FLIGHT – SHOOTING　レポート

学籍番号：72206344
西 大成
2023/01/25

↓こちらからコードを取得できます（GitHub）
https://github.com/Taisei24/FlightShooting-by-Pyxel

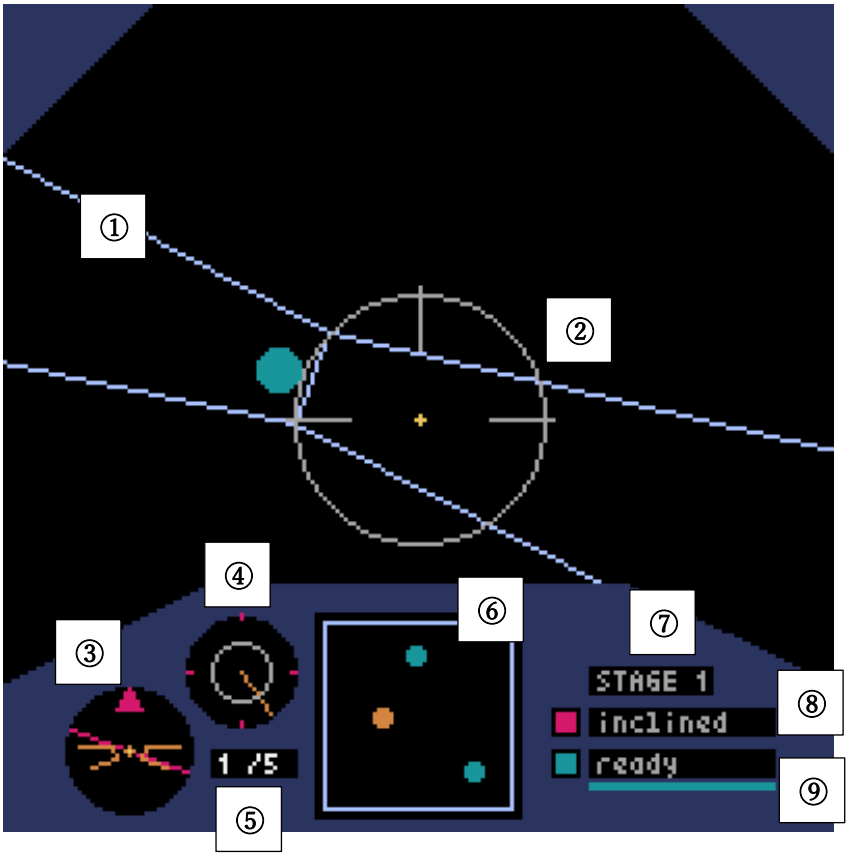
1. ゲーム概要

- 一人称視点で機体を操縦し、的を破壊するゲーム
- 全部で 3 つのステージがあり、的の個数や増え方が異なる
  - 1st stage は、的を 1 個破壊すると、新たに 2 個が生成され、5 個を破壊するとステージクリア
  - 2nd stage は、的を 1 個破壊すると、新たに 3 個が生成され、10 個を破壊するとステージクリア
  - 3rd stage は、的を 1 個破壊すると、新たに 4 個が生成され、20 個を破壊するとステージクリア
- エースコンバットシリーズを参考にしたが、真似た部分はほとんどない
  - エースコンバットはロックオン形式だが、本作はロックオンを採用していない
  - エースコンバットはレーダーを表示するが、本作はワールドマップを表示する
  - エースコンバットは機体の操縦が複雑だが、本作は左右キーのみで操縦する

2. 操作について

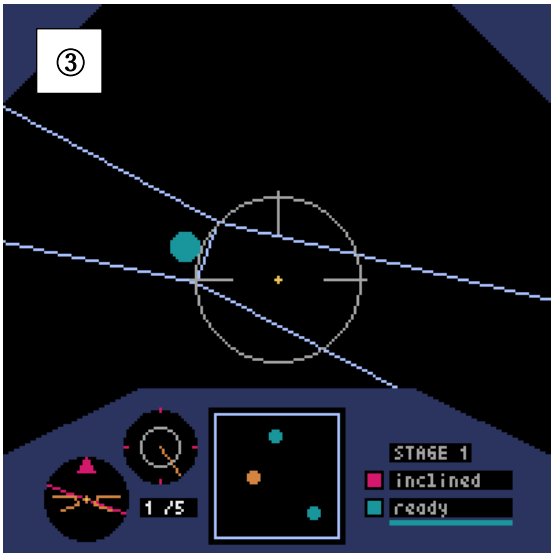
- 機体の操縦：左右キー
  - 左右キーを用いて、機体の傾き（バンク角）を操作する
- 弾丸の発射：スペースキー
  - 弾丸の発射には、3 つの制約がある
  - 1. チャージ
    - ✧　一度発射すると、次の発射までに時間がかかる
  - 2. 発射時の機体の水平
    - ✧　弾丸を発射するときは、機体が水平になっている必要がある
  - 3. 発射後の機体の水平
    - ✧　弾丸を発射した後は、しばらく機体の傾きがロックされる

3. 画面について



- コックピットから見えるワールド
  - ①：ワールド（壁・ターゲット・弾丸）が、一人称視点で描画される。
- コックピット内の情報
  - ②：照準器。射撃可能な状態であれば、照準円の内側に緑色の円が描画され、射撃可能であることを分かりやすくパイロットに伝える。また、壁やターゲットに近づきすぎると、警告“WARNING”が表示される。
  - ③：水準器。気体の傾きをパイロットに伝える。
  - ④：コンパス（方向指示器）。マップ上での機体の進行方向を示す。
  - ⑤：「すでに破壊したターゲットの個数／ステージクリアに必要なターゲット数」
  - ⑥：マップ。ワールド情報をパイロットに伝える。自分の機体は、黄色で描画される。
  - ⑦：ステージ情報
  - ⑧：機体が水平か否かを示す。傾いていれば“inclined”が表示され、赤色に点灯する。水平であれば“horizontal”が表示され、緑に点灯する。発射後、傾きがロックされている状態であれば“rocked”が表示され、黄色に点灯する。
  - ⑨：弾丸発射についてのチャージが完了しているかどうかを示す。チャージ中であれば“charging”が表示され、赤色に点灯する。完了していれば“ready”が表示され、緑に点灯する。発射中と発射後の数フレームは“shooting”が表示され、黄色に点灯する。さらに、文字の下側にチャージメーターがついており、チャージ状況を把握することができる。

4. プレイのシーケンスについて



- ①：スタート画面
  - スペースキーを押すと、ゲームが開始する。画面下側に、操作方法（USER GUIDE）が表示される。
- ②：ステージ開始時の遷移画面
  - ステージ開始前に表示される画面。18 フレーム後、自動でゲームが開始する。ステージ番号と、そのステージで倒すべきターゲットの個数が示される。
- ③：ゲーム画面
  - 機体を操縦し、弾丸を発射して、規定の個数のターゲットを破壊する。
- ④：ステージクリア画面
  - スペースキーを押すと、次のステージに移行する。その際、②の遷移画面も表示される。
- ⑤：ゲームクリア画面
  - スペースキーを押すと、最初からリスタートする。
- ※：ゲームオーバー画面
  - スペースキーを押すと、ゲームオーバーになったステージをリトライできる。

5. クリア条件

- STAGE 1
  - ターゲットを 5 個破壊する
- STAGE 2
  - ターゲットを 10 個破壊する
- STAGE 3
  - ターゲットを 20 個破壊する

6. ゲームオーバー条件

- 機体が壁にぶつかる
- 機体がターゲットにぶつかる

7. コードの解説

7.0. モジュールのインポート　　# line 1~2

- このゲームは、pyxel だけではなく numpy を必要とする

7.1. class Music():　　# line 4~192

- BGM と効果音を定義しているクラス
  - BGM は 3 つのチャンネルで演奏され、残り 1 つのチャンネルで効果音が演奏される
- BGM は、およそ 1 分でループする
  - ロックマン 2 の「おっくせんまん」と、UNDERTALE の「MEGALOVANIA」に触発され、作曲した
  - 曲全体の構成は、ロックマンに類似している。すなわち、①リズミカルなパート、②メロディアスなパート、③ベースがカッコいいパート、④移行局面、の 4 つが循環する
  - コード進行は、MEGALOVANIA に類似している。ベースラインが「A, G, F#, F」と下降していくなかで、同じメロディーフレーズが繰り返される。
- ゲーム中の効果音は、①弾丸発射、②ターゲット爆破、③機体激突（ゲームオーバー）、の 3 つを用意した。
  - ノイズ音源を効果的に使い、それっぽい音を再現
- ゲームクリアの効果音も用意した

7.2. class Start():　　# line 194~222

- このファイルが実行されたときに最初に呼び出されるクラス
- 画面を作ったり、音楽を定義したりなどの初期設定
- スペースキーを押すと、遷移画面を経て、ステージ 1 が始まる

7.3. class Game_Over():　# line 224~250

- ゲームオーバーのときに呼び出されるクラス
- スペースキーを押すと、ゲームオーバーになったステージをリトライする

7.4. class Next_Stage():　# line 252~280

- ステージ 1 またはステージ 2 をクリアしたときに呼び出されるクラス
- スペースキーを押すと、次のステージに進む

7.5. class Game_Clear():　# line 282~306

- ステージ 3 をクリアしたときに呼び出されるクラス
- ゲームクリアの効果音が演奏される
- スペースキーを押すと、ステージ 1 からリスタート

7.6. class Fighter():　　# line 308~509

- このゲームの心臓部
- 行列演算を繰り返し、一人称視点からのワールド（壁）を描画する
  - ➢ ① 二次元のワールドを、4 つの頂点で管理
    - ✧ ワールドの定義　　# line 314
  - ➢ ② そのワールドにおいて、機体が移動する
    - ✧ 機体情報の定義　　# line 310~313
      - ● tx：x 座標，ty：y 座標，dir：進行方向，bank：傾き
    - ✧ 機体の移動　　# line 323~351
      - ● プレイヤーが操作するのは、機体の傾き（bank）のみ。他はすべて従属変数となる
      - ● プレイヤーは機体の進行方向を直接操作できず、傾きを操作することによる間接的な操作になるため、「機体の重さ」が感じられる
  - ➢ ③ 機体の座標を原点に、機体の向きを x 軸の正として、4 つの頂点座標を変換
    - ✧ 変換行列を作成　　# line 356
    - ✧ ワールドの頂点座標を変換　　# line 361
  - ➢ ④ 一人称視点で、いくつの頂点が見えるのかで場合分け
    - ✧ 毎フレーム、リストを作成　　# line 322
    - ✧ それぞれの頂点の余弦を計算し、0.554 以上（片側視野 56.36 度，y=±1.5x に相当）であればリストに格納していく　　# line 362~369
    - ✧ 以下、そのリストの要素数によって場合分けを行う
      - ● 頂点が 0 個見える場合　　# line 371~389
      - ● 頂点が 1 個見える場合　　# line 393~425
      - ● 頂点が 2 個見える場合　　# line 427~448
      - ● 頂点が 3 個見える場合　　# line 450~470
  - ➢ ⑤ 二次元ワールドにおける一人称視点は、「奥行の情報を持った一次元」となる
  - ➢ ⑥ 奥行の情報から「高さ」を作り出し、一次元を二次元に拡張する
    - ✧ 機体からの「奥行」と、機体から見える「高さ」が反比例するように計算
  - ➢ ⑦ 新たな行列を作成し、スクリーン上の頂点座標（原点中心・無回転）を格納していく
    - ✧ “self.scr” という行列をフレームごとに作成し、計算結果を格納していく
      - ● フレームごとに作成する理由は、フレームごとに表示する頂点の数が異なるため
  - ➢ ⑧ 機体の傾きに合わせた回転行列をかける　　# line 472~475
  - ➢ ⑨ スクリーンの左上が原点になるように平行移動させる　　# line 476
  - ➢ ⑩ 頂点の計算結果を行列から取り出し、直線を描いていく　　# line 479~509

7.7. class Bullet():　　# line 512~542

- 弾丸の情報を管理するクラス
- インスタンスは常に 12 個存在し、“self.flag” によって存在するかどうかが判別される
- 弾丸はワールド座標上を移動し、それが壁に接触すると消滅する

- また、ターゲットに接触すると、弾丸は消滅し、ターゲットは爆発を開始する

7.8. class Target():　　# line 544~621

- ターゲットの情報を管理するクラス
- class Fighter の座標変換行列を流用して、ターゲットの座標を、ワールドの絶対座標から、機体を原点とした相対座標に変換する
- ターゲットの「奥行」の情報から、視点におけるターゲットの大きさを計算して描画する
- 弾丸が接触すると、爆発フェーズに入る
- コックピットから見えないターゲットについては、画面外に描画される
- 爆発局面において、新たにターゲットが生成される
  - def build(self, App):　　# line 606~621
  - ステージによって、異なる個数が生成される　　# line 581~595
  - 関数の再帰呼び出しによって、「座標のランダム性」と「機体との非接触」を両立

7.9. class App():　　# line 624~809

- 実際にゲームを動かすクラス
- インスタンス生成時に、ステージ番号“stage_num”を引数として受け取り、ステージに応じてゲームが進行する
- 弾丸発射にかかる３つの制約（「2. 操作について」を参照）は、このクラスで管理される
- コックピットの計器類の描画は、すべてこのクラスで行われる
  - def draw_cockpit(self):　　# line 730~809