平成 28 年度 卒業論文

初心者のためのゲームプログラミング教育用教材の開発

函館工業高等専門学校 情報工学科 5 年

東海林研究室 細川航汰

目次

1 章 序論

第 1 節 英文アブストラクト

The purpose of this research is to make a game production tool that is compatible with Unity and a text as teaching materials for game programming education for beginners.

Unity is a game development platform widely used for game production. However, it is difficult for beginners to use Unity. Therefore, we created a game production tool that is compatible with Unity and be developed by C #.

In addition, we also make a text to use this tool for game programming education.

Key words：C#, Unity, game production tool

第 2 節 研究目的

ゲームプログラム未経験者~初心者（15~17 歳程度）のための教育用教材として、Unity[1]と互換性があり、かつゲーム制作を簡単にできるプログラミング言語学習環境を開発する。

第 3 節 研究背景

私が所属しているゲームプログラミング研究会では、Unity によるゲーム開発が行われている[2]。また、多数の新入部員がいるが、彼らの多くはプログラミング未経験者であり効率的な新人教育方法が必要になった。

Unity がプログラミング未経験者にとって難しい理由として、Unity のファイルサイズが大きいこと、使用するにはユーザ登録が必要なこと、実行に高いスペックを要求し、低スペックの環境では快適に動作をしないこと、スクリプトを扱うには C#または JavaScript の知識が必要であること等があげられる。 類似のプログラミング言語学習環境として、Scratch[3]があるが、自由度が低く、本格的なゲームプログラミングには向いていない。

第 4 節 開発環境

開発環境は以下のとおりである。

使用 PC	Windows10 Home 32bit
	Intel Atom CPU Z3735F 1.33GHz
	メモリ 2.00GB
使用ソフト	Visual C# 2010
	Unity 5
	Github
	Microsoft Office 365
使用言語	C#

2 章 簡単なサンプルゲームの作成

開発環境を作成する前に Unity のどの機能を実装するか選択する目的で、Unity を用いた簡単なサンプルゲームを作成した（図 1）[4] 。

作成したゲームから、以下の機能を開発環境に実装することにした。

・ゲームオブジェクト（キャラクター）を複製する機能
・スクリプトの Update でオブジェクトを一定時間で更新し移動させる機能
・キーボードの入力状態を判別してゲームに反映させる機能[5]
・オブジェクト同士の接触を判定する機能[6]

図 1 Unity で作成したサンプルゲーム

3 章 プログラミング言語開発環境の開発

第1節　開発方法

Unity のスクリプトリファレンス[7]を参考に、簡単な 2D のゲームの作成に必要なクラスを実装した。

また、Unity に対応させるため、コンソールではなく Windows のフォーム上で動作するように C#で開発した。

図 2 は作画ツールの Dia[8]を用いて作成したクラス図である [9] 。

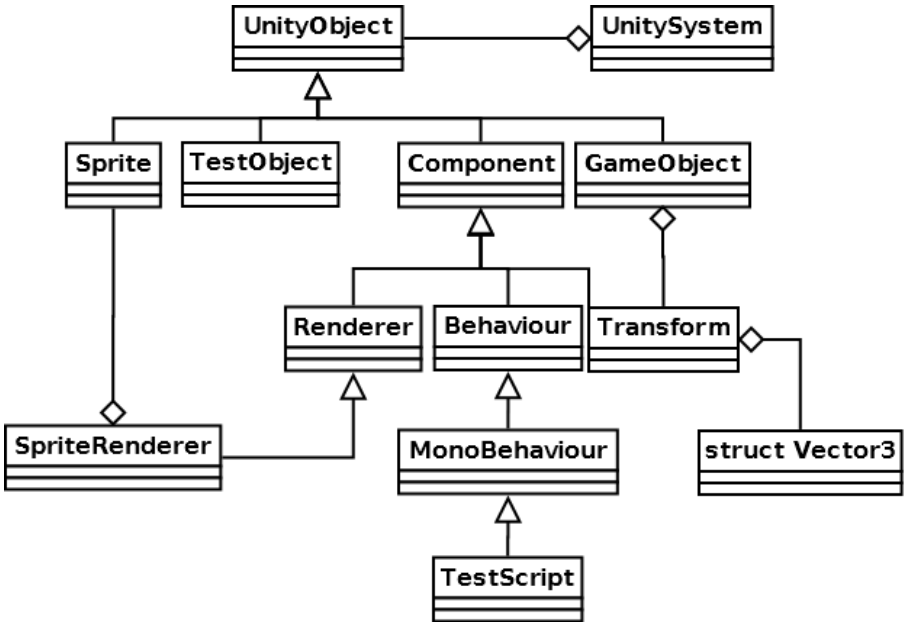

図 2 開発環境のクラス図

第 2 節 オブジェクトの作成

ユーザはゲーム内にキャラクターを表示するためにゲームオブジェクトを作成し、画像・座標・スクリプトをコンポーネントとしてオブジェクトに割り当てる。

図 3 に示した UnityObject クラスは Unity 上では Object クラスという名称である。

UnityObject
+ string name - bool DestroyFlag
+ UnityObject(string s) + void Destroy(UnityObject obj, float t) + static T Instantiate<T>(T original) + public static T Instantiate<T>(T original, Vector3 position) + virtual void Update()

図 3 UnityObject のクラス図

UnityObject クラスのオブジェクトは、開発環境の Form1.cs 内で作成する。コンストラクタには、オブジェクトの名前として、string 型の引数を使用する。

また、作成したオブジェクトは図 4 に示す UnitySystem クラスの AddNewObject を用いて UnityObject リストの hierarcy に追加する必要がある。

UnitySystem
+ static List<UnityObject> hierarcy
+ static void AddNewObject(UnityObject obj) + static void Update()

図 4 UnitySystem のクラス図

第 3 節 コンポーネントの作成

第 2 節で作成したオブジェクトをキャラクターとして表示し、移動などの動作をさせるためには、コンポーネントとして画像・座標・スクリプトが必要である。

図 5 は画像を表示する機能を持つ SpriteRenderer のクラス図である。コンストラクタには、引数として画像ファイルのパス、SpriteRenderer につける名前を string 型で渡す。画像の描画は UnitySystem 内の Update()を実行することで自動的に行われる。

SpriteRenderer
+ Sprite sprite - Image image
+ SpriteRenderer(string pass, string name) :base(name) - ~SpriteRenderer() + override void Update()

図 5 SpriteRenderer のクラス図

図 6 に示した Transform クラスは、座標を表すクラスである。メンバである postion の型 Vector3 は float 型の変数 x,y,z を持つ構造体である。

Transform
+ Vector3 postion
+ Transform(string name) : base(name)

図 6 Transform のクラス図

図 7 の MonoBehaviour クラスはユーザが作成するスクリプトの元となるクラスである。このクラスはコンストラクタ以外のメンバ・メソッドを持たないため、実際にユーザが作成するスクリプト内にメソッド Start(), Update()を記述する必要がある。

MonoBehaviour
+ MonoBehaviour(string name) : base(name)

図 7 MonoBehaviour のクラス図

第 4 節 コンポーネントの割り当て

オブジェクトをキャラクターとして動作させるために、第 2 節で作成したオブジェクトに第 3 節で作成したコンポーネントを割り当てる。

図 8 の GameObject クラスは UnityObject の子クラスで、コンポーネントのリストを持つ。Attach 関数に引数としてコンポーネントを渡して実行することで、オブジェクトにコンポーネントを割り当てることができる。

GameObject
- List<Component> Components
+ GameObject(string name) + void Attach(Component com) + override void Update()

図 8 GameObject のクラス図

第 5 節 実行画面

例としてスクリプトは図 9 の Testcript.cs, オブジェクトには図 10 の TestObject.cs を使用するものとする。

```
public class TestScript : MonoBehaviour
{
    public TestScript(string s):base(s)
    {
        Start();
    }

    public void Start()
    {
        this.transform.postion.x = 0;
        this.transform.postion.y = 0;
    }

    public override void Update()
    {
        this.transform.postion.x+=1;
        this.transform.postion.y += 2;
    }
}
```

図 9 TestScript.cs

```
public class TestObject : GameObject
{
    public TestObject(string s) : base(s)
    {
        Attach(new TestScript("Script_test"));
        Attach(new SpriteRenderer("tori.png", "ToriChara"));
    }
}
```

図 10 TestObject.cs

開発環境で作成したゲームを実行すると、図 11 のように画面には 4 つのボタンと白いキャンバスが表示される。起動時点ではキャンバスには何も表示されていない。

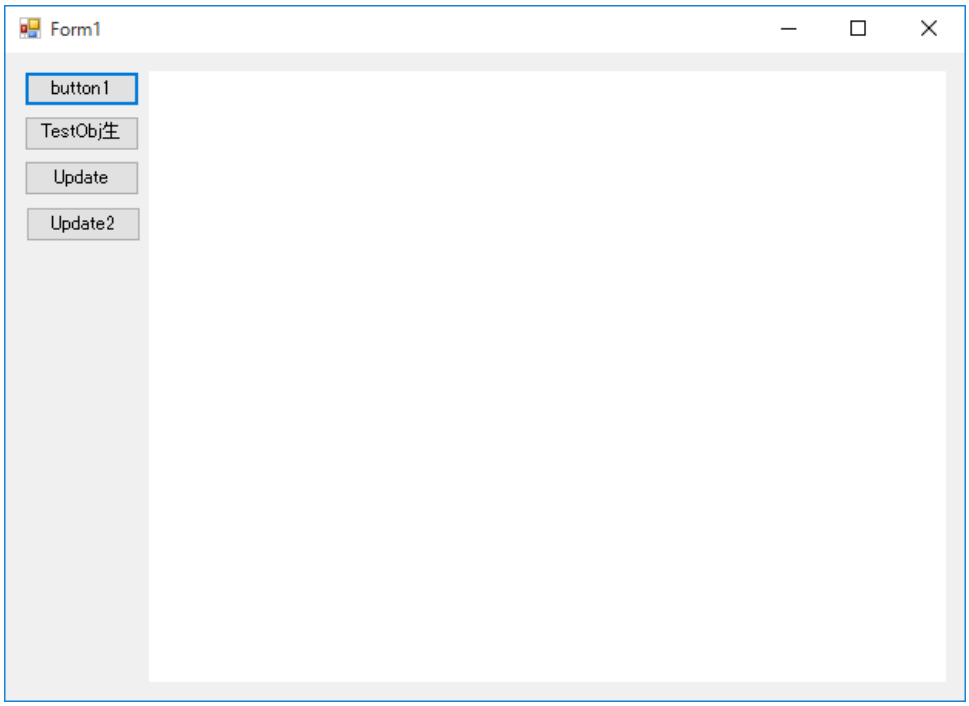

図 11 起動時の画面

今回のプログラムでは、図 12 のソースコードで示した通り起動した時点で”obj1”という名前のオブジェクトが生成される。

```
public Form1()
{
    InitializeComponent();
    obj1 = new TestObject("Tori_obj");
    UnitySystem.AddNewObject(obj1);
}
```

図 12 Form1.cs のコンストラクタ部分

ここで、上から 3 番目の[Update]ボタンを押すと、図 13 のようにキャラクターが画面上に表示される。

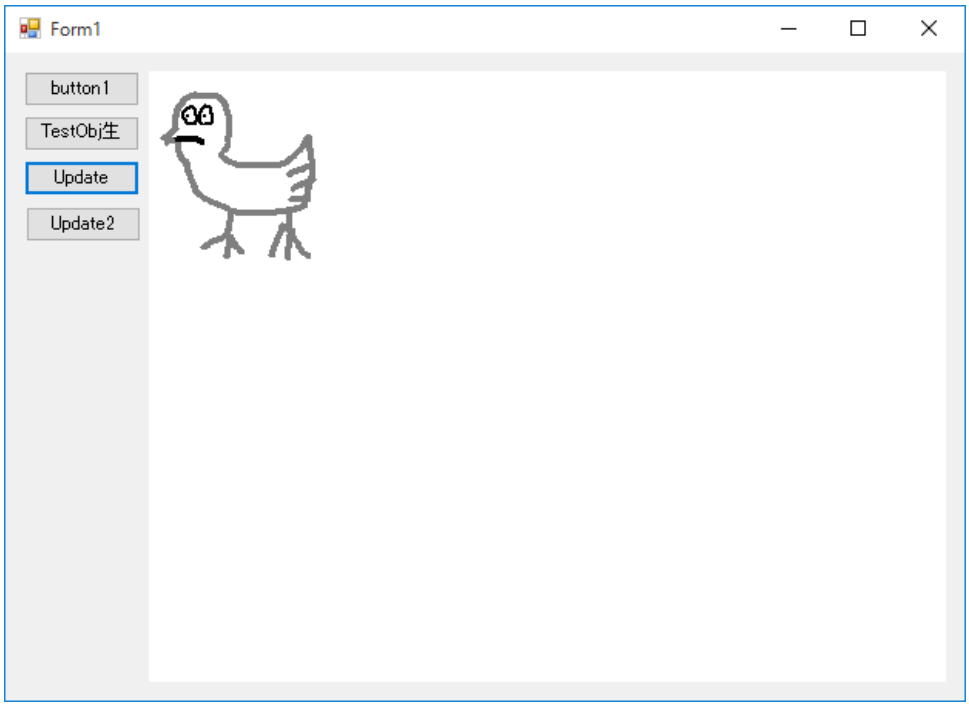

図 13 キャラクターが表示される

続けてもう一度[Update]ボタンを押すと、図 9 のスクリプトを一度だけ実行する。そのため、キャラクターが右斜め下方向に若干移動する。

一番下の[Update2]ボタンを押すと、図 9 のスクリプトを一定時間(例えば、30 ミリ秒)ごとに繰り返し実行する。そのため、キャラクターが右斜め下方向に連続的に移動する。

また、[TestObj 生成]ボタンを押すことで図 14 のようにオブジェクトの複製を行うことができる。複製されたオブジェクトも図 9 のスクリプトを持っているので同様に右斜め下方向に移動する。

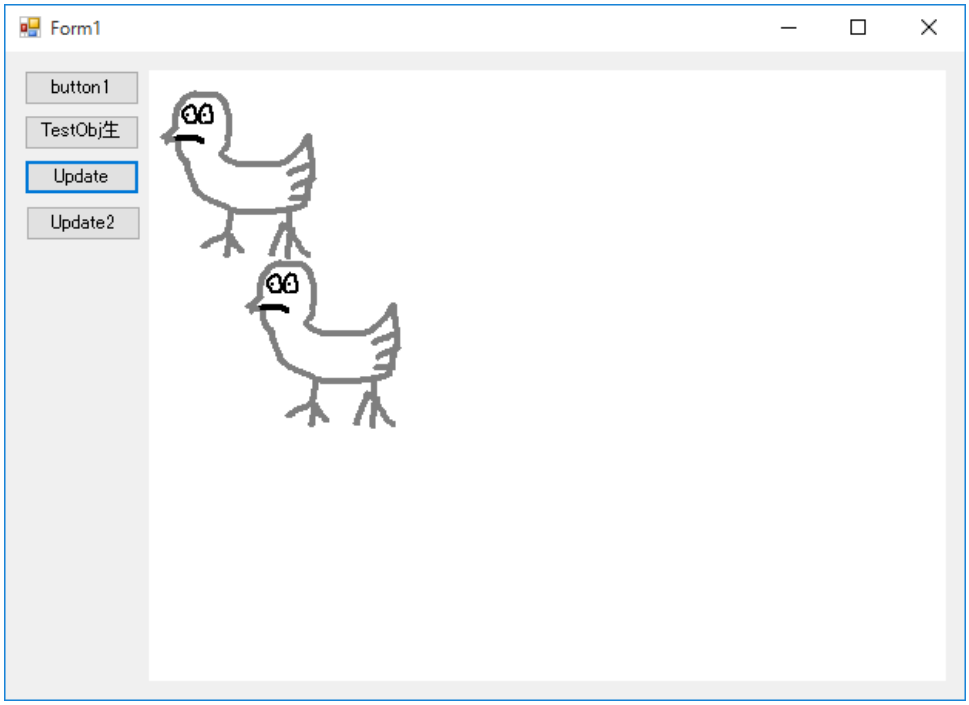

図 14 キャラクターが複製される

4 章 結果

当初の構想は、2 章で作成したサンプルゲームを本開発環境により再現することだったが、当たり判定とキー入力状態の判定機能が未実装である。

なお、開発環境の実行ファイルのファイルサイズは、約 13.5 キロバイトであることに対して Unity のサイズは 4 ギガバイトを超える。

このことからファイルの大きさは 10 万分の 3 まで小さくすることができたといえる。

5 章 課題

本研究を通して、Unity の機能をある程度実装することが出来たが、以下の様な課題が残る。

・未実装の機能

ゲームを制作するにあたって不可欠なキーボードやマウスからの入力を受け付ける処理と、キャラクター同士が接触しているかどうか判定する処理が未実装である。

・コマンドライン上でのコンパイル機能の実現

現時点でユーザが本アプリケーションを利用するには、Visual Studio などの開発環境でスクリプトをコンパイルする必要がある。バッチファイル等を利用して、プログラミング未経験者でも簡単にコンパイルできるようにすることで、開発環境をインストールする必要がなくなる。

・効率的な UI の実現

Unity では、ゲームオブジェクトの作成、ゲームオブジェクトの配置、ゲームオブジェクトとスプライト・スクリプトの関連付け等はすべてメニューやフォーム上で行うことができるが、本研究で作成した開発環境ではそれらの機能がなく、テキスト上で行う必要がある。

・学習用テキストの作成

プログラミング未経験者のための学習用テキストを作成する必要がある。

・ユーザ評価

ユーザから実際に使用してもらい、アンケートを通して評価や改善点などを得る必要がある。

6 章 まとめ

本研究はゲームプログラム未経験者のための教育用教材として、Unity と互換性があり、かつゲーム制作を簡単にできるプログラミング言語学習環境を開発した。

なお、作成した開発環境は GitHub 上[10]で公開している。ライセンスは ApacheLicense2.0 とする。

参考文献

[1] Unity - Game Engine:
http://japan.unity3d.com/

[2]函館高専ゲームプログラミング研究会ホームページ:
http://hnctgpgk.web.fc2.com/index.html

[3]Scratch - Imagine, Program, Share
https://scratch.mit.edu/

[4]Unity ゲーム開発 オンライン 3D アクションゲームの作り方:
竹内大五郎 石黒赳彦 髙橋誠史 香川寛和 河本健太郎 著

[5]Unity の Input で入力を扱う - Qiita
http://qiita.com/yando/items/c406690c9ad87ecfc8e5

[6]【Unity2D 入門 番外編】2 つのオブジェクトの物理判定とハンドラ呼び出し条件のマトリクス:
http://raharu0425.hatenablog.com/entry/2014/01/17/164919

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

[8]Apps/Dia - GNOME Wiki!:
https://wiki.gnome.org/action/show/Apps/Dia

[9]クラス図(Class Diagram) - UML 入門 - IT 専科:
http://www.itsenka.com/contents/development/uml/class.html

[10]GitHub - hnct12335/UnityEdu:
https://github.com/hnct12335/UnityEdu