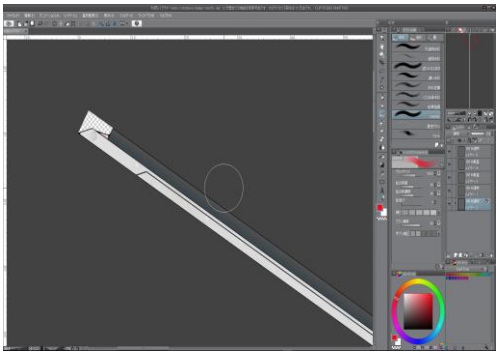

図５ Clip Studio Paint Pro

(f)Live2D Cubism Editor 3.0

Live2D はイラストをレイヤーごとにポリゴンを打ち込み，パラメータに合わせてイラストを動かすことができるソフトだ。トライアル版なら無料で使用できる(図６)。

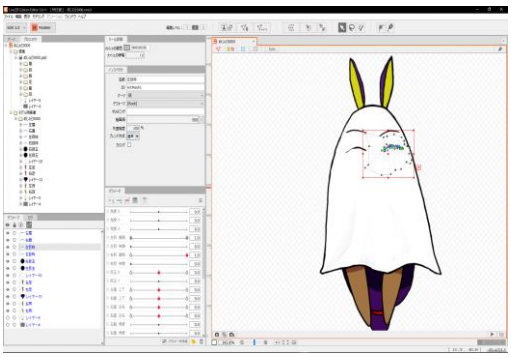

図６ Live2D Cubism Editor 3.0

(3)コンセプト

作り始める際，まずコンセプトから考えることにした。話し合った結果いろいろな候補が挙がったが，情報技術科と関係あるサイバー系に決定した。また，タイトルもコンセプトに合い，より伝わり

やすいものを話し合った結果，Rhythm の文字が入っている Algo=Rhythm(アルゴリズム)に決定した。

(4)タイトル画面

タイトル画面は,ゲームスタートとゲーム終了の移動のみでシンプルなものだが,ゲームを顔となる部分でもあるので,なるべくゲームの雰囲気が伝わるようなものにしようと考えた。ゲームの端から端まで楽しんでもらえるよう，Live2D というソフトを利用し，イラストを動かし，視覚的にも楽しんでもらえる工夫を施した(図7)。

図７　タイトル画面

(5)セレクト画面

(a)シーン移動

リズムゲームには必ず曲が複数存在するため，１曲をわかりやすくする必要がある(図８)。初めはカメラの視点移動で区切ろうとしていたが，今の技術的に難易度が高かったため，シーンを１曲ずつ作成しプログラムで移動させたほうが分かりやすいと思ったのでその方法を採用し，シーン移動に関するプログラムには，FadeManager を使用した。これを使用することでシーン移動だけでなく，フェードインとフェードアウトを簡単に行うことができる。また，タイトルやセレクトのボタンもカーソルがどこにあるか分かりやすくするために，重ねたときに少し透明にする処理や，矢印キーで移動する処理などを加えた(図８)。

シーン移動のソースコードの例

```
  using UnityEngine;
  using System.Collections
;

public class Move_Paradiss
ion : MonoBehaviour
{
    public void FadeScene(
)
    {
        FadeManager.Instan
ce.Loa
  dScene("select_Paradiss
        on",
        0.35f);
    }
}
```

図８　セレクト画面

図９ カーソルをボタンに重ねた時

(b)アニメーション

アニメーションはセレクト画面だけでなく，タイトル画面やリザルト画面にも使用されている Unity でできる機能で，Window のところを開き，Animation を押すことで使用できる。アニメーションは，指定した秒数の間にテキストやボタン，画像などを動かすことができ，ゲームを作成する際に必要な機能である。このセレクト画面では，シーン移動を終えた時に曲名，曲のジャケット，難易度選択ボタンを画面の外から中央に寄せるという感じにした(図 10)。

図 10 アニメーション制作一例

(6)プレイ画面

(a)プレイ画面について

プレイシーンに移動すると，時刻を格納する変数名 timar に 0 を代入し，曲を開始する。

リズムに合わせて判定ラインに飛んでくるノーツは，SONARLE(DTM)作成したタイミングデータの 1 秒前にノーツを生成し，生成した 1 秒後に判定ラインに当たるようにする。

指定されたキーを押すと押されたときの秒数（timer）とタイミングデータを比較し，誤差がであれば結果を変数にプラスしていく。

(b)ポーズ

ポーズ画面ではゲーム内の時間を操作する関数 Time.timeScale を使った。この関数に 0 を代入してやることでゲームが止まる。便利な関数だがこれを使うときに注意すべき点は，0 を代入しても Update 内は動き続けるということ。このゲームでは Esc キーが押されたときに timeScale が 0 か 1 かを判定する変数を作り（変数名:pose），timeScale が今まで 1 だったなら 0 に，0 だったならなら Esc キーが入力されたか見る変数(変数名:hazime)に 1 を代入する。

ポーズ画面からゲーム画面に戻るには先ほどの Esc 判定変数に 1 が代入されたうえで，シフトキーを押す必要がある 。今回は使わなかったが，Fixed Update(一定の秒数で呼ばれる。これなら中身が実行されなかったはず…)を使ってもよかったかもしれない。

ボタンを表示させるのは SetActive という関数を使った。この関数を false にすることでオブジェクトが非表示になり，true にすると表示する(図 11)。

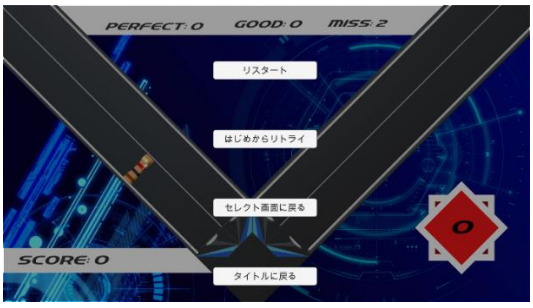

図 11 ポーズ画面 Esc 入力

このゲームでは始めにオブジェクトを非表示にしておき(図 12)Esc キーが入力

され，timeScale が 1 なら SetActive を true(表示)にする。そうでないなら false(非表示)にする。

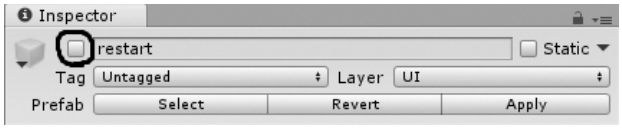

図 12 ボタン非表示

(7)リザルト画面

上記で記述した通り，リザルト画面(図 13）にはアニメーションを使用した。4 隅の 2 本の線は，1 つのアニメーションで動かす事が出来たのでそれぞれ右上，左上，右下，左下の 2 本の線をまとめて動かすアニメーションを作成した（図 14)。スコアや判定の振り分けは 1 つずつ作成し，ちょうどよく少しずつずらすようにメンバーとも協力し丁度良い重なり具合を作成した。

選択メニューに戻る際に押すボタンとは別にテキストボックスを作り，同時に定位置に移動・表示し，NEXT が表示しきったタイミングから押せるようにアニメーションを調節した。

図 13 リザルト画面

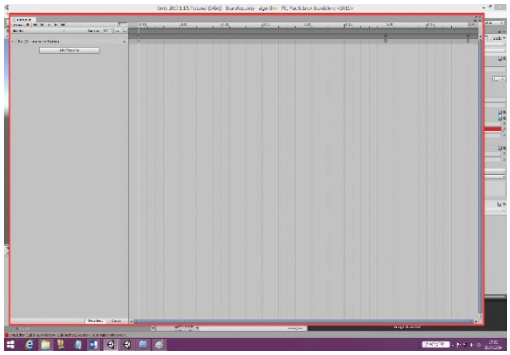

図 14 左上の線のアニメーション

3．研究のまとめ

実際，企画としては結構簡単に決まったことだったが，全員 Unity を使用したこともなく，また Unity で使用した言語が C#ということもあり，完全な手探り状態で始まった。分からない動作や挙動はインターネットや参考書で調べ，1 つ 1 つ僕らが目標とするゲームに近づけていき，結構満足する作品ができた。しかし，長押しのノーツやギミックを取り入れたノーツを考えていたが，それを取り入れることができなかったところが少し残念である。だがこれも，もう少し計画をしっかり固めていればできていたことだと思うので，この反省を就職先や進学先で活かせればいいと思った。

～川下　直輝からのメッセージ～

研究の課題が自由なので自分達が興味のあるゲームジャンルでやってみたがソースがたくさんあるゲーム（RPG など）とは違い，一から作るものが多かった。私はタイトルとセレクトを担当したが，最初に取り掛かっていたシーン移動から躓いてしまい，正直 1 年でどこまでできるか不安になったが,1 つの動作ができるごとに達成感を感じることができ，それが小さいことでもモチベーションを上げることができたので，これがモノづくりの大切さの 1 つなのだと感じることができた。

～川上　真心からのメッセージ～

私はゲームを作ったことがあるわけでも音ゲーができるわけでもなかったが，この班で一緒にゲームを作ることになった。最初はちゃんと作れるのか不安だったが経験者が居たため少しずつ教えてもらい，またネットを検索することによりとりあえず使えるものを作ることはできたと思う。ネットに転がっているコードを使うのは他人の力を使っているようで少し嫌だったがネットのコードを自分たちのゲームに合わせて少しずつ変えていくうちに段々と Unity 使い方を学べたので，今思えばネットを使っていてよかったと思った。思っていたよりも時間がないのでこの先課研をする人は，わからないときは積極的に先生に聞いたりネットなどで調べたりして少しずつ内容を理解していって欲しい。

～久保　友人からのメッセージ～

今回 Unity でリズムゲームを作る前から Unity でブロック崩しやボールころがしなどのゲーム作成をしていたこともあり，基本的な操作はできたのでその分の時間は短縮することができたが，リズムゲームのアルゴリズムを実際にゲームにするのに時間がかかった。ほかにもゲームを作るためには必要なイラストや，アニメーション素材を作成する時間を確保することができずスケジュールを圧迫してしまったので，計画段階で作業ごとの時間を挙げておけばよいと思いました。そのためには経験や予測を必要とするので，課題研究で気付けて良かった。

～築澤　輝彦からのメッセージ～

自分はリザルト画面の作成と作ってもらった譜面のデータをメモ帳に写す作業を行った。「何だ、そこまで大変な事やってないじゃないか」と言うかもしれないが譜面のデータの easy はまだ少ないが hard や隠し譜面のときは，1 曲 7～9kB を手作業で正確に打たなければならなかったので単純だが凄く疲れる作業だった。リザルト画面は，もっと良いものにしようと思い，熱が入る作業になった。ゲーム作りの大変さを改めて知った。この事が将来役に立つかは分からないが，凄く楽しかった。

参考文献

Unity スクリプトリファレンス
https://docs.unity3d.com/ja/current/ScriptReference/

Unity シーン遷移時のフェードイン・フェードアウトを実装してみた
http://naichilab.blogspot.jp/2013/12/unity.html

初心者のための Unity 超入門
http://libro.tuyano.com/index3?id=2816003

Unity で音ゲーのモックを作りました
http://developer-ryo.hatenablog.com/entry/2015/11/11/232205