9. 音楽ゲーム「GAT」の作成

9 番　後藤　樹　　14 番　髙橋　毅

指導教員　飯坂　覚

1.　目的

最近、スマートフォンや家庭用ゲーム機で音楽ゲームが多く発売されている。私たちは音楽ゲームをプレイしていて、どのように作られているのか関心があり、音楽ゲームを作成してみようと考えた。具体的には、ノーツ（リズムアイコン）を曲に合わせて流すときの処理やノーツが押された判定処理といった音楽ゲームに必要な知識を理解することが目的である。

開発言語は、就職後必要となる C#を用いている。

2.　研究概要

入力装置として、ゲームコントローラを作成し、それを用いてパソコンで遊べる音楽ゲームを作成する。開発環境を表 1 に示す。

Unity とは、ゲーム開発のためのエンジンである。メリットとして 3DS、PS4 などをはじめ、多くのプラットフォームに対応できる。

REVIVE USB [1)]とは、キーボード入力の割り当て、マウス操作ができるマイコンである。

表 1. 開発環境

OS	Windows10
IDE	Unity
使用言語	C#
使用機器	REVIVE USB (AD00007) 押しボタンスイッチ(BPS60-RD)

3.　ゲームコントローラの作成

3.1　REVIVE USB の作成

REVIVE USB は半田付けを用いて組み立て、押しボタンを取り付けた。

3.2　アクリル板の加工

コントローラの筐体には、透明なアクリル板を使用した。Illustrator で設計し、産業デザイン科にレーザ加工を依頼した。押しボタンスイッチの入る穴も寸法通りに加工することができた。曲げ加工には、アクリル板専用ヒーターを使用した。

3.3　コントローラの説明

図 1 のように 5 つ配置したボタンは、ゲームプレイシーンで作成したノーツラインのタップ位置に相当する。

曲選択時、真ん中のボタンとその左右のボタンを使用する。左が左移動、真ん中が決定、右が右移動となる。

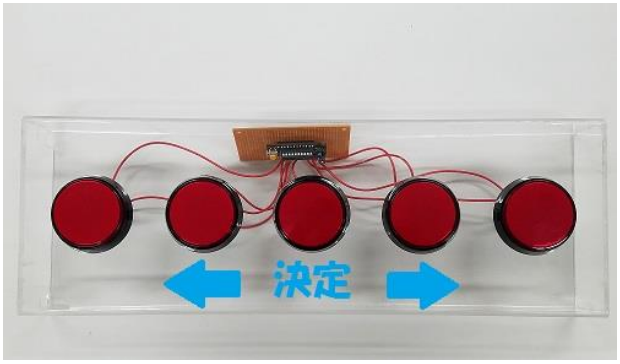


図 1. コントローラの説明

3.4　タップ音の改善

コントローラをタップ（ボタンを叩くこと）すると、ゲームの音楽が聞きにくくなる音だったため、コントローラの騒音改善を図った。

騒音測定には、電子技術科の無響室を借りて、スマートフォンアプリの騒音測定器 [2)]を使用した。図 2 で示すように改善前のタップ音は平均 64.4dB となった。

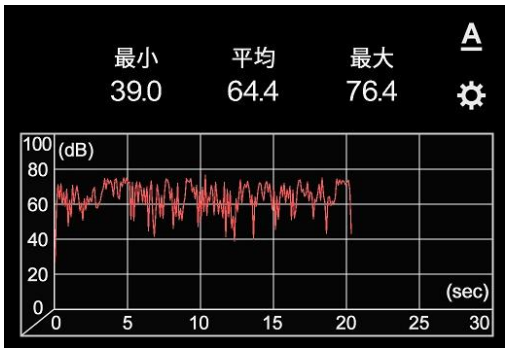


図 2. 改善前の騒音測定結果

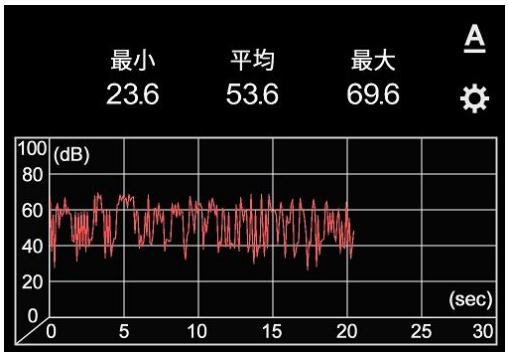


図 3. 改善を施したときの騒音測定結果

騒音改善にはボタン内部とコントローラ内部にグラスウールを詰め、さらにコントローラの下に防振マットを敷いた。そのときの音の大きさは平均 53.6dB となり、約 10dB 軽減された。

4. 譜面作成

譜面作成には「DuNo（デューノ） フリー音楽ゲーム [3]」の一部分であるノーツ専用の譜面作成ツールを使用した。クリックで好きなところにノーツを置くことができ、その譜面は音楽ゲームとしてプレイできる。

図 4 は、実際に譜面作成している様子である。ウィンドウを縦に 5 列均等に分けてノーツを作成した。また、ノーツのタイミングは、BPM（Beats per minute）を基準に作成した。実際に多くの友人にプレイしてもらい、音楽と譜面がずれているかを確認して、調整しながら作成した。

図 4. 譜面作成している画面

5. 音楽ゲーム「GAT」の作成

Unity ではゲームの 1 画面をシーンといい、3 次元で作成する。それぞれのシーンは異なるオブジェクトやスクリプトを追加することができる。

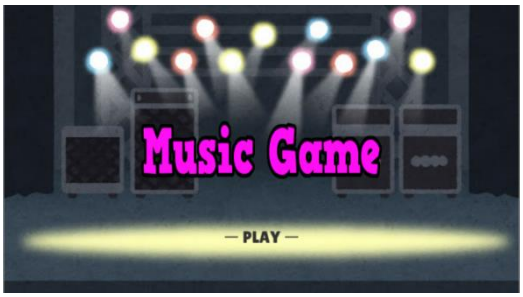
GAT には 4 つのシーンを用意することにした。

①ゲームスタートシーン
②曲選択シーン
③ゲームプレイシーン
④スコアシーン

5.1 ゲームスタートシーンの作成

ゲーム起動時､最初に表示する画面を作成した。コントローラのいずれかのボタンをタップすることで曲選択シーンに移動することができる。

図 5. ゲームスタートシーン



5.2 曲選択シーン

曲とその難易度を選択できる画面を作成した。図 6 の赤枠部分が曲のリスト、緑枠が難易度である。曲、難易度の順で選択する。これらは図 1 のコントローラのボタン操作で選択する。

難易度は Easy、Normal、Hard と設定している。難易度を選択することでゲームプレイシーンに移動することができる。

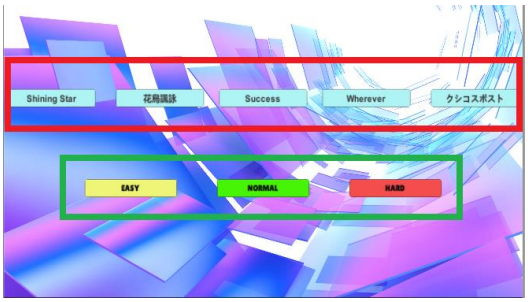


図 6. 曲選択シーン

5.3　ゲームプレイシーン

平成 28 年卒業研究など [4,5]を参考にノーツ、ノーツライン、判定ライン、ゲームスタートボタンのゲームオブジェクトを作成した。図 7 にゲームプレイシーンを示す。3 次元で作成しているので、オブジェクトを z 軸方向に変化させることができる。

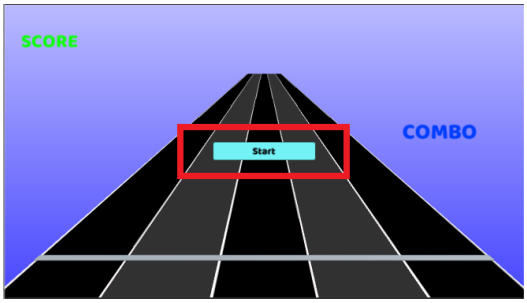


図 7. ゲームプレイシーン

ゲームプレイシーンでは、次の 2 つの処理が行われている。

(1) 音楽の再生とスタート

図 7 の画面真ん中のボタン（赤枠）をコントローラで押すと音楽が再生され、ゲームスタートとなる。

(2) タイミング判定処理

図 8 の譜面ファイルでノーツが流れてくるタイミング（赤枠）と場所（緑枠）を決めている。

また、流れてくる場所は左から順に 0～4 とノーツラインに割り当てた（図 9）。

例えば、図 8 の譜面ファイルの 1 行目はゲームスタート時から 0.92 秒経過するとノーツを生成し、図 9 で割り当てた 1 のノーツラインから流れてくることになる。

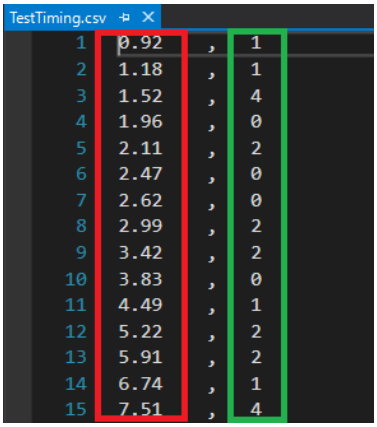


図 8. 譜面ファイル

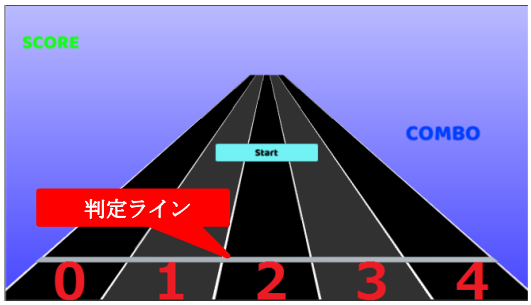


図 9. ノーツの流れてくる場所

判定ラインにノーツが接触しているときにタップすることでスコアを加算する。譜面ファイルで設定されたタイミングとコントローラをタップしたときのタイミングで良悪を判定している。

具体的なタイミング判定は、ノーツが判定ラインに接触した際のタップ時のタイミングと譜面のタイミングを計算し、表 2 のとおり処理している。その様子を図 10 に示す。

表 2. 判定処理

ノーツの状況	時間	結果	スコア
判定ラインに接触	±0.03 秒以内	Perfect	+1000
	上記以外	Good	+500
判定ラインを超過	—	Miss	±0

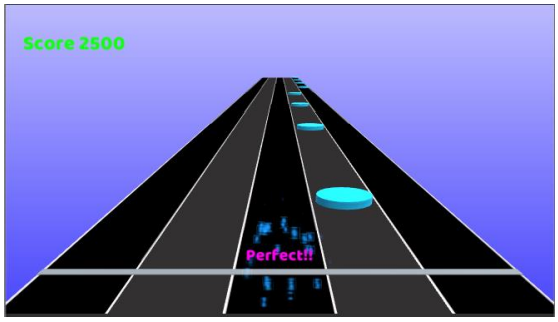


図 10. タイミングの判定

5.4　スコアシーン

図 11 では最終的なスコアや、Perfect、Good、Miss の頻度を表示している。

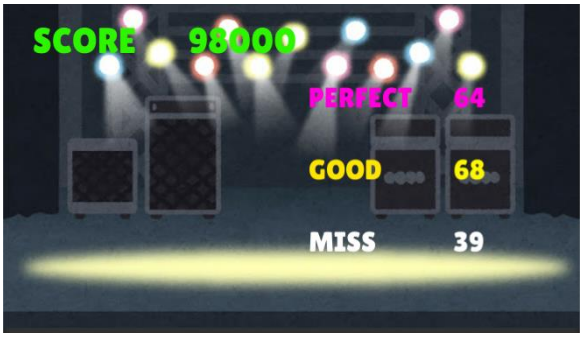


図 11. スコアシーン

また、スコアシーンでコントローラをタップすると図 12 のようにランキングが表示される。

図 12. ランキング表示

6.　動作確認

コントローラをタップすることで図 5 が図 6 に移動することができた。また、譜面どおりにノーツを生成することができた。コントローラをタップすると判定ラインにエフェクトが発生し、問題なくコントローラ操作ができた。タイミング判定の部分では、判定ラインにノーツが接触しているときにタップすることで、スコアが加算され譜面ファイルとタップ時のタイミングに応じた図 10 のような判定が表示された。

7.　まとめ

今回の研究は、音楽ゲームの作成にあたって必要な知識を理解することや就職先で使う C#の勉強が目的であった。はじめて使う言語で戸惑いもあったが作成を進めていき、基本ではあるが理解を深めることができた。また、譜面ファイルのタイミングに合わせてノーツの生成ができ、さらに、Unity に備わっている衝突判定を利用することでタイミング判定を実装することができた。以上のことから、音楽ゲームに必要な知識を学ぶことができた。

8.　参考文献

1)REVIVE USB
http://bit-trade-one.co.jp/product/assemblydisk/revive-usb/
2)騒音測定器（Sound Meter）
https://play.google.com/store/apps/details?id=com.gamebasic.decibel&hl=ja
3)DuNo フリー音楽ゲーム
http://duno.nagished.com/
4)Unity を使った音楽ゲームアプリの開発
（H28　福山将樹　尾形祐紀）
5)見てわかる Unity2017C#スクリプト超入門
（著者　掌田　津耶乃）