Unityによるタイピングゲーム制作

高見 翔平　山口 志門

１．研究概要

ゲームはタイピングゲームを作ることにし，ノンプレイヤー対戦型にすることで楽しんでもらいながらタイピング技術を身に付けてもらえるようなものを作った。ゲームエンジンはUnityを使うことにした。３つの難易度を用意し基準は情報技術科入学当初，タイピングに慣れ始めた頃のタッチタイピングレベル辺りを想定して設定した。

２．研究の具体的内容

(１)Unityとは

Unityは統合開発環境を内蔵し，複数のプラットホームに対応するゲームエンジンである。開発はユニティ・テクノロジーズ社である。ウェブプラグイン，デスクトッププラットフォーム，ゲーム機，携帯機器向けのコンピュータゲームを開発するために用いる。ゲームエンジン自体はC言語/C++で書かれており，ユーザー（開発者）はC#を用いたプログラミングが可能である。

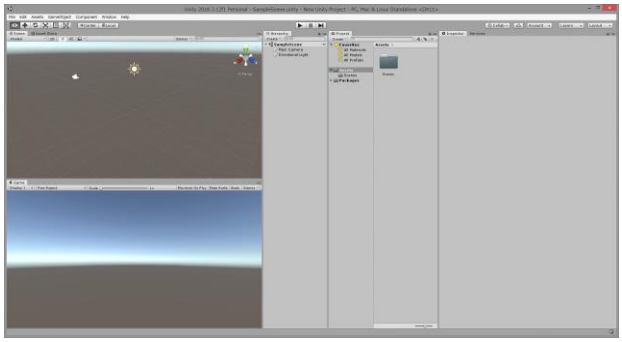

図１ Unity

(２)タイピングゲームの概要

コンピュータのみに今回のタイピングゲームは対応させることにした。難易度は３つに分けて制作し，出された問題を打ち込み正しく入力することで敵にダメージを与え敵の体力を０にすることでクリアする形式にした。

また，タイムリミットが設定されておりタイムオーバーでプレイヤー側がゲームオーバーになる。難易度ごとに正解したときのダメージ量やプレイヤーの体力，出題するお題の内容を調整した。プレイ中はお題の下にタイプミスの数を表示するようにした。クリア後はクリア画面へ自動で飛ぶようにし，その後スペースキーを押すことでリザルト画面へ移動し，正解率やキーのミスタッチ数を表示する。そしてもう一度スペースキーを押すことで難易度選択画面へ戻れるようにした。

また，ステージごとにBGMを搭載して無音で飽きることのないようにした。

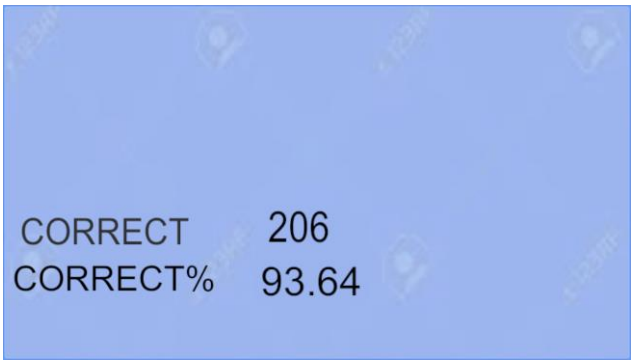

図２ リザルト画面

図２のCORRECTはキータイプの正解数，CORRECT%はキータイプの正解率の表示をするようにした。

図３のNormal, Hard, Hellの右に表示されているアルファベットをキーボードで打ち込むことで選択した難易度へ飛ぶことができるようにした。難易度をクリックするたびにマウスを持たないといけない手間を省くため，キーボード入力のみで全てのシーン移動をできるようにした。

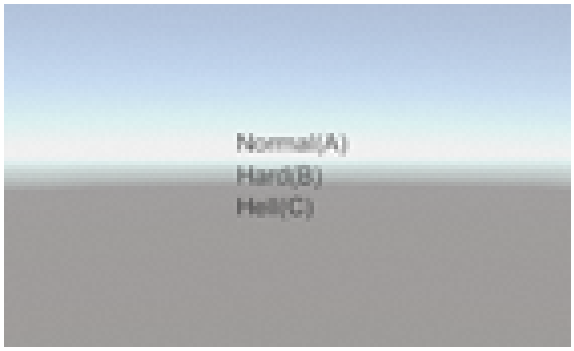

図３ 難易度選択画面

当初はロード画面をシーンで切り替えする予定だったが，正解率の値の受け渡しや複数のプログラミングしたスクリプトの噛み合わせでエラーが多発し難しかったので，１つのシーンでパネルの表示，非表示をすることでリザルト画面とロード画面を切り替えることができるようにした。

(３)シーンの作成

図４のように難易度選択画面(Choise)，最低難易度(Normalscene)，普通難易度(Hardscene)と最高難易度(Hellscene)の４つのシーンを作成した。最初は他にもいろいろな機能を持たせたシーンを作成していたが勉強しながら進めている間にまとめられるものをまとめて最終的に４つに落ち着いた。

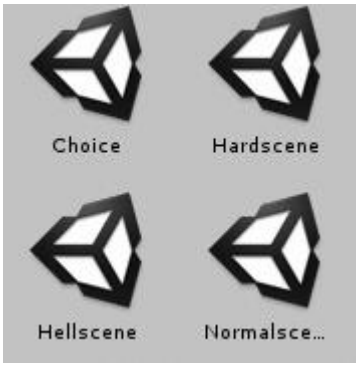

図４ 作成したシーン

まずは，Asset Store のフリー素材の中から敵となるオブジェクトをインポートしてきた。その後，体力ゲージとなるものを置き，テキストを配置した。背景の画像の置き方や図５の体力ゲージの作成方法においては「Qitta」というサイトを参考にしたのでリンクを最後に記載する。

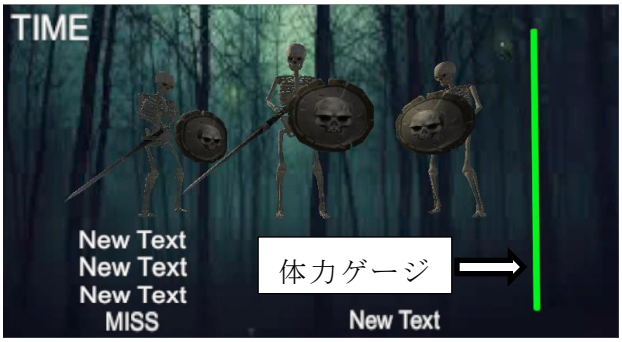

図５ 作成したシーン

(４)スクリプトの作成

図６は Visual Studio というマイクロソフトにおける統合開発環境ソフトでプログラミング言語は C#を使い必要な役割に応じて作成したスクリプトである。主に難易度ごとに動作するタイピングゲームのお題表示や体力判定，カウントダウンを作成した。このゲームを作成する際に初めて C#を使ったためまだ慣れておらず，今まで習ってきた C 言語と勝手が違うこともあったため不具合が続出したが試行錯誤で何とか対処することができた。

図６ 作成したスクリプト

また，タイピングゲームの稼働できる状態にしたときに他の人にテストプレイしてもらった。その際人によってキーボードの入力パターンに違いがあることが分かった。そこで対応させるスクリプトを追加で作成することにした。１つの文字に対して複数の入力パターンがあるものは，ゲームプレイ中の表示上では１通りになっているがスクリプトで複数のパターンを同じ判定に統一するようにしたことで正解として判断されるようにした。最初は判定の設定が甘く本来間違いのはずのキーでも判定を通り抜けていたので，より正確に設定することで複数のパターンを間違いなく判定させることができた。

```
    if (Input.anyKeyDown
&& (!Input.GetMouseButton(0) && !Input.GetMouseButton(1) && !Input.GetMouseButton(2)))
    {
        // 今見ている文字とキーボードから打った文字が同じかどうか
        if (Input.GetKeyDown(nOR[index].ToString())) {
            old_char = nOR[index];
            // 正解時の処理を呼び出す
            Correct();
        } else if (nOR[index] == 'n' && old_char == 'n' && Input.GetKeyDown(nOR[index + 1].ToString())) {
            Correct();
            Correct();
        } else if (nOR[index] == 'z' || nOR[index] == 'j' && nOR[index + 1] == 'i') {
            Correct();
        } else if (nOR[index] == 'f' || nOR[index] == 'h' && nOR[index + 1] == 'u') {
            Correct();
        } else if (nOR[index] == 't' && nOR[index + 1] == 'y' && Input.GetKeyDown(KeyCode.C)) {
            Correct();
        } else if (old_char == 'c' && nOR[index] == 'y' && Input.GetKeyDown(KeyCode.H)) {
            Correct();
        } else if (old_char == 'c' && nOR[index] == 'h' && nOR[index + 1] == 'i') {
            Mismis();
            Correct();

        } else if (old_char == 's' && nOR[index] == 'h' && nOR[index + 1] == 'i') {

            Mismis();
            Correct();
        } else if (old_char == 's' && nOR[index] == 'y' && Input.GetKeyDown(KeyCode.H)) {
            Correct();
        } else {
            // 失敗時の処理を呼び出す
            Mistake();
        }
    }
```

図７　複数パターンを判定させるプログラム

(５)BGM の取得

フリー音楽素材提供サイト「魔王魂」というところからタイピングゲームに使用する BGM を引用した。対戦中に使うもの，難易度選択画面用，リザルト画面用とロード画面の効果音用の計６つをダウンロードした。

(６)素材のドット絵の作成

ロード画面に使用するドット絵を作成したいと思い MagicaVoxel というモデリングソフトを使って作成した。キャラは場違いなキャラではなく愛嬌のあるもので敵として登場するスケルトンをドット絵で作った。

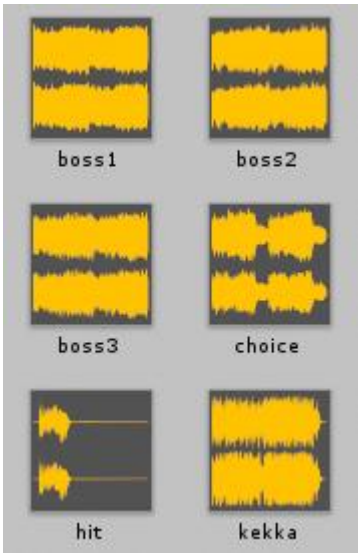

図８　ゲームで使う BGM

図９　MagicaVoxel で作ったスケルトン

(７)組み合わせ

スクリプトやシーンをアタッチ(ドラッグ＆ドロップ)し，ビルド＆ランを実行することでタイピングゲームとして動作するようにした。アタッチした際にスクリプトの体力ゲージの動きと表示上で誤差があり調整する必要があった。またレイヤーの前後の関係でオブジェクトやテキストの配置を調整した。試しに実行すると大きく画像がズレており，メインカメラにサイズを対応させていなかったので対応させた。

図 10 動作中のゲーム

３．研究のまとめ

（１）まとめ

もともとゲーム制作には興味があった。今回，課題研究で制作するゲームを決定する際に情報技術科へ入学している人たちがタイピング技術に差が大きく出ていることに着目して技術差に対応したタイピングゲームを制作することにした。最初は，Unity も C#の使い方もわからず練習しながら勉強するだけで手一杯になってしまい作業を本格的に始められたのは夏休み前くらいからになった。作業をプログラミングとシーン作成の２つに分担していたが，組み合わせるときに方法を考えていなかったのでスクリプトを地道にコピーして移していた。手間がかかり無駄に時間を割いてしまった。その後パッケージのエクスポート方法を学ぶことで円滑に進めることができた。結果的に完成したからよかったが，放課後遅くまで残り作業する日が多かったので段取りをしっかり組めていたら余裕が作れると思った。

（２）感想

課題研究の１年という短い時間の中でタイピングゲームを制作することで Unity や C#を知ることができ，ゲーム制作の難しさや新しいことをする難しさも知ることができた。分担作業することでかなりの時間短縮ができた。また，自主的にものを進めることが新鮮で楽しむことができた。文化祭までの期間で予定していなかった BGM の追加や他の人に意見をもらい同時押し対応を実装することができたのはよかったと思う。敵キャラのモーションを追加できなかったのは少し心残りである。今回使用した Unity はまだまだ奥が深く多機能なものだったのでもっと理解を深めたいと思った。制作したタイピングゲームが情報技術科で使ってくれると私たちとしては嬉しい。

４．参考文献

「Wikipedia Unity(ゲームエンジン)」
https://ja.wikipedia.org/wiki/Unity_(%E3%82%B2%E3%83%BC%E3%83%A0%E3%82%A8%E3%83%B3%E3%82%B8%E3%83%B3)

「Qitta」
https://qiita.com/

「魔王魂」
https://maoudamashii.jokersounds.com/