卒業論文

タイピングアプリ RenType の開発

関西学院大学理工学部

情報科学科 西谷研究室

学籍番号 27019611

田畑蓮

2023 年 3 月

概 要

西谷が担当している授業にコンピュータ演習がある. 生徒のタイピング力向上のために授業では MikaType を使用している [1]. しかし, キーボードにはホームポジションやそれぞれのキーを打つ指が定められているにもかかわらず, MikaType はキーの段ごとの練習になっているため, 練習効率が良くない.

よって, 本研究では西谷の授業で 1 回生がタッチタイピングを習得するための教材を前提とした Graphical User Interface (GUI) で操作するタイピングアプリ RenType を開発した.

まず, アプリの構成を考えていく上で既存の 3 つのアプリ MikaType, Shunkuntype, 寿司打を比較した. それぞれを比較し特に取り入れたいこと 3 点を主な目標としてアプリの構成を練った. それが, 単語のランダム性, 画面下部に手の形を模した図形を作成, 実力を簡単に測れるテスト機能の導入である. そこで, 構築には Python の標準ライブラリである Tkinter と, グラフを作成するためのライブラリである Matplotlib を用いた.

結果, RenType はキーの段ごとの練習ではなく各指ごとの練習ができ, キーボードにまだ馴染みのない初心者にとって有用なアプリとなった.

目 次

図 目 次

第1章　序論

1.1　研究の背景

西谷が担当している授業にコンピュータ演習がある. 生徒のタイピング力向上のため, 授業では MikaType というタイピングアプリを使用している [1]. しかし, MikaType を使って練習する上で主に 2 つ懸念点がある.

1 つ目は, 決められた数の単語を打ち終えた時のタイムを測定できないところである. MikaType にはポジション練習を除くと制限時間が設けられた練習の項目しかなく, 実際にプレイしてみて 60 秒キーボードを打ち続けることにしんどさを覚えた. 2 つ目は, タッチタイピングを習得する効率が良くないところである. キー配置を覚える練習において MikaType はキーの段ごとの練習をしており, ホームポジションに戻すという基本動作の習得ができない.

MikaType 以外にも, 西谷が開発した Shunkuntype というソフトがある [2]. こちらは MikaType とは違い, 各指ごとに練習するタイプのタイピング練習ソフトである. では, Shunkuntype を練習で使えば良いのではないだろうか. ところが, Shunkuntype は Character User Interface (CUI) で開発されており, 処理を行うときにコマンドを覚える必要があるため, 初心者向けではない.

よって, 本研究では西谷の授業で 1 回生がタッチタイピングを習得するための教材を前提とした GUI のタイピングアプリ RenType を開発した.

第2章　手法

2.1　既存アプリの比較

今回自作アプリの中身を考えるときに, MikaType, Shunkuntype, 寿司打 の３つの既存アプリを比較し参考にした [3]. 各アプリの特徴を以下にまとめる.

MikaType の良いところは, 比較したアプリの中で一番練習できる内容が豊富なところだ. 基本英単語を始めとした C 言語などのプログラミング言語で使われる単語, かな文字, キーポジションなどがあり, 練習をしていて飽きが来ない. また, 手の形が画面下部についており, 次に打ってほしい単語と指先が連動している. キーボードに視線を動かすことがないように工夫されているため, タッチタイピングの習得に向いている. しかし, キー配置を覚える練習をするにあたって, キーの段ごとの練習となっており, ホームポジションに戻すという基本動作の習得ができないため, 効率が良くない.

Shunkuntype は MikaType と違い, 1 回打ってホームポジションに指を戻すという基本動作を取り入れているため, タッチタイピングを習得する効率が良い. しかし, CUI で構築されている Shunkuntype は, GUI のアプリと比べてボタンなどで分かりやすく表現されていない. よって, 処理を行うときにコマンドを覚える必要があるため, パソコンに馴染みのない利用者に向けたアプリではない.

寿司打は, 回転寿司の席に座って寿司を次々に取るような感覚で遊べて, 利用者は楽しみながらタイピング練習できる工夫がなされている. しかし, 寿司打はタッチタイピングの基本を身につける工夫はされておらず, キーボードを直接見ながら速く打つという習慣がつくかもしれない.

以上の特徴を比較して、

- ランダムに単語が表示されること
- 画面下部に手の形を模した図形を作成すること

- テスト機能を入れて実力を簡単に測れる項目を取り入れること

以上3点を主な目標として RenType の中身を構築した. 理由としては, 西谷の授業で1回生がタッチタイピングを習得するための教材を前提としたアプリを作りたいと考えたからである.

2.2 Pythonで開発

RenType を利用して GUI のアプリに興味が湧いた学生のために, GUI アプリを容易に開発できる Python の標準ライブラリの Tkinter で開発した. これは図形やボタン等を簡単に作成できるもので, GUI のアプリを初めて作る時に使いやすいと考えられる. また, 練習やテストの履歴から, 成長具合の視認が可能なグラフを作成するために Matplotlib というライブラリを用いた.

2.2.1 Tkinter

Tkinter は, Window, Mac OS, Linux といった主要な OS に対応しており, Python で GUI を構築・操作するための標準ライブラリである [4]. Tkinter は, Tcl/Tk をベースにしたもので, Tk 部分を Python で利用できるようにしたものである. Tcl/Tk は, スクリプト言語 Tcl と GUI ツールキット Tk からなるものである. これにより、スクリプト言語である Python から簡単に GUI 画面をもったアプリケーションを作ることが可能になる.

2.2.2 Matplotlib

Python でアニメーション, グラフの作成などを行うためのライブラリである. 公式サイトに基本的な棒グラフや折れ線グラフを始め, 箱ヒゲ図やヒストのような統計図のサンプルコードが豊富に紹介されているページがあるため, 図を作成するときの参考になる [5].

第3章　結果と考察

開発したアプリはGithubに公開している[8]. そのファイル構成を付録Aにまとめている. RenTypeはMacとWindowsどちらでも利用可能である.

3.1　利用環境の構築

このアプリを利用するためには, Python 3 と Matplotlib をインストールしていることが必要である. インターネットで検索するとどちらもインストールの仕方をわかりやすくまとめている記事が出てくるため, ここでは省略する.

3.2　インストール

利用環境を構築し終えたら, 次はアプリのファイルをインストールする. 適当な場所にディレクトリを作成して付録Aに記載しているファイルを下記の手順に従ってインストールできる.

インストールの方法として以下に手順を記す.

1. RenTypeが置かれているGithubのページに行く[8].
2. 画面上部のSettingsの下にあるCodeボタンをクリックし, Localの中にあるDownload ZIPボタンをクリックする.
3. ZIPファイルを保存したい場所に全て展開する. RenType-mainというディレクトリが作成される.

あとはターミナル(コマンドプロンプト)を起動し, コピーしたディレクトリの中でpython(python3) main.pyと打ち込み実行することで, アプリが作動する.

3.3 RenTypeの機能

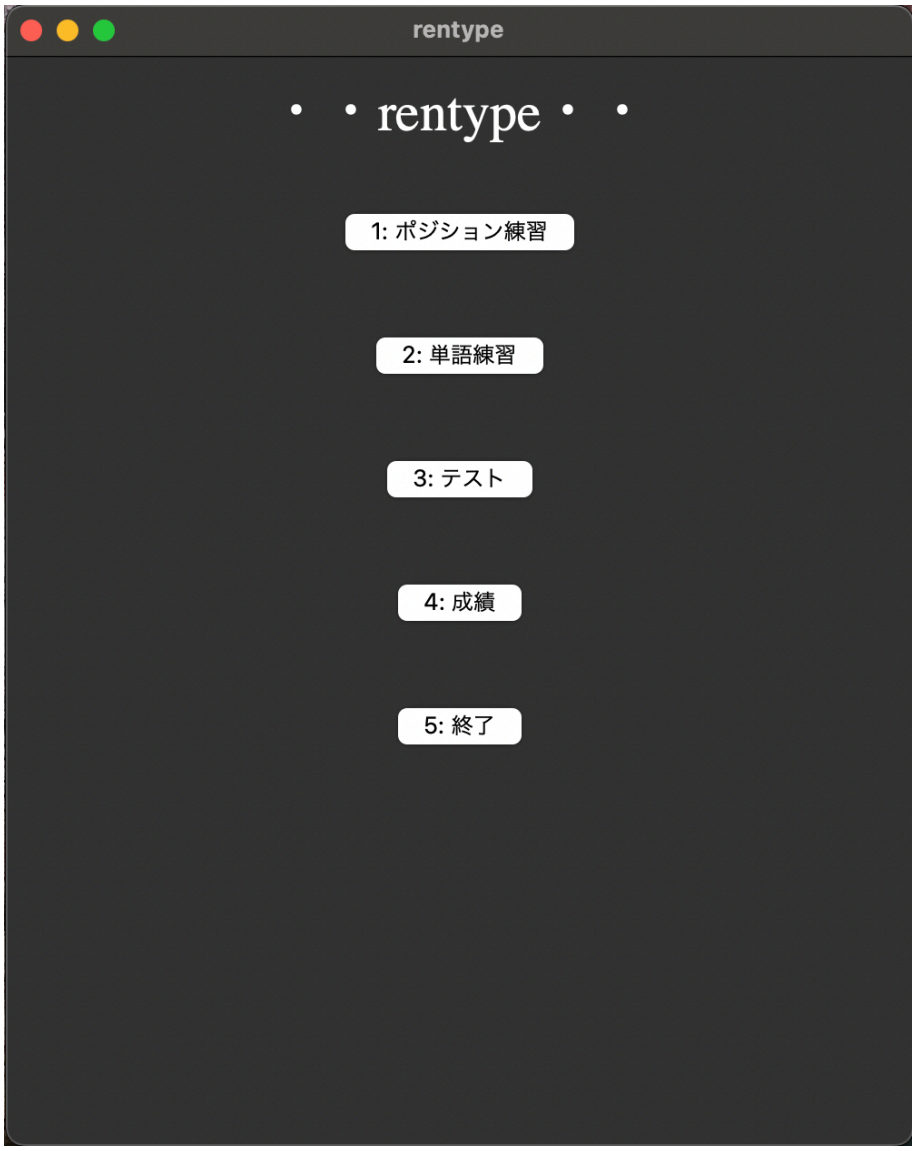

図 3.1: アプリのメイン画面.

実行して初めに表示される図 3.1は, アプリのメイン画面である. アプリのデザインは, ”誰がみても分かりやすい” ことを目的としたため, 背景やボタンをシンプルに設定している. シンプルなデザインにすることで外出先で周りの目を気にすることなく使用できる.

メイン画面には, [ポジション練習 ・ 単語練習 ・ テスト ・ 記録 ・ 終了] の 5 個のボタンが表示されている. RenType では, ボタンを押して行いたい練習を利用者が選択する仕組みである. 以下, 各ボタンを押した時の中身を説明していく.

3.3.1 ポジション練習

親指以外の各指１つずつの練習と, 全ての指での練習が可能となっている. 計 5 つの練習項目が存在し, 利用者が苦手な指を練習できる. MikaType のキーの段ごとの練習ではなく, Shunkuntype の練習方法と同じように指ごとの練習にしている.

以下は単語をランダムに表示させるコードと人差し指練習時の word1 の中身, それらの説明である.

```
1  words = hito_word.word1
2
3  def choise(self, words):
4    self.mojicount += 1
5    max = len(words) - 1
6    rnd = random.randint(0, max)
7    return words[rnd]

word1 = [
    'fgf',
    'frf',
    'ftf',
    'fvf',
    'fbf',
    'jhj',
    'juj',
    'jyj',
    'jnj',
    'jmj'
]
```

ここでは, words 配列の中に入っている単語をランダムに返している [6].

1 行目で hito_word.py の中にある word1 を取ってきてそれを words に代入している. word1 は上記にある文字が入った配列である.

2 行目でその words を引数として関数を実行している. 5 行目で配列の中に入っている文字の総数 (= max) を調べている. 6 行目で 0 から max までランダムに選んだ数字を rnd に代入する. そして return するときに words[rnd] とすることで, 配列の中身の文字をランダムに返している. choise 関数ではこのような流れが行われている. ポジション練習では時間制限を設けていない.

メイン画面でポジション練習を押すと画面には, [人差し指 ・ 中指 ・ 薬指 ・ 小指 ・ 全般 ・ 戻る] の 6 個のボタンが表示されている. それぞれの指で押すべきキーを練習できるため, キー配置を覚える練習法として有用であると考えられる.

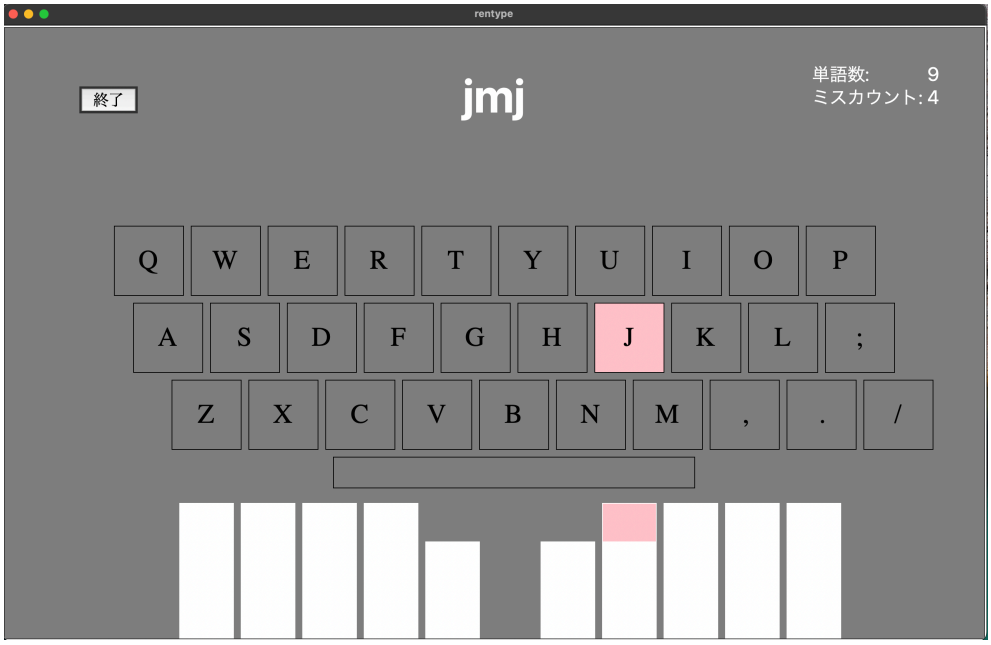

図 3.2: ポジション練習の実行画面.

図 3.2はポジション練習の実行画面である. 画面上部には打ってほしいキー文字を表示し, 入力したものが合っていれば 1 つずつ消えていく. 入力したものが間違えていれば, 文字が一瞬赤く光りまた元に戻る. 表示されている文字を打ち終えることができれば, 次の文字へと移っていく.

また, 画面下部には手を模した図形を作成した. これにより, 次に打つキーはどの指で押すべきであるかを画面上で確認できる. Canvas というウィジェットでキーボードや手の形を表現し, 打つべき指とキーを連動させるために tag を使用した. tag は図形につける目印のようなもので, 各図形に好きなタグをつけておく事で後から特定の図形に対して操作することが出来るようになる.

RenType では数字キーの部分を作成していない. 理由は, 数字キーは普段打つことが少なくセットポジションから遠い位置にあるため, キー配置を覚える上で優先度が低いからである.

以下は, キーを押した時の処理をするコードとその説明である.

```
 1  def keypush(self, event):
 2    global word
 3    key = event.char
 4    if word[0] == key:
 5      word = word[1:]
 6      self.delete()
 7      if len(word) == 0:
 8          word = self.choise(words)
 9          label2.config(text=self.mojicount)
10      else:
11        self.colorwarning()
12      self.kurikaesi()
```

3 行目で押されたキーの属性を取得している. 今回使用した char はキーの文字を取得するものである. 4 行目でその入力されたキーが, 表示されている単語の 1 文字目と一致しているかを判定をしている. もし一致していれば, 表示されている単語の頭の文字を削除して 1 つ右の文字を先頭にする処理を 5 行目で行っている. 6 行目で色をつけた図形を削除する関数を呼び出している. もし, 表示されている単語を打ち切ったら次の単語を表示させる処理を 7,8 行目でおこなっている. 9 行目は打ち切った単語数をカウントする処理である. 入力したキーが間違っていたら表示されている文字の色を変える関数を 11 行目で呼び出している. 12 行目はこの一連の処理を繰り返す関数を呼び出している. こうすることでループし永遠と続くようになっている.

それぞれ練習してみて「別の練習がしたい」となった時は, 画面左上の終了のボタンを押すことで終了でき, 練習時間を記録することが出来る.　終了ボタンを押さないと記録されないため, 注意が必要である.

3.3.2 単語練習

ある程度キー配置を覚えてきたら, 次は単語練習に移行すると良い. 単語練習のボタンを押すと画面には, [単語 ・ 戻る] の 2 個のボタンが表示されており, 単語のボタンを押すことで練習を行うことができる. ここでは 200 単語がランダムで表示されるため, 単語練習を開始して毎回同じものを打つということがない [7]. ここでもポジション練習と同じように終了ボタンを押して終了すると, 練習したことを記録する. プログラムコードはポジション練習と同じで, words 配列の中身を変えるだけで良いものにしている. ポジション練習の項目で紹介できていないコードを, この場所で説明していく. 以下は間違えたキーが押された時に実行される処理のコードとその説明である.

```
1  def colorwarning(self):
2    self.miscount += 1
3    label1.config(text=self.miscount)
4    canvas.itemconfig(text1, fill="red")
5    self.master.after(200, self.colorNormal)
6
7  def colorNormal(self):
8    canvas.itemconfig(text1, fill="white")
```

2 行目でミスタイプした数を数え, 3 行目で画面に表示されているミスタイプ数を変更している. また, 4 行目でタイピングする文字を赤く光らせる. このままでは文字が赤いままであるため, 5 行目で colorNomal 関数を呼び出し白色に戻している. ここの after メソッドの引数は, 第一引数が秒数で, 第二引数が呼び出す関数である. よって, 200ms(= 0.2 秒) 後に colorNomal を呼び出している. そうすることで, ミスタイプをした時に 0.2 秒だけ赤く光り, 0.2 秒経てば元の色に戻るようになっている. colorNomal 関数は 7,8 行目にあるものである.

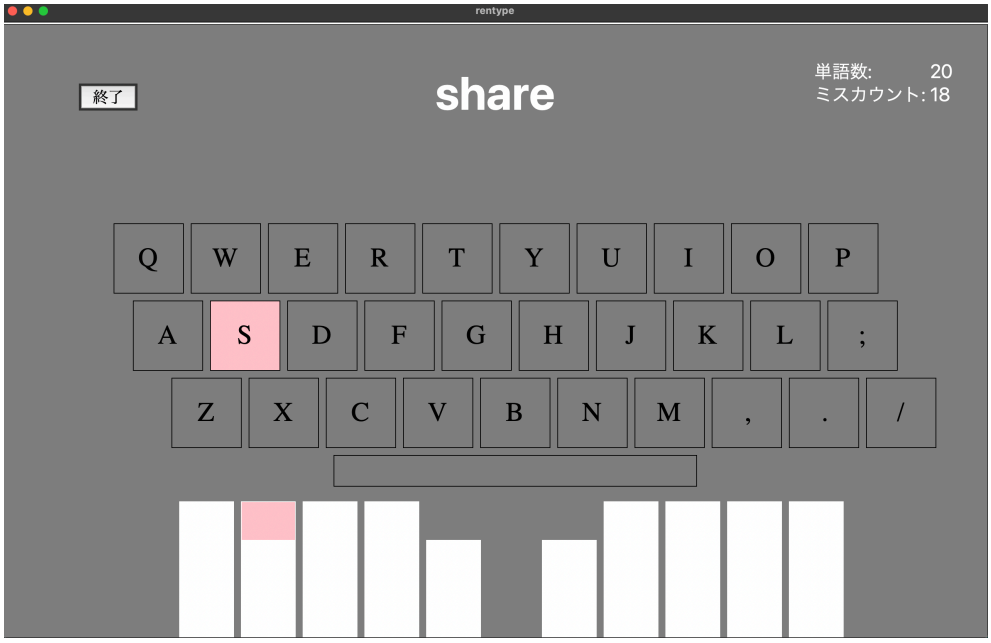

図 3.3: 単語練習の実行画面.

図 3.3は単語練習を行うときの実際のプレイ画面である. 表示される文字が単語に変化するとこのような形になる.

3.3.3 テスト

今自分がどのぐらいのタイピングスピードなのか計測したい時はテストのボタンを押す. テストのボタンを押すと画面には, [英単語 ・ 戻る] の 2 個のボタンが表示されており, 英単語のボタンを押すことでテストを行うことができる. 表示される単語は, 単語練習でも用いている 200 単語の中からランダムで表示される. Return キーを押してタイマースタートさせ, 20 単語を打ち切ることで自動的にタイマーストップする仕組みである. ここでも今までの練習同様, 終了ボタンを押すことで記録する.

以下はテストのコードの一部とその説明である.

```
1  key = event.char
2    if event.keysym == "Return":
3     self.start_time = time.time()
4     self.timer = self.master.after(INTERVAL, self.update_time)
5     self.mojicount = 0
6
7    if self.mojicount == 20:
8     self.master.after_cancel(self.timer)
9     self.t_end = time.time()
10    return
```

1 行目で押されたキーの値を受け取る. 押されたキーが Return ならタイマーをスタートさせる処理を 3,4 行目に書いている. 5 行目は途中で Return キーを押してタイマーが初めからになっても文字カウントがリセットされない問題があるため, 0 を代入して最初から数えるようにしている. カウントされている文字数が 20 を超えるとタイマーがストップする処理を 8 行目で行い, 9 行目でストップした時の現在時間を t_end に代入している. t_end から start_time を引くことで, テストに要した時間を記録している.

以下は経過時間を測るコードとその説明である.

```
1  INTERVAL=10
2    def update_time(self):
3      self.timer = self.master.after(INTERVAL, self.update_time)
4      now_time = time.time()
5      elap_time = now_time - self.start_time
6      elap_time2 = '{:.2f}'.format(elap_time)
7      self.label.config(text=elap_time2)
```

4 行目で now_time に現在の時間を取得したものを代入する. 5 行目で now_time から start_time を引いた値を elap_time に入れる. この時の start_time は Return キーが押されたときの時間が代入されている. 6 行目で elap_time を小数点 2 位までで表示するための処理をしており, 7 行目で画面に表示されている経過時間を変更している. この経過時間を更新していきたいため, 3 行目で画面に表示されている時間を 10ms ごとに更新する処理をしている.

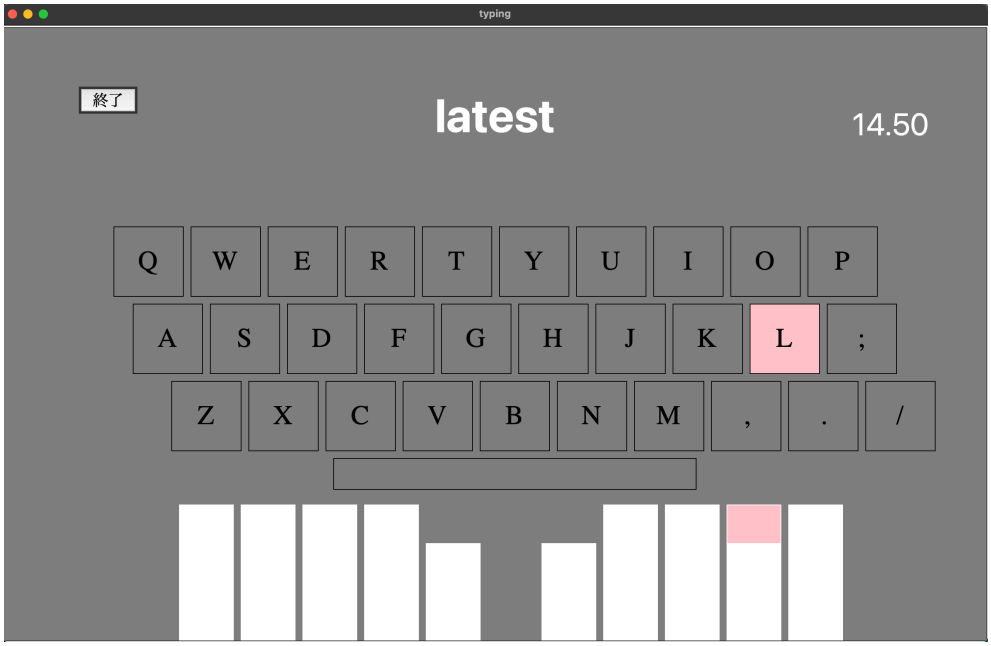

図 3.4: テストの実行画面.

図3.4はテストを行った時のプレイ画面である. 今のプログラムコードは, Return(Enter)キーを押してから時間の計測を開始する. 20 単語打ち終え, タイマーがストップしたら終了ボタンを押すことで記録が保存されるため, 誤操作でウィンドウ自体を消さないように注意が必要である.

3.3.4 記録

記録のボタンを押すことで, これまでの練習やテストの記録を見ることが出来る.

記録のボタンを押すと画面には, [練習記録 ・ スピード記録] の 2 個のボタンが表示されている. 練習記録は, ポジション練習と単語練習の 2 種類の練習時間の合計を目視するためのグラフを表示させる. そのため, 「これだけ練習した」という自信を自覚できる. スピード記録は, テストを行った時の最速タイムを更新していくさまをグラフで確認可能である. そのため, とても達成感を感じられるものとなっている.

記録の中身に, 少なくとも 2 日分の記録があれば線が横に伸びる. また, 記録については練習記録・スピード記録ともに 1 回行えばファイルが自動的に作成されるため, 利用者が新しくファイルを作成する必要はない.

ここで作成されるファイルは, 練習時間の記録のための　training_data.dat　と, テストの記録のための　test_data.dat　である.

以下は, 練習の合計を表示するグラフのコードとその説明である.

```
1   with open('./training_data.dat') as f:
2    lines = f.readlines()
3
4    fig, ax = plt.subplots()
5    fig.autofmt_xdate()
6
7    for line in lines[0:]:
8     sum += (int(line.split(' ')[9].replace('\n', '')))
9     if line.split(' ')[3].replace('\n', '') in X:
10      X.pop(-1)
11      Y.pop(-1)
12    X.append(line.split(' ')[0].replace('\n', ""))
13    plt.xticks(rotation=90)
14    Y.append(sum/24)
15    ax.set_xlabel('date')
16    ax.set_ylabel('min')
17    ax.plot(X, Y)
18    ax.grid(which='major', axis='both',color='#999999', linestyle='-.')
```

1 行目で training_data.dat を読み取る. (テストのグラフの場合は test_data.dat を読み取る.) 2 行目で 1 行ずつ lines という配列に代入している. 7〜14 行目で for 文の処理をしている. ここでは lines に入っている情報を 1 個ずつ取り出し, 練習した時間と日付を取り出している. 8 行目は練習した時間を sum に足していく処理である. ここで同じ日付が横軸に何個も並ぶことを避けるため, lines を見て既に横軸に入っている日付の場合, 被った日付を消す処理をしている. 13 行目では日付を縦で表示する処理をしている. 18 行目では grid を用いて線の色や, 線の種類を設定し見やすくしている.

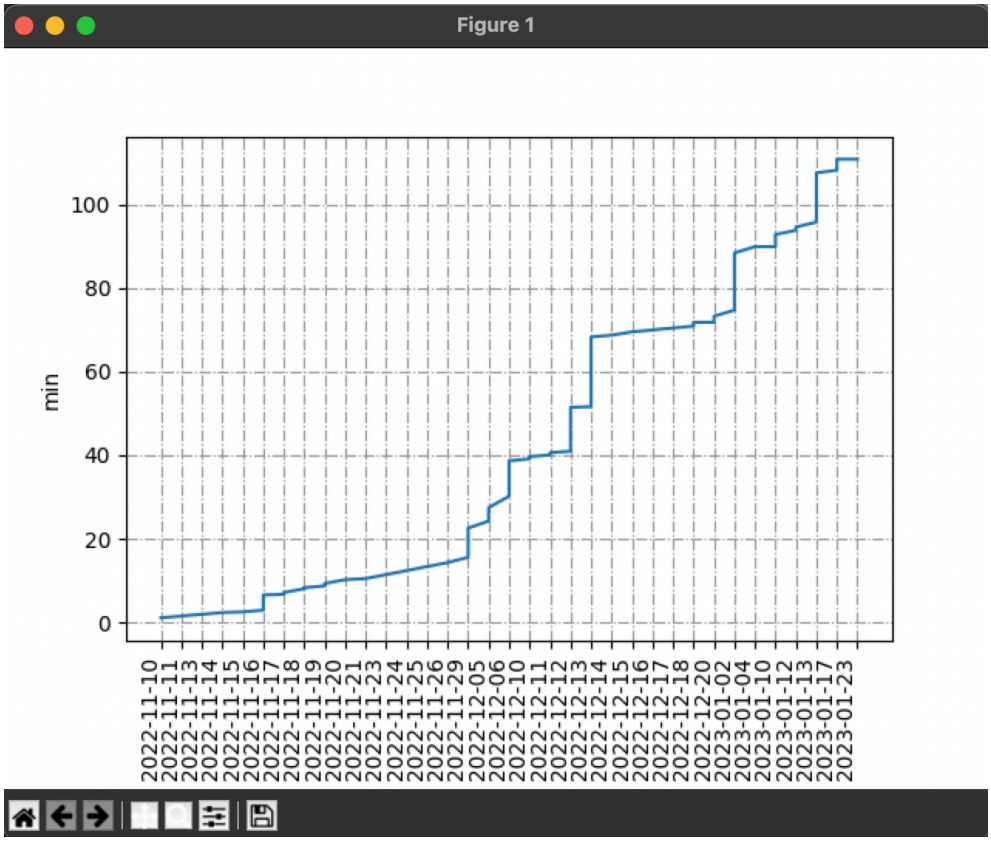

図 3.5: 練習記録のグラフ.

図 3.5は練習のグラフである. これまで練習してきた時間の合計を見ることができる. 縦軸は, 練習時間の合計値を分単位で表示している. 横軸は, 練習した日付を表示している. 同じ日付が並ぶことはなくその日にたくさん練習すれば, その日のグラフ線が縦に伸びるようになっている.

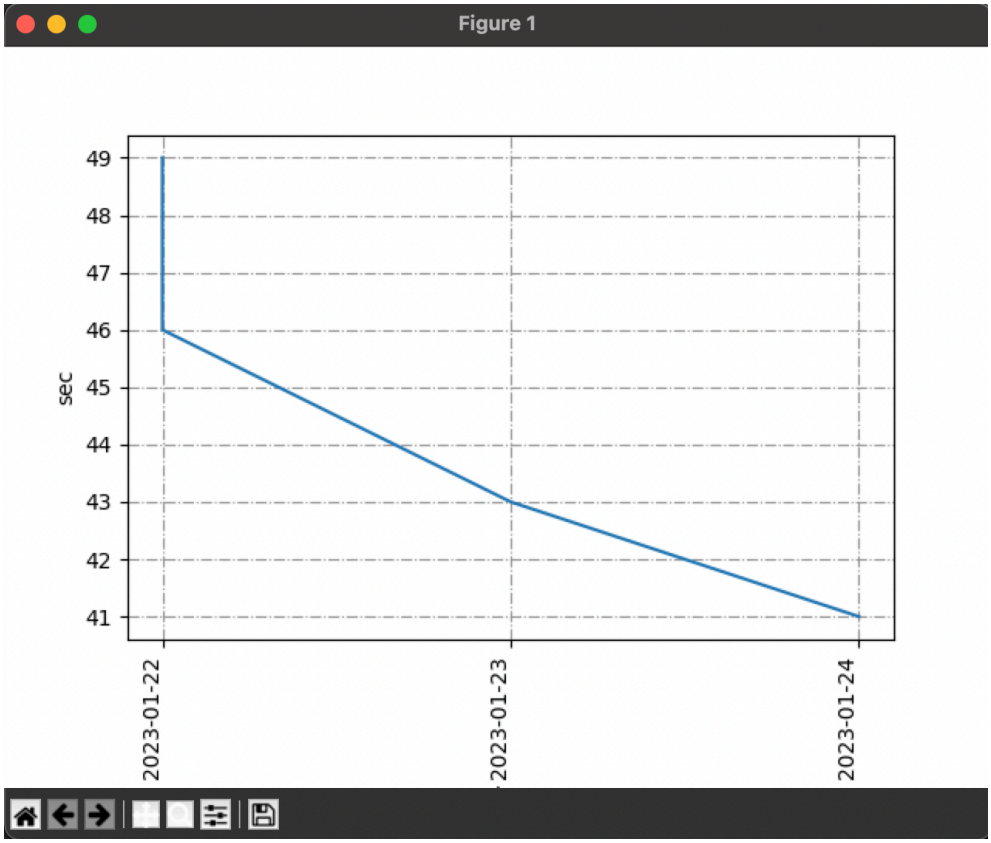

図 3.6: スピード記録のグラフ.

図 3.6はテストのグラフである. 縦軸は最速タイムの秒数である. 横軸は, テストをした日付である. テストのグラフは最速タイムを更新していく形にしている.

練習とテストで用いる単語は, プログラミング必須英単語 600+の中から 200 単語抜粋したものである. プログラミング必須英単語 600+は, コーパス言語学の手法で 1,200 万語を超えるプログラミング関連資料（ソースコード、API リファレンス、マニュアルなど）から英単語を選定したものである [7]. このリストはクリエイティブコモンズの「CC BY-NC-SA 4.0」でライセンスされており, 非商用利用に限って頒布が認められている.

3.3.5　終了

終了ボタンを押して表示しているウィンドウを閉じるとともに, 練習履歴を記録している. 終了ボタンを押さないと正しく保存されないから注意が必要である. アプリを終了したいときは, メイン画面の終了ボタンを押すか, ウィンドウを閉じることで終了できる.

以下のコードは，終了ボタンを押した時に行われる練習のデータを保存するコードとその説明である.

```
1  nowtime = datetime.datetime.now()
2  t_end = time.time()
3  t_result = int(t_end - self.t_start)
4  with open('training_data.dat', 'a')as f:
5    print(nowtime, '', nowtime.day, '', self.mojicount,
6      '', self.miscount, '', t_result, file=f)
7  self.master.destroy()
```

1 行目の nowtime には現在の時間を入れている. 3 行目の t_result には終了ボタンを押した時の時間から, 入力を開始した時間を引いた整数値を入れている. この値を用いてグラフを作成しているため, グラフの表示は整数となっている. 4 行目で training_data を開いて, 5〜6 行目でその中に print 文で記録している. 7 行目の self.master.destroy() により, ページを除去できる仕組みとなっている. この中の処理は, 他の練習やテストをした時も同じ動きである.

以上が RenType の機能と使用方法である.

第4章　まとめ

本研究では, Shunkuntype の練習順序にならって GUI のタイピングアプリ RenType を開発した. 構築には Python の標準ライブラリである Tkinter と, グラフを作成するために Matplotlib というライブラリを用いた. 3 つの主な目標であった, ランダムに単語を表示, 画面下部に手の形を模した図形を作成, 実力を簡単に測定できる機能の導入を達成できた. ポジション練習においてキーの段ごとの練習ではなく指ごとに練習でき, タッチタイピングを習得する効率が良くなった. これらより RenType は, 利用者にとってタッチタイピングを習得するための教材として有用なアプリとなったと考えている.

第5章　謝辞

本研究を進めるにあたり, 西谷滋人教授には指導教員として終始熱心なご指導を頂きました. 心から感謝いたします. また, Python による開発を進めていくうえで壁に突き当たった際, 終始温かいご助言を頂いた西谷研究室に所属している大学院 2 年生の堀川恭平氏にも大変お世話になりました. お礼申し上げます。

参考文献

[1] 美佳のタイプトレーナ, 2021, ”MikaType”, http://www.asahi-net.or.jp/~BG8J-IMMR/ (accessed on 10 Oct 2022).

[2] daddygongon, 2022, ”Shunkuntype”, https://github.com/daddygongon/shunkuntype, (accessed on 10 Oct 2022).

[3] Neutral, 2005, ”寿司打”, https://sushida.net, (accessed on 10 Oct 2022).

[4] ”Tkinter”, https://docs.python.org/ja/3/library/tkinter.html, (accessed on 10 Oct 2022).

[5] ”Matplotlib”, https://matplotlib.org, (accessed on 10 Oct 2022).

[6] HajimeteProgram, 2018, ”Python3+Tkinter でタイピングゲームを作ってみた”, https://hajimete-program.com/blog/2018/07/10/python3tkinter, (accessed on 10 Oct 2022).

[7] 合同会社グローバリゼーションデザイン研究所, 2021, ”プログラミング必須英単語 600+”, https://progeigo.org/learning/essential-words-600-plus, (accessed on 8 Dec 2022).

[8] 71211, ”RenType”, https://github.com/71211/RenType, (accessed on 12 Dec 2022).

付 録A　ファイル構成

メイン画面

main.py

ポジション練習

hito_typing.py

hito_word.py

naka_typing.py

naka_word.py

kusiri_typing.py

kusuri_word.py

ko_typing.py

ko_word.py

zen_typing.py

zen_word.py

単語練習

eng_typing.py

eng_word.py

テスト

test_typing.py

test_word.py

記録

graph.py

graph2.py

今はこれだけのファイルでアプリは構成されている．ポジション練習とテストを行い, 終了ボタンを押すと勝手に作られるファイルがある．それが, test_data.dat と training_data.dat

である．これらのファイルは練習やテストの記録を保存しているものである．これらのファイルを使ってグラフを表示しているため, 削除等しないように注意が必要である．