TypingGame の説明

文章作成日：2020 年 12 月 28 日

1. アプリタイトル:TypingGame

2. 概要；自分でタイプする文章を決められるタイピングソフト

3. 製作意図：自作キーボードを作ったため、その配列に慣れるために自分で文章を決められるタイピングソフトが欲しかったため製作した。また、git を用いての開発を体験するために製作した。

4. 開発期間:2020 年 12 月 17 日～2020 年 12 月 28 日

5. 開発メンバー：自分一人

6. 担当範囲
   6.1. Scripts フォルダ内の cs ファイル、すなわち CustomInputField.cs、ExampleTextController.cs、GameDirector.cs、InputFieldController.cs、ResultDirector.cs、SelectDirector.cs の 6 ファイル
   6.2. Figures フォルダ内の folder_gray.png
   6.3. Scenes フォルダ内のシーンにおけるオブジェクトの配置
   6.4. その他必要な設定

7. 動作環境：windows10

8. 操作方法
   8.1. ファイルの選択:本アプリは指定されたテキストファイルを読み込み、そのテキストファイルの文章をタイピングするアプリである。そのため、まずファイルの選択をする必要がある。図 1 のファイル選択画面では Path と書かれたテキストボックスでファイルの絶対パスを指定するか、ファイルの形をしたボタンを押すことで現れるファイル選択ダイアログを用いてファイルを選択する。
   選択したファイルに問題が無ければ、Game Start!と書かれたボタンを押すことで次のゲーム画面に移る。問題がある場合、Path の下に赤字でテキストが現れるため、それを

参考に他のファイルを選択する必要がある。図 1 の場合は Path がファイルを示していないため、赤字が現れている。

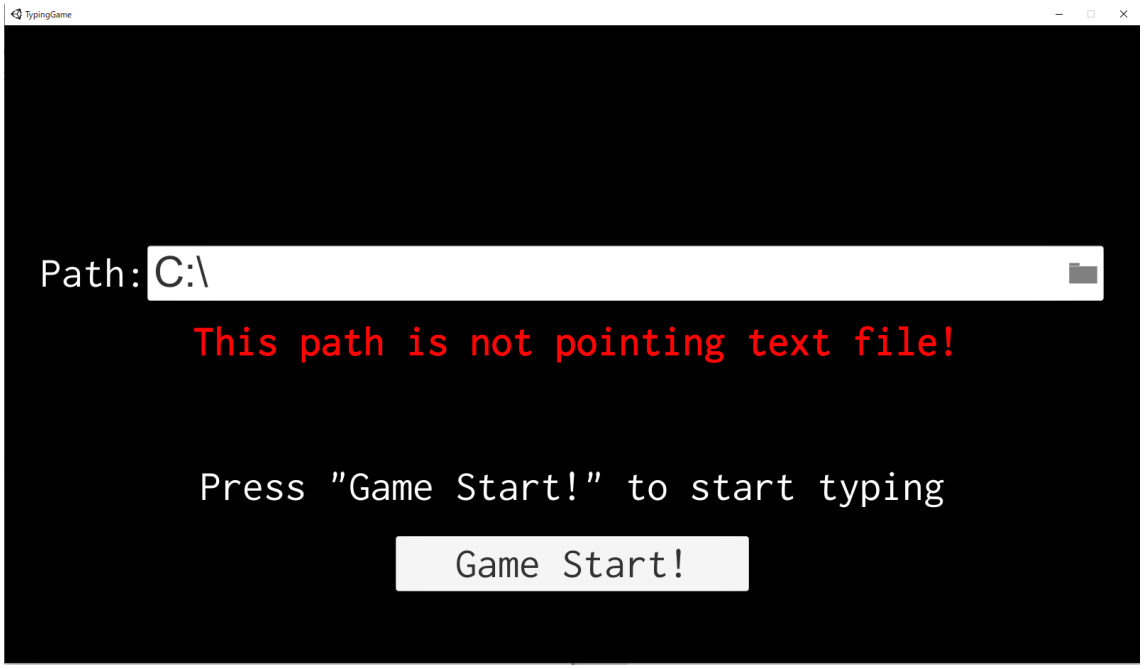

図 1. ファイル選択画面

8.2. タイピング：実際にタイピングをする。図 2 はタイピングを実際に行うときの画面である。下に現れる文字をテキストボックスに打ち込んでいく。そのとき、ミスがある場合はテキストボックスの色が赤になり、ミスなく文を打ち込めたときは緑色になる。緑色になったときに Enter を押すことで次の文に移る。図 2 のようにテキストボックスが黒い場合はまだミスもしていないが、文を全部打ち込んでいない状態である。途中でゲームを中断する場合は Esc を押すとファイル選択画面に戻る。

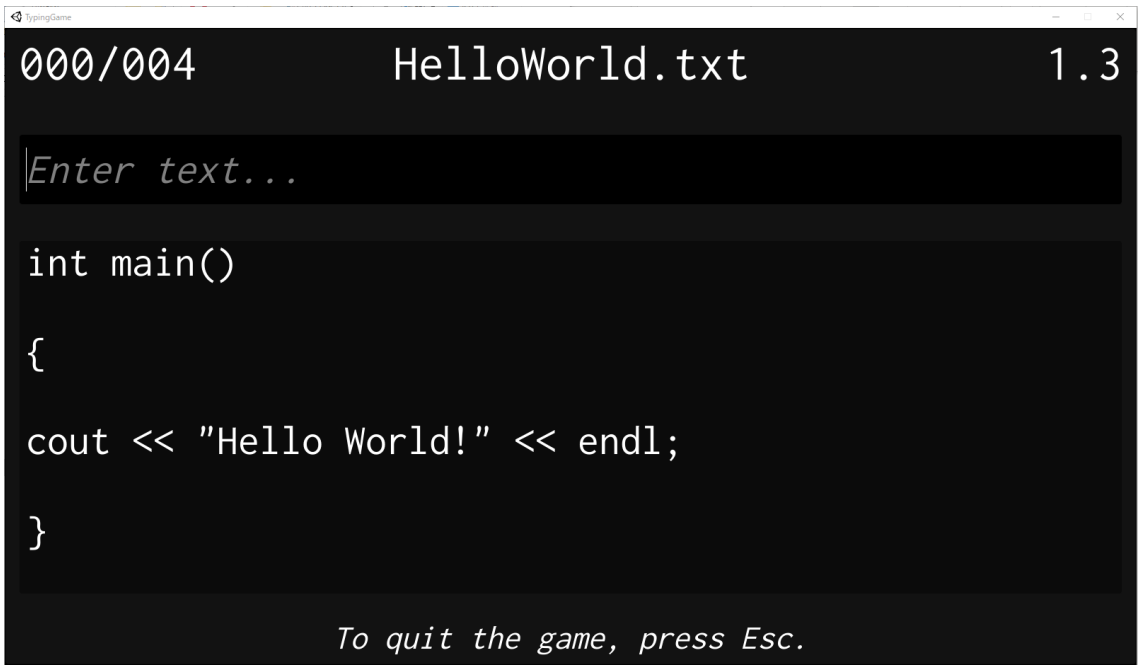

図 2. ゲーム画面

8.3. 結果を見る:文章を全て打ち込んだ時点で結果が表示される。図 3 は先ほどのゲームの結果の一例である。time が打ち終わるまでの時間、wps が一秒あたりに打ち込んだ文字数の平均である。この状態で Enter を押すともう一度同じファイルのタイピングが始まり、Esc を押すとファイル選択画面に戻る。

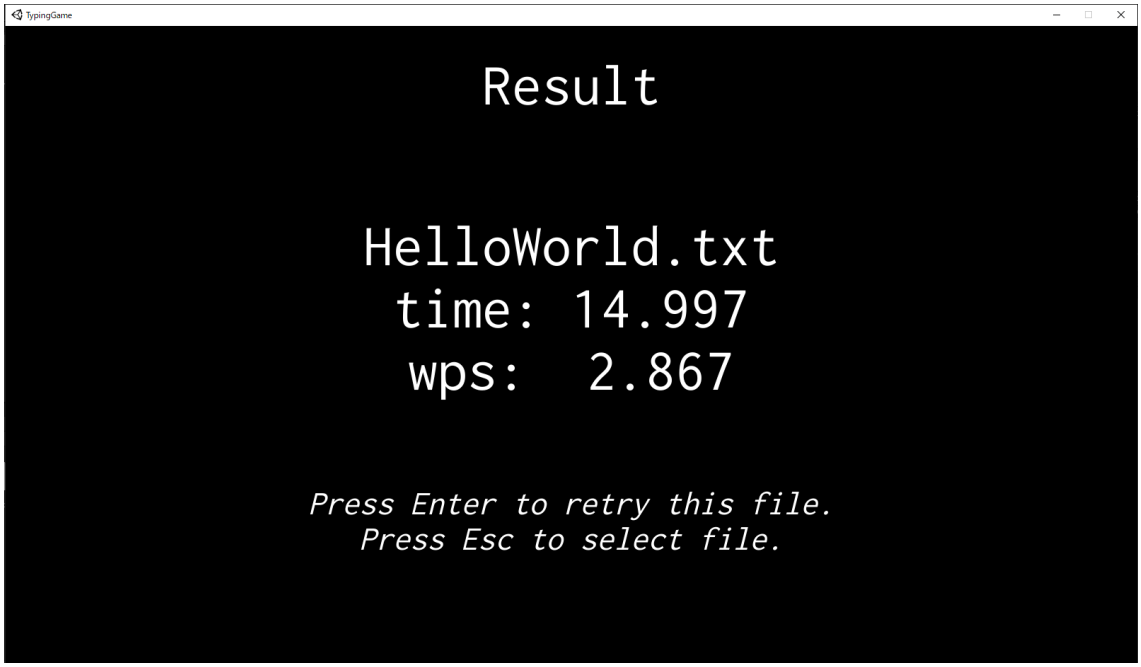

図 3. リザルト画面

9. 達成できたこと

9.1. よくあるタイピングソフトと違い、間違えた文字を削除する必要がある実際に使うときに近いソフトを制作できた。

9.2. 自分でファイルを選択できるタイピングソフトを制作できた。

9.3. ファイル選択の際にファイルダイアログを使用することができた。

9.4. Git を用いて開発できた。

10. 苦労した点

10.1. 選択画面のファイルダイアログを使用するために、dll ファイルをプラグインとして使用する必要があったが、Unity のプログラムファイルに置かれているものをコピーしてくる必要があったこと。

10.2. ビルドした exe ファイルを実行すると unity 上では上手く動作したダイアログが動作しなくなってしまったこと。

10.3. ゲーム画面でタイピングをする際に、unity のデフォルトのテキストボックスではいちいちマウスで選択する必要があったが、それを改善するために、自分でカスタマイズする必要があったこと。

10.4. Git において、ブランチ名やコミットに関する慣例が良く分からなかったこと。

11. 課題

11.1. public なメンバ変数を持つような設計をしてしまったこと。ある程度プロパティを用いるように変更したが、いくつかのメンバ変数に関して変更を忘れている可能性がある。また、設計に関してそもそも他のクラスに対してアクセスする必要の少ない設計を 学ぶ必要がある。

11.2. どのような操作をすればいいのか若干分かりづらいデザインになっていること。

11.3. 間違えた文字を記録しておき、苦手な文字を表示する機能がないこと。

11.4. Git のブランチ名やコミットの粒度、コメント等が適切でない可能性があること。

12. 使用したソフト

12.1. Unity

12.2. VisualStudio

12.3. InkScape

13. 参考

13.1. Unity スクリプトリファレンス
（https://docs.unity3d.com/ja/2018.4/ScriptReference/index.html）

13.2. C#プログラミングガイド（https://docs.microsoft.com/ja-jp/dotnet/csharp/programming-guide/）
13.3. Qiita(https://qiita.com/)