科目名	年度	レポート番号	クラス	学籍番号	名前
API 実習	2023	5	B	20122077	ROGER MARVIN

ページ数や文字数よりも、読んでわかりやすく書けているかどうかが、点数アップの分かれ目です。

API を使ったアプリやゲームが作ったけど「動きませんでした、完成しませんでした」は評価に値しません。 単位取得は、きちんと動くものが評価対象です。 API を使うこと、そしてプログラミングは 1 年生からの講義で学ぶことをすべて活用すれば実現できるはずです。

設問(1)

この科目で学んだ内容を第 3 者(他学部の学生や親など)にわかるように説明せよ。

この科目では API について学びました。API というのはデータベース（.db, .sqlite, .csv など）からデータをとって、あるエンドポイント（HTML、ゲームなど）に出すことです。仕組みは簡単にいうと、エンドポイント（例えば HTML）から何かをリクエストして、API がそこリクエストに基づいてデータベースから必要なデータを読み込んで、JSON としてレスポンスします。その JSON 形データが複数な人につかっるものなります。

API でホームページに表示することです。まず、データをリクエストして、JSON のレスポンスがもらいます。そこで、データを綺麗に表示するためには、その JSON を Javascript などで HTML 形にして、HTML のページに表示することです。例えば、ある WEB サイトにログインする時には入力したメールとパスワードを API でデータベースから同じのメールのパスワードを取って、入力したパスワードと比較します。もしも、正しければ、ログイン成功で、次のページ移動できます。もしも、間違ったら、404 のページに移動するか、再入力してくださいのメッセージが出すことです。というわけで、API はアプリやサービスのシステムを管理するためのものです。

設問(2)

レポート(4)をもとに、API 連携作成または API を用いたサービス開発結果を書いてください。何かしら動くものが出来ている前提です。

名称

Deadly Death Deadliest / デッドリー デス デッドリアス /「デデデ」

概要(作ったものの説明)

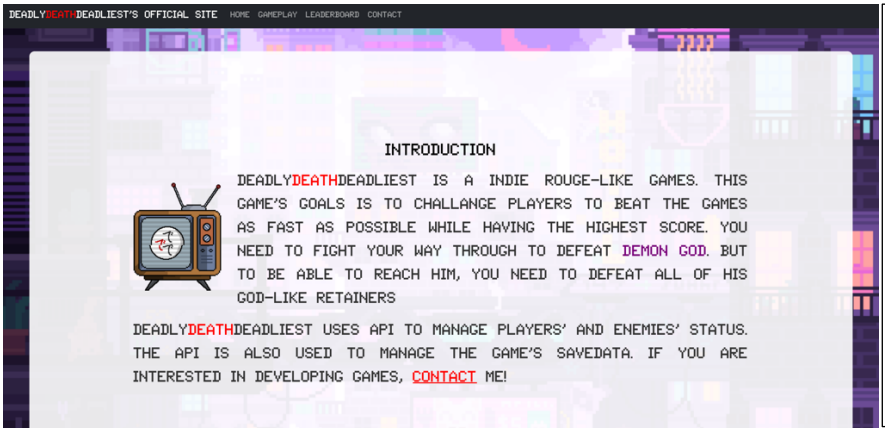
WEB ゲームです。このゲームはシングルプレイヤー・ローグライク・スピードラン・オクション・スラッシュゲームです。このゲームをやる目的はゲームは最短時間でゲームをクリアしながら、一番高いスコアを狙うことです。ゲームをクリアする度に、もしも、スコア一番高いとクリア時間一番早いだったら、ゲームの公式ウエブサイトにー自分の成果を表示します。

このゲームをアクセスためにはゲームの公式サイトにアクセスできます。そこで、自分または他のプレイヤーの達成を見え

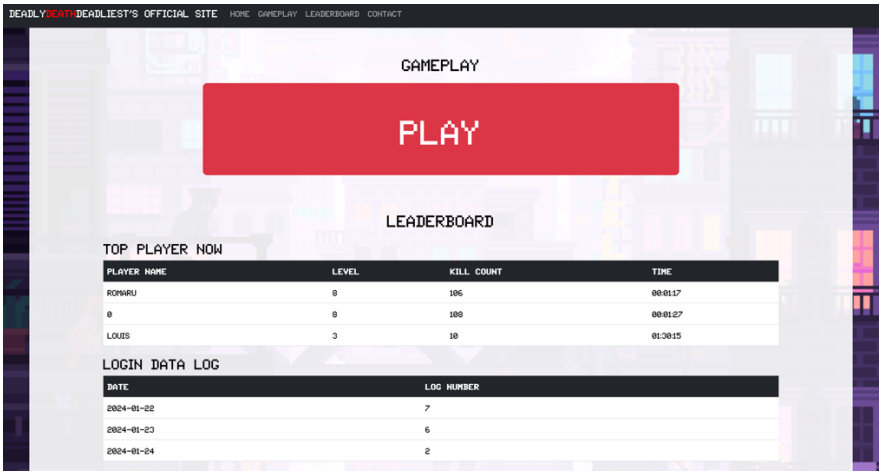
サービス説明(動作がわかるように画面を交えて説明すること)

この WEB ゲームは、ゲームの一覧を見えるし、プレイヤーがゲームをクリアしたプレイヤー名やスコアなどを他のユーザーに公開できて、TOP プレイヤーを 10 人までに表示できます。また、このゲームを 1 日何人ゲームをやるのかを計算して、表とグラフに出せます。

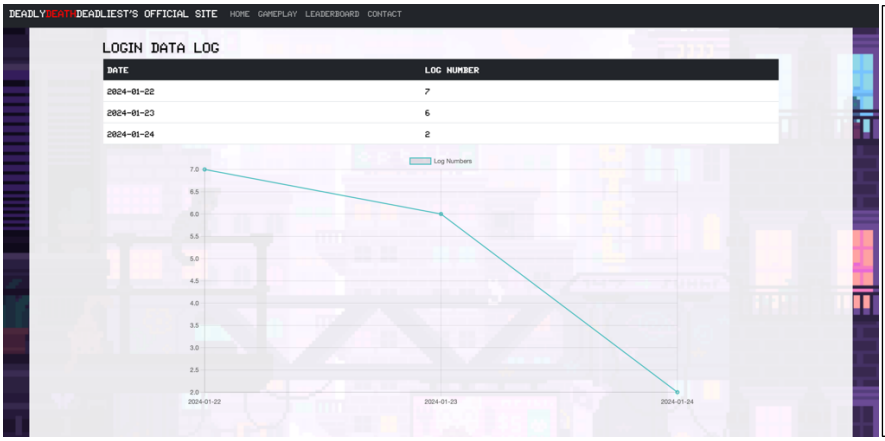
実際のものは以下の通りです。



こちらの部分はゲームのイントロダックションの部分です。ここでこのゲームはどんなゲームなのか、ゲームのストリーについてのものです。簡単に言うとあらすじです。



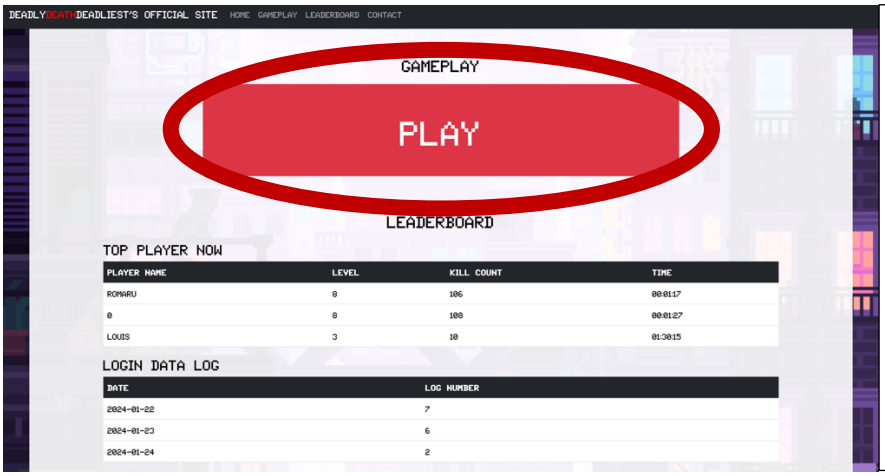
この部分はゲームを実際にやれるところです。この「PLAY」のボタンを押せば、HTML５式ゲームをやれます。
その下の部分は「Leaderboard」の部分です。この部分では、ゲームクリアするのプレイヤー名やスコアの表とこのゲームをやっている人数が１日に何人いるのかの表です。



この部分はゲームをやっている人数の表のデータをグラフ式に表示するものです。X 軸は人数で、Y 軸は日付です。
このグラフと表は十日間のデータを表示することです。

この部分は連絡の部分です。ここで、個人のメールアドレスとメッセージを送られ、開発者または管理者に直接にメールのやり取りができます。

ここは E mail.js API を利用します。



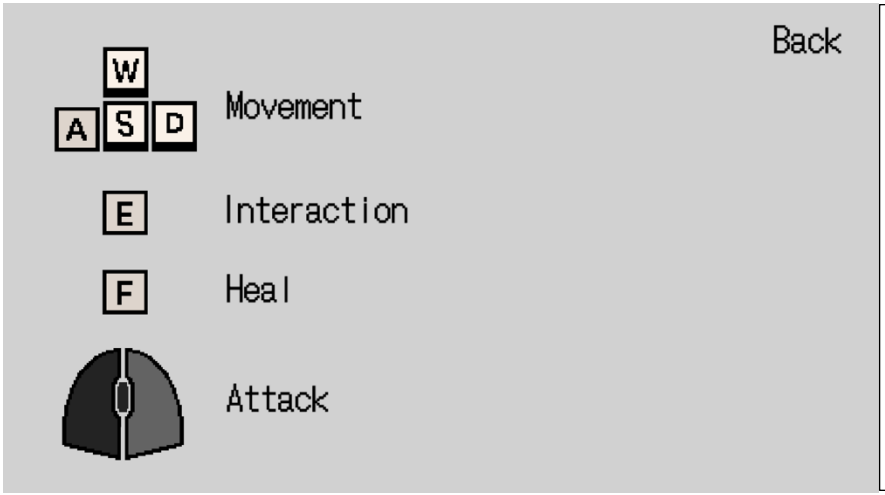
この「PLAY」のボタンを押せば、ゲームをできるようになります。

この「PLAY」のボタンを押せば、ゲームをできるようになります。

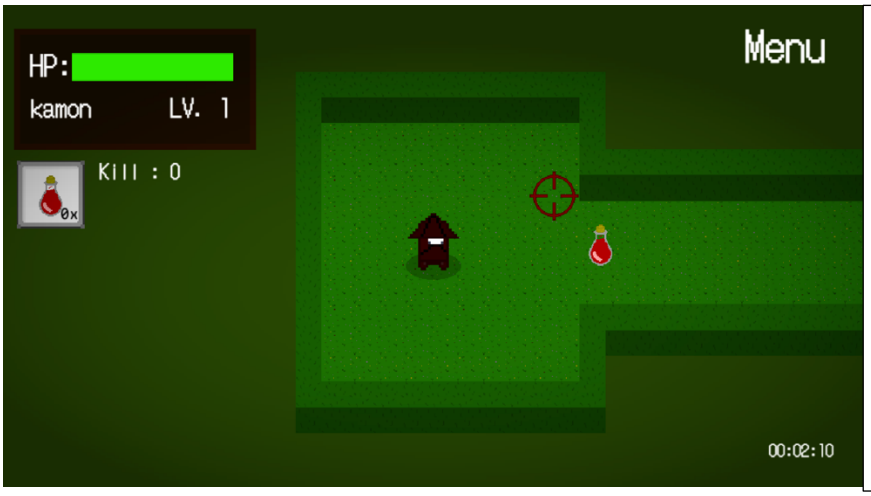
ゲームの説明

このゲームはシングルプレイヤーゲームで、パソコンしかできないゲームです。このゲームをアクセスために上のウエブサイトからアクセスできます。

このゲームをパソコンのゲームで、ゲームの操作はキーボードとマウスで動かせます。



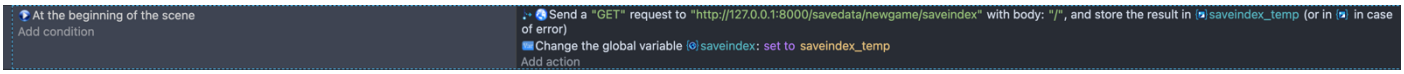
- 左の画像のように、キャラの動きは「ASWD」で動けます。
- ゲーム内の行為は「E」でできる。
  ＊行為→チェスト開ける；ドア開ける
- 回復アイテムを利用するのは「F」でできる
- 攻撃のアクションは「マウス左」でできる



- こちらはゲームの画面です。
- 見た通りに HP、レベル、回復アイテムの個数、スコア、時間、メニューボタンとターゲット Cursor が表示しています。これはどのステージでも、同じ形で表示します。

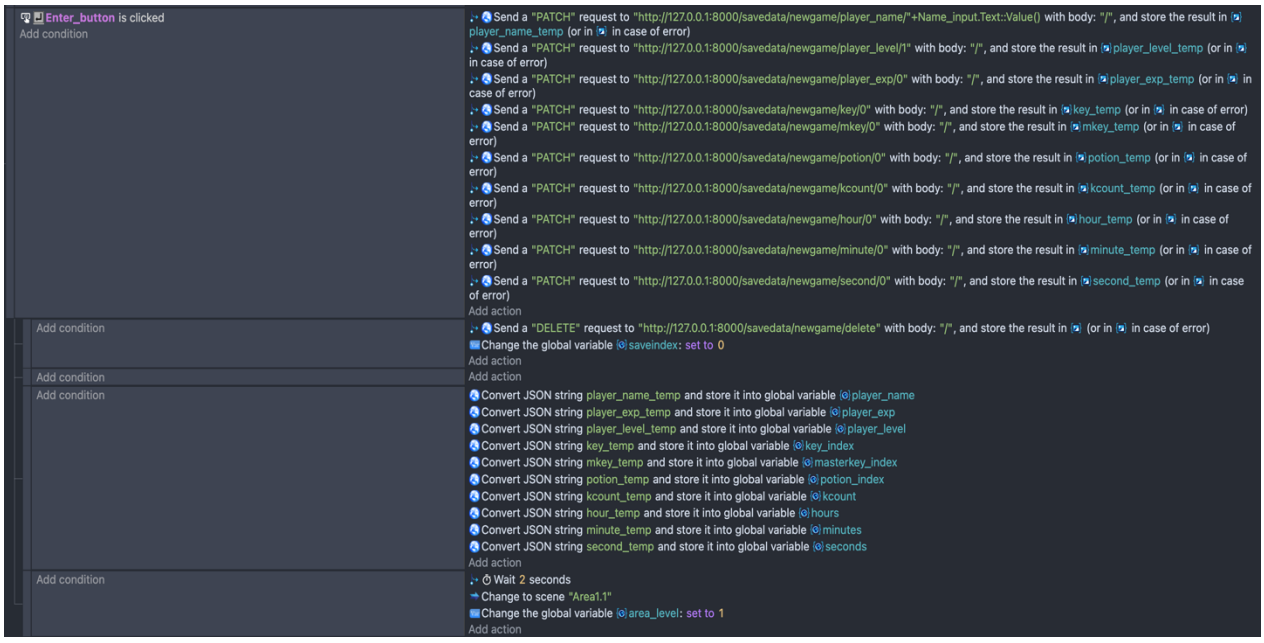
ゲームと API の関連の説明（API の動作）

1. ゲームを開始する際に、ゲームが API にサーブデータを要求する。



これは、セーブデータが存在しているかどうかを確認するためです。もしも、Saveindex が 0 ではないという場合は、セーブデータが存在していると分かって、「Continue」のボタンがクリックできるとなります。

2. 「New Game」を選択したら、プレイヤーの名前を入力ボックスがあります。それを入力して、ゲーム内の「Enter」ボタンをしたら、ゲームが開始します。NewGame が開始する際に、ゲームは API に“PATCH”と“DELETE”をします。

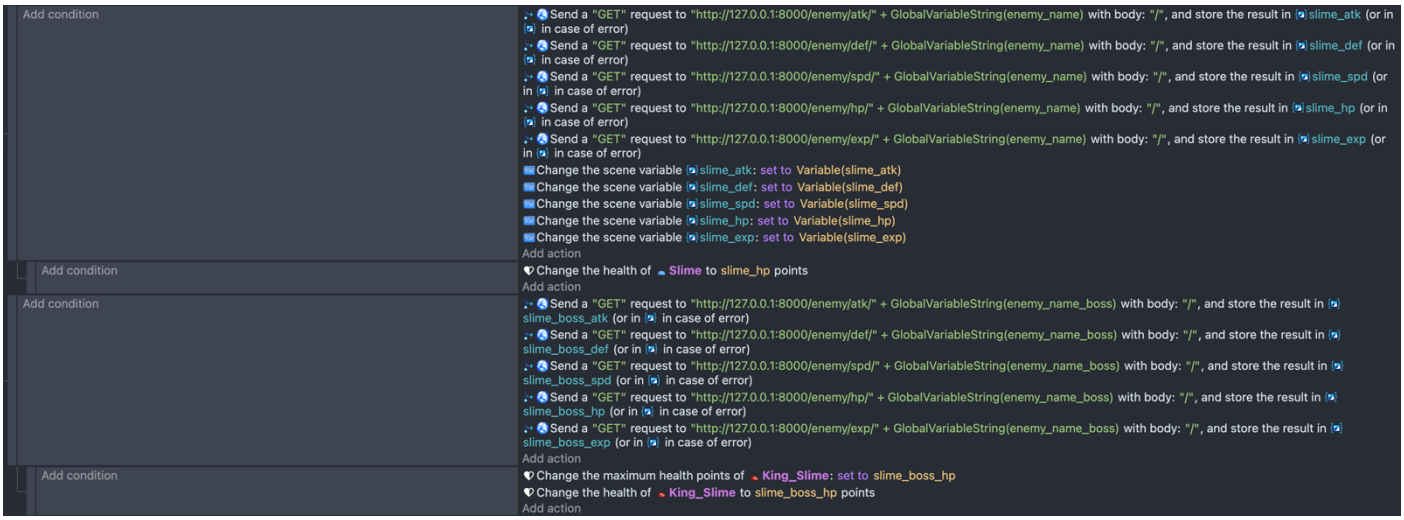


ここで Savedata.sqlite にあるの Id=0 の行を NewGame のデータに書き直すことです。そして、Id≠ 0 の行を全部削除することです。Id＝ 0 行のデータを書き直した後で、ゲームの変数に入れます。

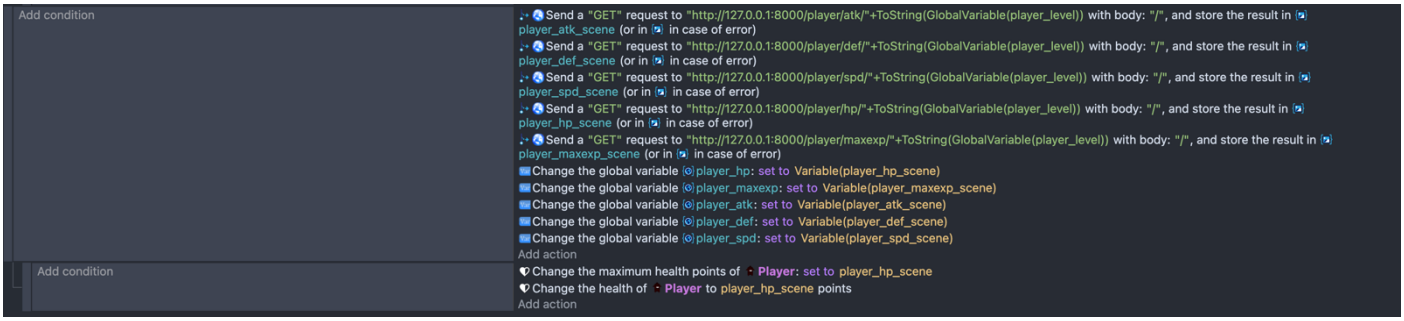
＊Gdevelop のエンジンが API に要求する機能があって、要求したものをある変数に入れる機能が持っています。例えば、上のように、要求するのレスポンス(上から 1 行目の player_name）を player_name_temp に入力することです。

データベースから削除した後、Saveindex を 0 にセットして、temp の変数をグローバル変数（よく使う変数）に入れることです。全てがおわたら、ゲームの Area1. 1 に移動して、ゲームを始めます。

3. ゲームが始まる時に、ゲームシステムが API にプレイヤーと敵のステータス情報を要求する。
   敵のデータ（ATK、DEF、SPD、HP、EXP）

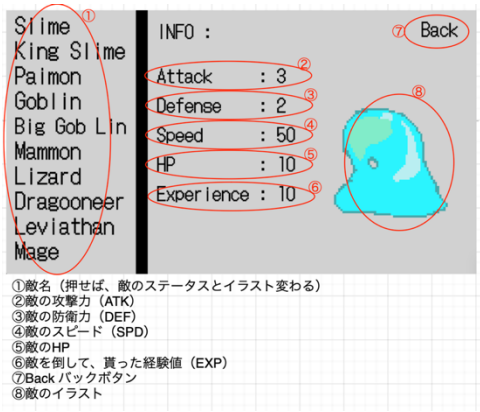


   プレイヤーのデータ（ATK、DEF、SPD、HP、MAX EXP）



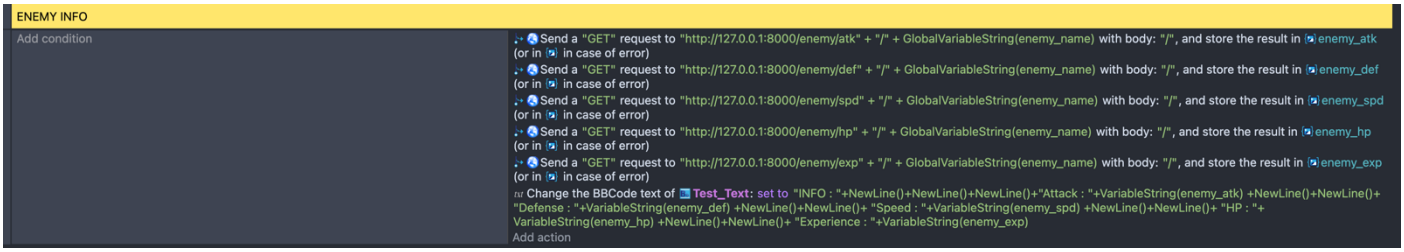
   敵の方で（enemy_name）と（enemy_name_boss）の変数があって、それをこのステージに登場する敵の名前です。その変数で、敵のステータスをゲットできます。プレイヤーも同じですが、名前だけではなく、レベルでステータスを要求することとなります。レベルによる、ステータスが違います。
   ステータスのデータをゲットした後、適切な変数に入力します。

4. 「Almanac」を押せば、敵の図鑑を見える。



   ここで、左の敵のリストの中のいずれかを押せば、敵の情報やイレストが表示します。
   敵の名前を押すたびに、ゲームシステムが API に要求します。

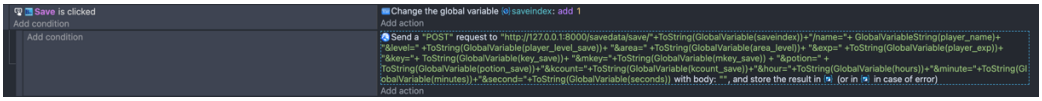
```python
#ENEMY API

@app.get("/enemy/{param}/{enemy_name}", response_model=Any)
def get_enemy_param(param: str, enemy_name: str):
    conn2 = db_connection("enemy.sqlite")
    cursor2 = conn2.cursor()

    cursor2.execute(f"SELECT {param} FROM enemy WHERE name=?", (enemy_name,))
    enemy = cursor2.fetchall()

    if not enemy:
        raise HTTPException(status_code=404, detail=f"ERROR")

    return enemy[0][0]
```



ここで、敵の名前を利用して、敵の情報を要求します。レスポンスを Test_Text に構造して、表示します。

5. 「Save」を押せば、ゲームセーブできます。

セーブボタンを押す際に、ゲームシステムが API に POST の要求をします。



POST する前に、saveindex を 1 プラスします。そして、データベースにデータを入れます。

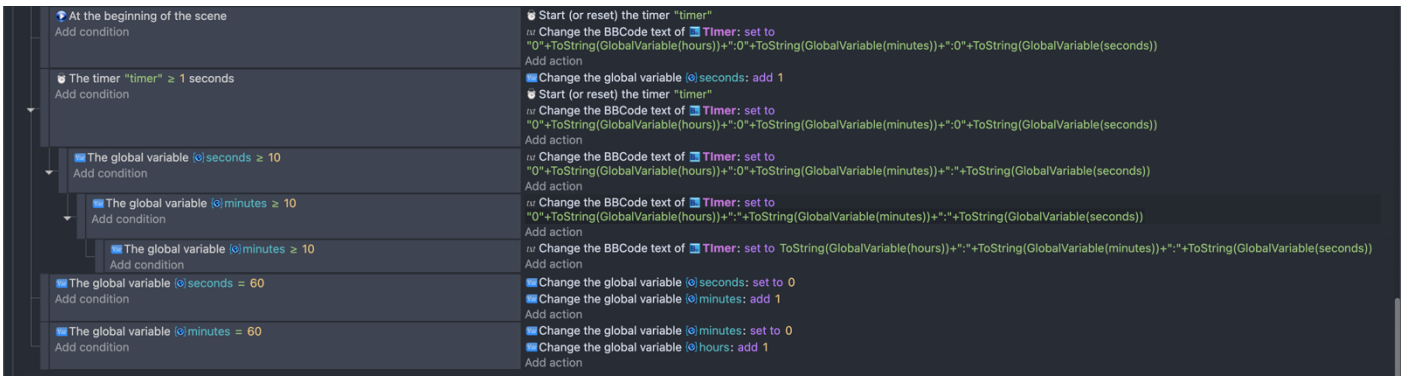
```
"http://127.0.0.1:8000/savedata/save/"+ToString(GlobalVariable(saveindex))+"/name="+ GlobalVariableString(player_name)+ "&level=" +ToString(GlobalVariable(player_level_save))+ "&area=" +ToString(GlobalVariable(area_level))+ "&exp=" +ToString(GlobalVariable(player_exp))+ "&key="+ ToString(GlobalVariable(key_save))+ "&mkey="+ToString(GlobalVariable(mkey_save)) + "&potion=" + ToString(GlobalVariable(potion_save))+"&kcount="+ToString(GlobalVariable(kcount_save))+"&hour="+ToString(GlobalVariable(hours))+"&minute="+ToString(GlobalVariable(minutes))+"&second="+ToString(GlobalVariable(seconds))
```

このクエリでの説明：

セーブする内容はプレイヤー名、プレイヤーレベル、エリアレベル、現在の EXP 数、インベントリのものの個数（鍵、ボス鍵、Potion）、キルカウント（敵を倒した数字）、時間の数字（Hour,Minute,Second）

ここでプレイヤーレベルとインベントリアイテムはゲームを登場する時に別の変数にセーブした変数を使っています。このゲームのセーブメカニズムは、ステージに登場したばかりの状態をセーブします。もしも、ステージの途中までにセーブしても、ゲームを「Continue」する際に、ステージの途中から続くではなくて、ステージの最初から続くこととなります。

6. 時間の動作は下のようにです。



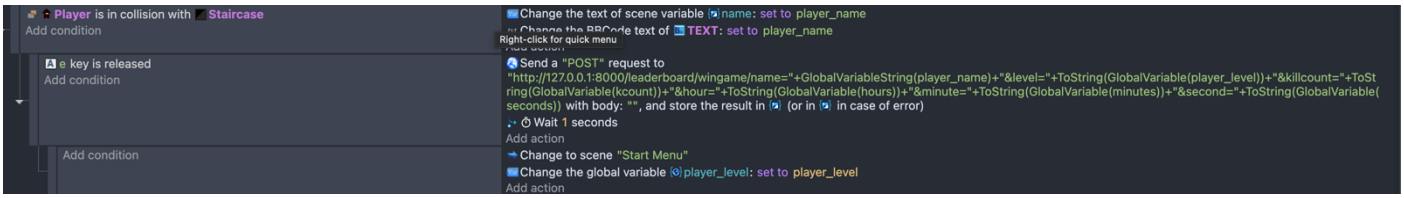
ゲームが始まる時に、savedata.sqlite からデータを取って、ゲームに表示します。

もしも、hours/minutes/seconds が 10<場合は、表示するときに”0”hour+”:0”+minute”:0”+seconds

10＞の場合は、hour+”:”+minute+”:”+second です。

普通の時間のように second が 60>なら、seconds を 0 にセットして、minutes+1 となる。Hour の場合も同じです。

7. Area1.3（現在のバージョン）をクリアできたら、ゲームシステムが API に“POST”の要求です。



このクエリは leaderboard.sqlite のデータベースにプレイヤーの情を POST することです。

POST するものはプレイヤー名、プレイヤーレベル、Kill Count、と時間です。

データを投稿する前に、API 先にそのプレイヤー名を確認します。もしも、存在している場合は、そのデータを書き直します。もしもなかったら、新しい投稿します。

＊しかし、投稿する際に条件があります。Kill Count とレベルの場合は、レベルと Kill Count の数値が高いの方を投稿します。逆に、時間は数値が低いの方を投稿します。これで、プレイヤーがゲームをチャンレンジできて、一番多いキルカウントを目指していながら、短時間でゲームをクリアします。

拡張性

この WEB ゲームの仕組みで、ゲームだけではなくて、色なサービスとして使えます。まず、この WEB ゲームが使った API がプレイヤーのスコアとゲームのログイン回数のデータを集取ことが一つの機能です。同じのような、仕組みで、例えば、教育用の WEB サイトが作れます。例えば、「PLAY」のボタンを押したら、ゲームの代わりに、テスト問題のアプリがでたら、その学生が問題を答えて、学生がよくできたか、あまりできないのかを集取したデータから自動分析できます。学生は自分の能力が見えるし、先生や教員などの担当の方々がデータを見て、この学生をどのように教えるのかをわかるようになります。

教育以外は、政府がアンケートなどのデータ収集時でも、このサービスの仕組みで、データ収集が早めにできるし、解凍する方々もいいビジュアルで楽しみながら答えます。もちろん、その時は、個人情報が関係ある可能性があるので、セキュリティのシステムをつけることが必要です。

レポート(4)の記載内容の実現状況（原則 100%となること）

内容の実現状況 99%です。ゲームのデザインが思ったものと違いましたから。