提出日　2022 年 1 月 27 日

プログラム言語 II 最終レポート

Pyxel を用いたシューティングゲームの開発

提出期限：2022 年 1 月 28 日

所属：情報工学科 4 年

学籍番号：18-429

名前：長谷川 駿一

目次

1. 作成したゲームの内容

1.1 ゲームの概要

個人で遊ぶことのできる，昔ながらのシューティングゲームである．画像及び動作の処理はすべて Python で実装し，それらの処理の根本は Python の「Pyxel」というゲーム作成用のライブラリが担っている．

今回のゲーム作成のコンセプトは，自分が幼少期にゲームセンターで遊んでいた戦闘機のゲームを再現することである．そのため，Pyxel の特徴であるドット絵を採用し，シンプルな操作のみで遊べるようなゲームに仕上がっている．

ちなみに，タイトルの「GOOD　FLY 2」は「快適な空旅」という意味が込められており，前作であろうと思われる「GOOD FLY 1」は特に存在しない．

1.2 動作環境

OS や PC のスペックに寄らず，以下の環境が PC にインストールされていれば実行可能であると思われる．

- Python の開発環境
- Python のライブラリ「Pyxel」

なお，実行は，「GOODFLY2.py」と「my_resource.pyxres」がおかれた場所にて，以下のコマンドを入力する．

```
$python GOODFLY2.py
```

1.3 遊び方

本ゲームは大きく分けて，①タイトル画面，②プレイ画面，③ゲームオーバー時の画面の 3 種類が存在する．

① タイトル画面

タイトル画面では，ゲームを開始するか，ゲームを終了するかの 2 択を聞かれるため，それぞれに合った以下のキーボードを押す．

- ゲーム開始：Space キー
- ゲーム終了：q キー

② プレイ画面

自機を操作し，次々に襲ってくる敵機の弾をよけながら敵機を撃墜する．途中に「自

機動作 UP」,「発射する弾の進む速さ UP」,「連射速度 UP」のスキルアイテムが上から流れてくる．これを拾うことにより，そのアイテムに見合ったスキルポイントを得ることができる．なお，明確なゲームクリアを設定していないため，敵機が発射した弾に一回被弾したらゲームオーバーとなる．したがって，画面右上に表示されるスコアで競ってもらう．

以下に，プレイ画面で操作するキーボードを示す．

- 自機の操作：十字キー
- 弾の発射　：e キー
- ゲーム終了：q キー

③　ゲームオーバー時の画面

敵機の弾に被弾しゲームオーバーになった場合，以下の 3 択を聞かれるため，それぞれに合った以下のキーボードを押す．

- タイトル画面へ：Enter キー
- リトライ　　：Space キー
- ゲーム終了　　：q キー

1.4　動作画面

図 1．タイトル画面

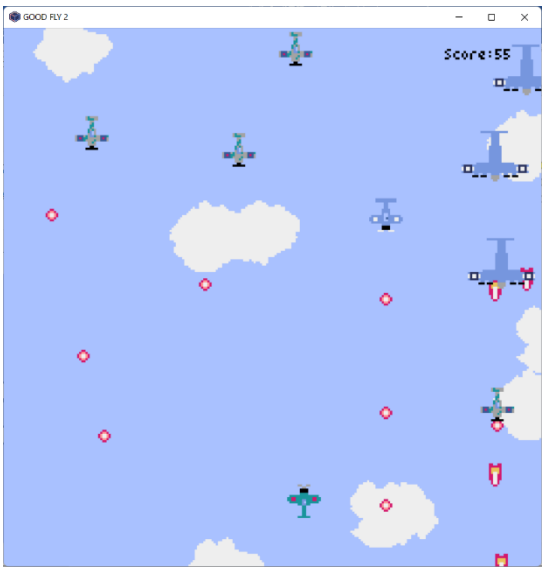


図 2．プレイ画面 その 1

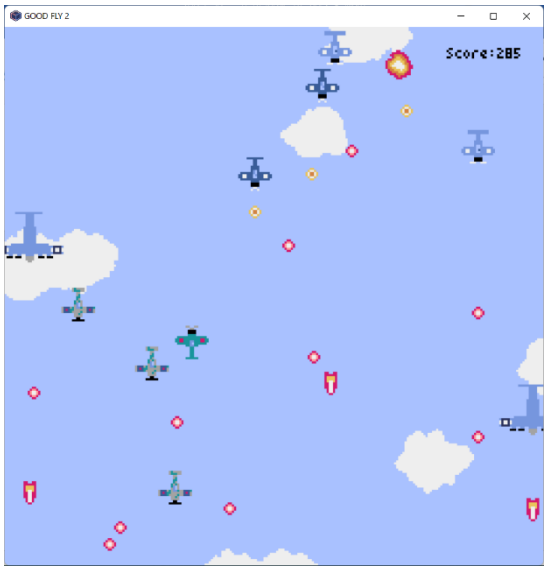


図 3. プレイ画面 その 2

図 4. ゲームオーバー画面

1.5 特徴・工夫した点

① キャラクターデザイン

本作品で登場している戦闘機及び弾などのデザインも，すべて自ら作成したものである．作成するにあたり，pyxel の描画ツールである「pyxeleditor」を使用した．

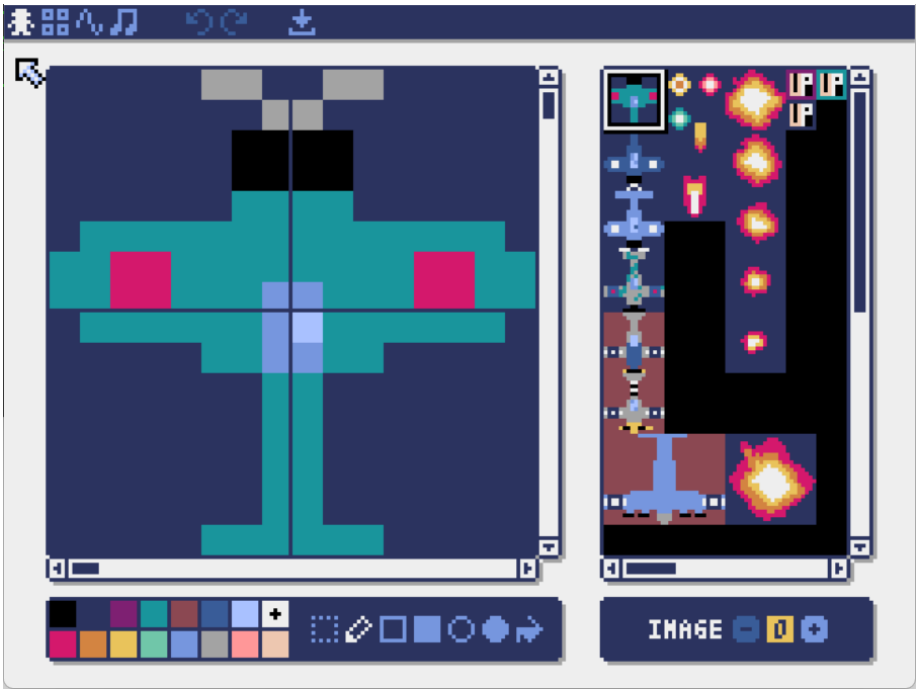


図 5. キャラクター描写画面の様子

②　雲及び爆破の描写

雲は 4 種類作成し，それぞれの雲をランダムに生成させ，画面の上から下に流れるように描写した．爆破は，小さい機体は 5 段階，大きい機体にはその 5 段階に加えもう 1 段階追加し，合計で 6 段階の描写を作成した．これらの工夫により，ゲームの本格性を高めることができた．

③　スキルアップアイテムの導入

「自機の動作速度」,「自機の発射する弾の進行速度」,「自機の発射する弾の連射速度」の 3 つの要点について，それらのスキルをアップさせるアイテムを導入した．スコアを 200 点獲得ごとに，3 種類のスキルアップアイテムのどれか一つをランダムに排出させ，敵機の弾と同様に上から流れてくる．スキルアップアイテムを獲得した自機は獲得したスキルアップアイテムに対応したスキルの速度を上げることができる．

④　背景の変更

タイトル画面で Tab キーを押すと，プレイ画面の背景を変更することができる．背景はデフォルトの背景を含め 3 種類用意されている．以下に図 1～4 で示した背景以外のものを示す．

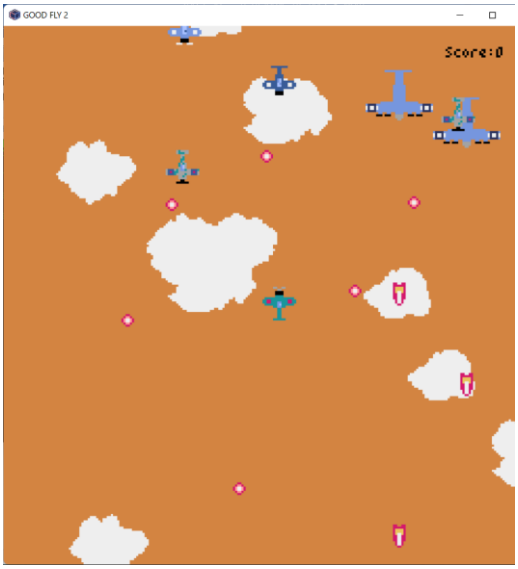


図 6．背景（その 2）

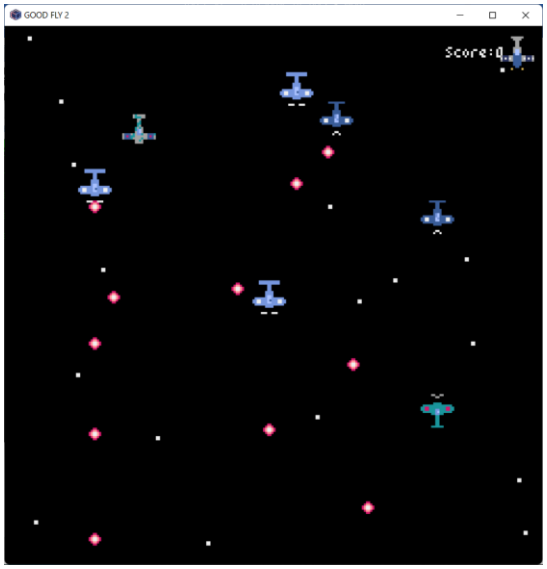


図 7．背景（その 3）

1.6　開発中に遭遇した問題と解決策

①　敵機の弾のデータ構造について

自機の当たり判定には，敵機の弾の座標の情報が必要不可欠である．この敵機の弾の情報を一つのグローバル変数であるリストによって実現することで，大量に生成される敵機がそれぞれ発射する弾の情報を，一元的に管理することができる．敵機によって弾のスピードが違うため，弾の座標を更新する際，そのグローバル変数の情報を更新する作業を実装することに苦労した．しかし，それぞれの敵機のクラスが持つ変数として，そのグローバル変数において，その敵機が発射した弾の情報に該当する場所の要素番号を格納したリストを追加し，その変数をもとに更新するようにした．

②　当たり判定における座標の設定

自機及び敵機の当たり判定において，弾が触れただけで撃墜した判定を下すのはあまりにも早く感じたので，±4 程度の座標の微調整を行った．本作品に最適な当たり判定を実装するために，今回のゲーム作成で最も多くの時間をかけた．

③　敵のクラスの継承について

本作品に登場する敵機の種類は全部で 6 種類ある．開発の初期段階では，基本のクラスをもとに継承を重ね，全部で 6 種類のクラスを作成していたが，コードが冗長になってしまったため，敵機のクラスが持つ変数に，1～6 の番号を振り当てる変数を追加し，その変数をもとに描写などの処理を施すことによって，クラスの数を 4 つに抑えることができた．

1.6　今後の課題

今後の課題において，以下の点が挙げられる．

- 本作品は 1 つのプログラムファイルで実現されている．そのため，プログラムの行数が 700 を超えてしまい，メンテナンス性に欠けるプログラムを作成してしまった．これを解決するために，Python のファイル分割について学習し，また，グローバル変数を極限まで減らす努力をする必要がある．

- 当たり判定の座標の処理，スキルアップアイテムにおける速度上昇の割合，敵の生成時間間隔などの値を，自分の主観によって設定してしまったため，プレイする人にとっては違和感を覚えることがあるかもしれない．厳密な計算のもと，それらの値を決定する必要がある．

1.7　プロジェクトマネジメントとして，企画・開発計画を見直したこと

①　企画を見直した点

- 開発当初はゴールを設定していたが，ゴールを撤廃し，敵の弾に被弾するまでプレイする仕様に変更した.理由として,時間の都合とゲームの単純性を追い求めた結果,敵機は 6 種類しか用意せず，この 6 種類のみでゴールを設定してしまうとチープな印象を与えてしまうと考えたためである.

  その分，開発当初の予定であるスキルアップアイテムを導入し，更に敵機の生成頻度を徐々に高めることによって，ゲームが単調になることを防いだ.

- リスボーン機能やその際の無敵時間の実装を考えていたが，その機能は実装しなかった.ゲームの単純性を考慮すると，一回当たったらゲームオーバーにしたほうがよいと考えたからである.

②　開発計画を見直した点

- ゲームオーバー時に描写される「GAMEOVER」の文字やタイトルの文字について，文字の大きさや縁なのを指定して描写してくれる関数が pyxel に存在しなかったため，一文字ずつ pyxeleditor で作成した．そのため，開発計画書には「ゲームオーバー及びタイトル画面の文字の描写」が計画されていなかったので，その項目が新たに追加された.

- 「他の敵機の描写」について，開発計画書には 2021/12/3~2021/12/5 の一か所しか記載されていなかったが，随時変更を加える可能性があることがわかった．また，他の描写についても 3,4 日の時間を設けていたが,あまり時間がかからないことがわかったので，描写については全期間を通じて随時変更するようにした.

③　全体を通じて

全体的に当初予定していた完成日よりも 1 か月ほど早く完成してしまった．これは良い傾向であると思えるが，反面，計画書どおりに遂行できていないことから，計画の精度について反省する必要がある.

今回のプロジェクトは自分一人で完成させたため計画どおりに遂行できなくてもある程度融通をきかせることができた．しかし，実際の業務は複数の開発者及び部門が携わることがほとんどであり，予算やステークホルダの管理などが必須になってくる.そのため，計画どおりに遂行することの大切さ及び計画の精度の向上を改めて考え直すことが重要であると思われた.

2. 週報

週報を以下にそのまま掲載する.

J4 プログラミング言語 II＆演習 III

開発工程　週 報

令和 3 年 10 月 4 日

/全 7 回の授業中

学籍番号: 18-429　氏名: 長谷川 駿一

開発作業内容
これまでに完成していること • 作成するゲームの大まかな概要が決定した. 今回作成するゲームの, 現段階の予定は以下の通りである. ・飛行機のシューティングゲーム. ・pyxel を使用したレトロな雰囲気を取り入れる.
反省点・問題点
python でのゲーム開発は初めてであるため, ゲーム用のライブラリを調査し, 実装するにはかなりの時間がかかると考えられる. 日々, 少しずつ自分のやりたいことに向けて, 調査していく必要がある.
来週の授業までにやっておくこと
• ゲーム画面の大きさや, プレイヤーの初期位置などを決定する.
来週の授業中にやること
• プレイヤーの操作ができるようにする.

完成度
1/100%

J4 プログラミング言語 II＆演習 III

開発工程　週 報

令和 3 年 10 月 7 日

/全 7 回の授業中

学籍番号: 18-429　氏名: 長谷川 駿一

開発作業内容
これまでに完成していること • pyxel を用いたゲーム開発における, ゲーム初期画面の設定 ・ウィンドウサイズの決定 ・プレイヤー位置の決定
反省点・問題点
まだ初期画面を作成しただけなので, これからも pyxel の調査を進めていきたい.
来週の授業までにやっておくこと
• プレイヤーをキーボード入力により動かせるようにする.
来週の授業中にやること
• プレイヤーの操作(弾の発射など)ができるようにする.
完成度
2/100%

J4 プログラミング言語 II＆演習 III

開発工程　週 報

令和 3 年 10 月 11 日

/全 7 回の授業中

学籍番号: 18-429　氏名: 長谷川 駿一

開発作業内容
これまでに完成していること • pyxel を用いたゲーム開発における，ゲーム初期画面の設定 ・ウィンドウサイズの決定 ・プレイヤー位置の決定 ・プレイヤーの画面描写
反省点・問題点
プレイヤーの描写と動きの速さなど，大まかなところは出来たので，プレイヤーの動きを突き詰めることが次回以降重要課題である.
来週の授業までにやっておくこと
• プレイヤーをキーボード入力により動かせるようにする.
来週の授業中にやること
• プレイヤーの操作（弾の発射など）ができるようにする. • 速さ UP のイベントを設ける.
完成度
2/100%

J4 プログラミング言語 II＆演習 III

開発工程　週 報

令和 3 年 10 月 13 日

/全 7 回の授業中

学籍番号: 18-429　氏名: 長谷川 駿一

開発作業内容
これまでに完成していること • pyxel を用いたゲーム開発における，ゲーム初期画面の設定 ・プレイヤー位置の決定 ・プレイヤーの画面描写 ・プレイヤーの弾の発射描写
反省点・問題点
プレイヤーの弾の描写において，メモリ領域を使用しすぎており，動きが緩慢になっているので，メモリ解消の方法を模索する必要がある.
来週の授業までにやっておくこと
• プレイヤーの弾の描写を完全に仕上げる.
来週の授業中にやること
• プレイヤーの操作（弾の発射など）を完璧にできるようにする. • 速さ UP のイベントを設ける.
完成度
3/100%

J4 プログラミング言語 II＆演習 III

開発工程　週 報

令和 3 年 10 月 18 日

/全 7 回の授業中

学籍番号: 18-429　氏名: 長谷川 駿一

開発作業内容
これまでに完成していること ・プレイヤー位置の決定 ・プレイヤーの画面描写 ・プレイヤーの弾の発射描写 ・プレイヤーの弾の発射速度の変更
反省点・問題点
弾を発射するときに人間のキーボードからの入力だと連射しすぎてしまうため， 連射速度を遅くする必要がある.
来週の授業までにやっておくこと
• プレイヤーの弾の発射間隔を設定する.
来週の授業中にやること
• 速さ UP のイベントを設ける. • 連射速度 UP のイベントを設ける.
完成度
3/100%

J4 プログラミング言語 II＆演習 III

開発工程　週 報

令和 3 年 10 月 20 日

/全 7 回の授業中

学籍番号: 18-429　氏名: 長谷川 駿一

開発作業内容
これまでに完成していること ・プレイヤーの弾の発射速度の変更 ・プレイヤーの弾の連射制御
反省点・問題点
連射の制御は一見できたように見えるが，実は間の弾を表示させていないだけであるため，当たり判定の設定を行う際に支障をきたさないかが心配である.
来週の授業までにやっておくこと
• 速さ UP のイベントを設ける.
来週の授業中にやること
• 連射速度 UP のイベントを設ける.
完成度
5/100%

J4 プログラミング言語 II＆演習 III

開発工程　週 報

令和 3 年 10 月 20 日

/全 7 回の授業中

学籍番号: 18-429　氏名: 長谷川 駿一

開発作業内容
これまでに完成していること ・プレイヤーの弾の発射速度の変更 ・プレイヤーの弾の連射制御 ・当たり判定の制御の途中
反省点・問題点
連射の制御は一見できたように見えるが，実は間の弾を表示させていないだけであるため，当たり判定の設定を行う際に支障をきたさないかが心配である.
来週の授業までにやっておくこと
• 速さ UP のイベントを設ける. • 当たり判定を完璧にする.
来週の授業中にやること
• 連射速度 UP のイベントを設ける.
完成度
5/100%

J4 プログラミング言語 II＆演習 III

開発工程　週 報

令和 3 年 10 月 27 日

/全 7 回の授業中

学籍番号: 18-429　氏名: 長谷川 駿一

開発作業内容
これまでに完成していること ・プレイヤーの弾の発射速度の変更 ・プレイヤーの弾の連射制御 ・敵の描写
反省点・問題点
当たり判定の前に，敵の描写を行うことにした. プレイヤーのクラスを継承するつもりであるが，うまくいくか心配である.
来週の授業までにやっておくこと
• 敵を描写させる.
来週の授業中にやること
• 敵が弾を発射できるようにする.
完成度
5/100%

J4 プログラミング言語 II＆演習 III

開発工程　週 報

令和 3 年 11 月 2 日

/全 7 回の授業中

学籍番号: 18-429　氏名: 長谷川 駿一

開発作業内容
これまでに完成していること ・プレイヤーの弾の発射速度の変更 ・プレイヤーの弾の連射制御 ・敵の描写
反省点・問題点
当たり判定の前に，敵の描写を行うことにした. 今現在，Python のクラスについて学習中である. 大体の継承および拡張機能を付けることができたので，次回は敵のキャラクターを作成する.
来週の授業までにやっておくこと
• 敵を描写させる.
来週の授業中にやること
• 敵が弾を発射できるようにする.
完成度
5/100%

J4 プログラミング言語 II＆演習 III

開発工程　週 報

令和 3 年 11 月 10 日

/全 7 回の授業中

学籍番号: 18-429　氏名: 長谷川 駿一

開発作業内容
これまでに完成していること ・プレイヤーの動き全般 ・敵の描写 ・敵の当たり判定
反省点・問題点
敵に対する当たり判定ができた. 次回から, 敵の弾の発射に取り掛かるが, 実際の時間を取得して制御しなければいけないため, 勉強していこうと思う.
来週の授業までにやっておくこと
• 時間に関する制御の仕方を学習する.
来週の授業中にやること
• 敵が弾を発射できるようにする.
完成度
10/100%

J4 プログラミング言語 II＆演習 III

開発工程　週 報

令和 3 年 11 月 15 日

/全 7 回の授業中

学籍番号: 18-429　氏名: 長谷川 駿一

開発作業内容
・敵及びプレイヤーの当たり判定. ・敵が弾を出す時間設定
反省点・問題点
敵に対する当たり判定ができたと思っていたが, 出力するとかなりずれが生じていることがわかった. 座標の取り方について, 再度検討する必要がある.
来週の授業までにやっておくこと
• 当たり判定の訂正
来週の授業中にやること
• 弾が当たった時の爆発のモーションや, プレイヤーに当たった場合のゲームオーバーの処理
完成度
12/100%

J4 プログラミング言語 II＆演習 III

開発工程　週 報

令和 3 年 11 月 17 日

/全 7 回の授業中

学籍番号: 18-429　氏名: 長谷川 駿一

開発作業内容
・敵及びプレイヤーの当たり判定. ・敵が弾を出す時間設定 ・各要素の変数化
反省点・問題点
前回問題となっていたプレイヤーの当たり判定を完璧に仕上げることができたが, それに伴い, プレイヤーの縦横の長さなどを変数に書き換える方が効率の良いことに気が付いた. 少し時間をロスしているように思える.
来週の授業までにやっておくこと
• 敵を x 軸方向だけでなく, y軸方向にも移動させる.
来週の授業中にやること
• 弾が当たった時の爆発のモーションや, プレイヤーに当たった場合のゲームオーバーの処理
完成度
12/100%

J4 プログラミング言語 II＆演習 III

開発工程　週 報

令和 3 年 11 月 21 日

/全 7 回の授業中

学籍番号: 18-429　氏名: 長谷川 駿一

開発作業内容
・敵及びプレイヤーの当たり判定. ・敵が弾を出す時間設定 ・各要素の変数化
反省点・問題点
敵を, y軸方向に移動させる処理の途中. 敵機がプレイヤーに接触した際の当たり判定, 及びフェードイン, フェードアウトの描写を考える必要がある.
来週の授業までにやっておくこと
• 敵を x 軸方向だけでなく, y軸方向にも移動させる.
来週の授業中にやること
• 弾が当たった時の爆発のモーションや, プレイヤーに当たった場合のゲームオーバーの処理
完成度
12/100%

J4 プログラミング言語 II＆演習 III

開発工程　週 報

令和 3 年 12 月 6 日

/全 7 回の授業中

学籍番号: 18-429　氏名: 長谷川 駿一

開発作業内容
・敵及びプレイヤーの当たり判定. ・敵が弾を出す時間設定 ・各要素の変数化
反省点・問題点
先週と同様のバグを修正中なので、以下に再掲する。 すべての敵機からの弾の当たり判定について、バグが見つかった。 この原因を調査する必要がある。 敵機がプレイヤーに接触した際の当たり判定を考える必要がある。
来週の授業までにやっておくこと
• 上記のバグの修正
来週の授業中にやること
• 弾が当たった時の爆発のモーションや, プレイヤーに当たった場合のゲームオーバーの処理
完成度
12/100%

J4 プログラミング言語 II＆演習 III

開発工程　週 報

令和 3 年 12 月 6 日

/全 7 回の授業中

学籍番号: 18-429　氏名: 長谷川 駿一

開発作業内容
・敵及びプレイヤーの当たり判定. ・敵が弾を出す時間設定 ・各要素の変数化
反省点・問題点
先週と同様のバグを修正中なので、以下に再掲する。 すべての敵機からの弾の当たり判定について、バグが見つかった。 この原因を調査する必要がある。 敵機がプレイヤーに接触した際の当たり判定を考える必要がある。
来週の授業までにやっておくこと
• 上記のバグの修正
来週の授業中にやること
• 弾が当たった時の爆発のモーションや, プレイヤーに当たった場合のゲームオーバーの処理
完成度
12/100%

J4 プログラミング言語 II＆演習 III

開発工程　週 報

令和 3 年 12 月 8 日

/全 7 回の授業中

学籍番号: 18-429　氏名: 長谷川 駿一

開発作業内容
・敵及びプレイヤーの当たり判定. ・敵が弾を出す時間設定 ・各要素の変数化
反省点・問題点
先週と同様のバグを修正中なので、以下に再掲する。 このバグの修正はかなり重要かつ大変な作業になると予測される。 すべての敵機からの弾の当たり判定について、バグが見つかった。 この原因を調査する必要がある。 敵機がプレイヤーに接触した際の当たり判定を考える必要がある。
来週の授業までにやっておくこと
• 上記のバグの修正
来週の授業中にやること
• 弾が当たった時の爆発のモーションや，プレイヤーに当たった場合のゲームオーバーの処理
完成度
12/100%

J4 プログラミング言語 II＆演習 III

開発工程　週 報

令和 3 年 12 月 13 日

/全 7 回の授業中

学籍番号: 18-429　氏名: 長谷川 駿一

開発作業内容
・敵及びプレイヤーの当たり判定. ・敵が弾を出す時間設定 ・各要素の変数化
反省点・問題点
敵と弾のフェードアウト時のオブジェクトの削除に成功した. 次回からは, プレイヤー及び敵の弾が当たった時の描写及び, 敵のオブジェクトの削除に取り組んでいく.
来週の授業までにやっておくこと
• 爆発の描写を作成しておく.
来週の授業中にやること
• 弾が当たった時の爆発のモーションや, プレイヤーに当たった場合のゲームオーバーの処理
完成度
20/100%

J4 プログラミング言語 II＆演習 III

開発工程　週 報

令和 3 年 12 月 20 日

/全 7 回の授業中

学籍番号: 18-429　氏名: 長谷川 駿一

開発作業内容
・新たな敵の描写
反省点・問題点
クラスを継承し，新たな動きをする敵を描写することができた. ゲームがあらかた完成してきたので，面白さの追求として，スピードアップやゴールの設定をすべて省き，ゲームオーバーまでスコアを加算していく仕様に変更する予定である.
来週の授業までにやっておくこと
• 雲の描写
来週の授業中にやること
• 最終調整
完成度
80/100%

J4 プログラミング言語 II＆演習 III

開発工程　週 報

令和 3 年 12 月 22 日

/全 7 回の授業中

開発作業内容
・新たな敵の描写 ・ゲームスタート，ゲームオーバーの描写
反省点・問題点
特になし．年内には完成しそう．
来週の授業までにやっておくこと
• 雲の描写
来週の授業中にやること
• 最終調整
完成度
80/100%

J4 プログラミング言語 II＆演習 III

開発工程　週 報

令和 4 年 1 月 12 日

/全 7 回の授業中

開発作業内容
・最終レポート作成 ・最終動作確認
反省点・問題点
大きな敵機の爆破の描写がずれていたため，そこを修正した．
来週の授業までにやっておくこと
• 最終レポートを終わらせる
来週の授業中にやること
• 最終レポート
完成度
95/100%

J4 プログラミング言語 II＆演習 III

開発工程　週 報

令和 4 年 1 月 17 日

/全 7 回の授業中

開発作業内容
・最終レポート作成 ・最終動作確認
反省点・問題点
特になし
来週の授業までにやっておくこと
• 最終レポートを終わらせる
来週の授業中にやること
• 最終レポート
完成度
96/100%

J4 プログラミング言語 II＆演習 III

開発工程　週 報

令和 4 年 1 月 19 日

/全 7 回の授業中

開発作業内容
・最終レポート作成 ・最終動作確認
反省点・問題点
特になし
来週の授業までにやっておくこと
• 最終レポートを終わらせる
来週の授業中にやること
• 最終レポート
完成度
96/100%

J4 プログラミング言語 II＆演習 III

開発工程　週 報

令和 4 年 1 月 24 日

/全 7 回の授業中

開発作業内容
・最終レポート作成 ・最終動作確認
反省点・問題点
特になし
来週の授業までにやっておくこと
• 最終レポートを終わらせる
来週の授業中にやること
• 最終レポート
完成度
98/100%

J4 プログラミング言語 II＆演習 III

開発工程　週 報

令和 4 年 1 月 26 日

/全 7 回の授業中

開発作業内容
・最終レポート作成 ・最終動作確認
反省点・問題点
最終レポートが完成した. 以上をもって，今回のゲーム作成のプロジェクトの作業を一旦終了する.
来週の授業までにやっておくこと
• 特になし
来週の授業中にやること
• 特になし
完成度
100/100%

3. 開発当初の企画書と開発計画書

開発当初の企画書と開発計画書をそのまま掲載する

2021 年度 J4 プログラミング言語＆演習 ゲーム製作 締切 2021 年 10 月 27 日（水）
企画書
以下のマスの大きさは変更しても構わない.

学籍番号 18-429　氏名 長谷川駿一

1. 作るゲームのジャンル

飛行機のシューティングゲーム

2. 作るゲームの内容

プレイヤーが自分の飛行機を操縦し，前から来る敵を打ち落としたり弾をよけたりして，ゴールまでたどり着く２D ゲーム.

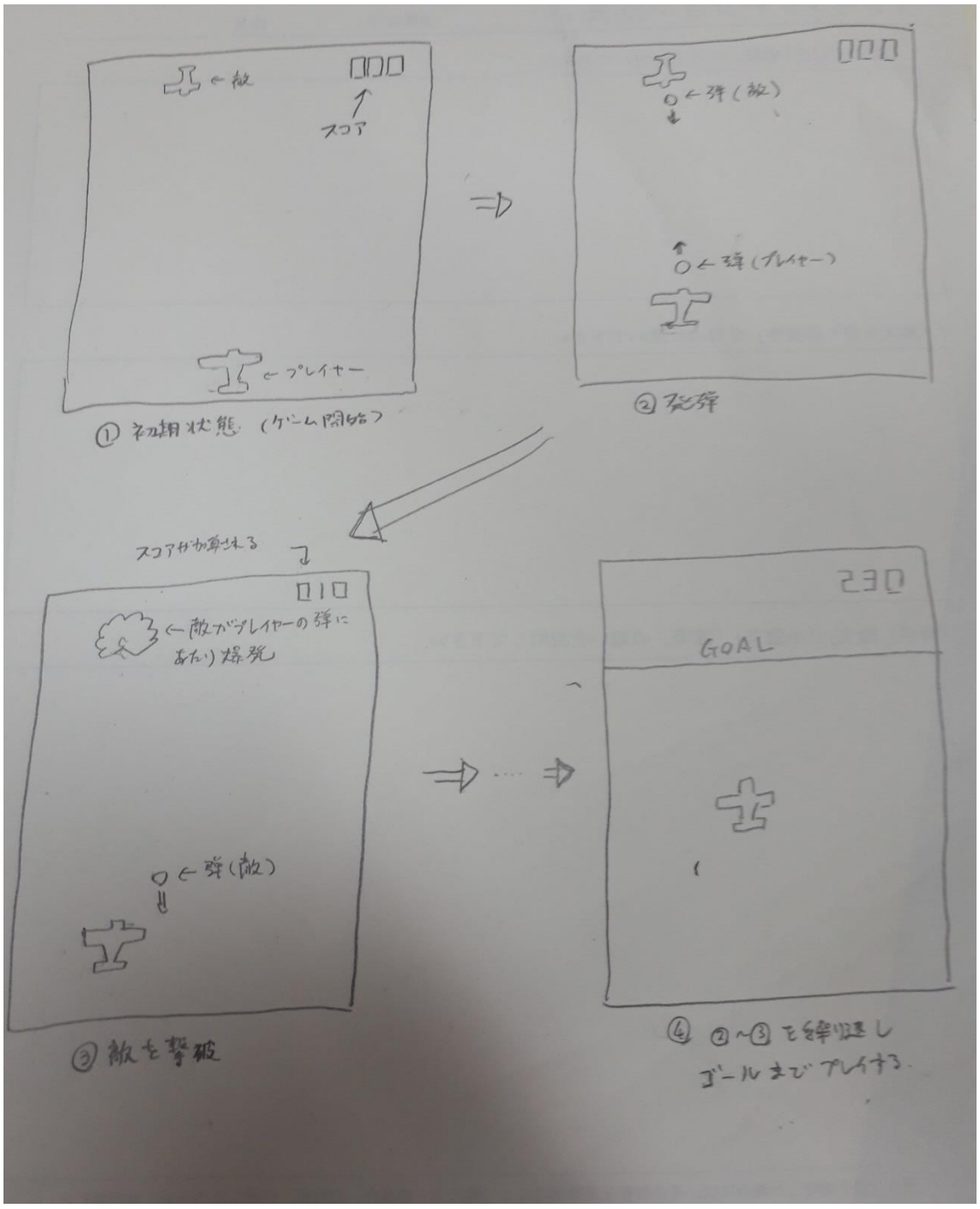
3. ゲームに必要な機能

● 敵を倒したときに加算するスコア表示 ● 「プレイヤーの動きの速さ」,「プレイヤーが放つ弾の速さ」,「弾の威力」など，それぞれのステータスを UP するためのイベント ● リスボーン機能やその際の無敵時間

4. 完成までにどのくらいの時間がかかると予想するか.

60 時間

5. ゲーム動作の主要な画面（4 コマ漫画などを使って説明する）



ゲーム作成で困難が予想される点．その理由やその解決方法．

● 敵の動きのプログラム（プレイヤーを追撃する敵や追撃の速さなど） →一つひとつの敵の特性を簡単に設定できるように，変数を多く使用する．

● あたり判定（相手の弾に“少し”当たった場合との対処の違い）
→プレイヤーと敵，及び敵の弾の座標を取得し，できるだけメモリを使用しない効率的なプログラムを作成する．
● 背景の描写及び爆発の描写
→専用のエディタを使用して，アニメーションとして描写できるように一つひとつ絵を作成し，描写する座標とその変化のルールを決めていく．

プロジェクト名	Pyxelでつくるシューティングゲーム	開始日	2021年11月27日
		終了日	2021年2月4日

作業名	開始日	終了日	11				12															
			27	28	29	30	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	
			土	日	月	火	水	木	金	土	日	月	火	水	木	金	土	日	月	火	水	
・敵のy軸方向の動作追加	11/27	11/30																				
・敵のランダム生成及び複数生成	12/1	12/15																				
・爆破の描写	12/3	12/5																				
・他の敵機の描写	12/3	12/5																				
・他の敵機の実装	12/16	12/22																				
・自機のスキルアップアイテムの描写	12/20	12/22																				
・スキルアップアイテムの実装	12/23	1/5																				
・ゲームオーバーの際の描写	1/1	1/5																				
・ゲームオーバー実装	1/5	1/17																				
・スコア及びゲームスタートの設定	1/18	1/24																				
・背景描写	1/21	1/24																				
・最終調整（主に描写）	1/25	2/3																				

	2021年11月27日
	2021年2月4日

																	1								
16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	1	2	3	4	5	6	7	8	9	
木	金	土	日	月	火	水	木	金	土	日	月	火	水	木	金	土	日	月	火	水	木	金	土	日	

																									2			
10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	1	2	3	4			
月	火	水	木	金	土	日	月	火	水	木	金	土	日	月	火	水	木	金	土	日	月	火	水	木	金			

4. 最終的に見直した企画書と開発計画書

最終段階の企画書と開発計画書をそのまま掲載する

2021 年度 J4 プログラミング言語＆演習 ゲーム製作 締切 2021 年 10 月 27 日（水）
企画書
以下のマスの大きさは変更しても構わない.

学籍番号 18-429　氏名 長谷川駿一

1. 作るゲームのジャンル

飛行機のシューティングゲーム

2. 作るゲームの内容

プレイヤーが自分の飛行機を操縦し，前から来る敵を打ち落としたり弾をよけたりして，ゴールまでたどり着く２D ゲーム.

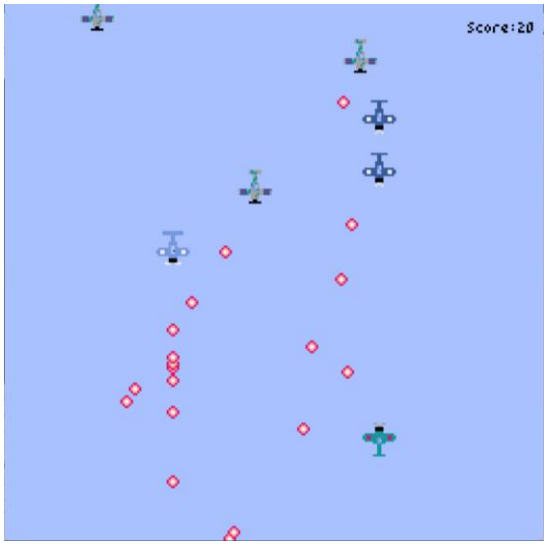
3. ゲームに必要な機能

● 敵を倒したときに加算するスコア表示 ● 「プレイヤーの動きの速さ」,「プレイヤーが放つ弾の速さ」,「弾の威力」など，それぞれのステータスを UP するためのイベント

4. 完成までにどのくらいの時間がかかると予想するか.

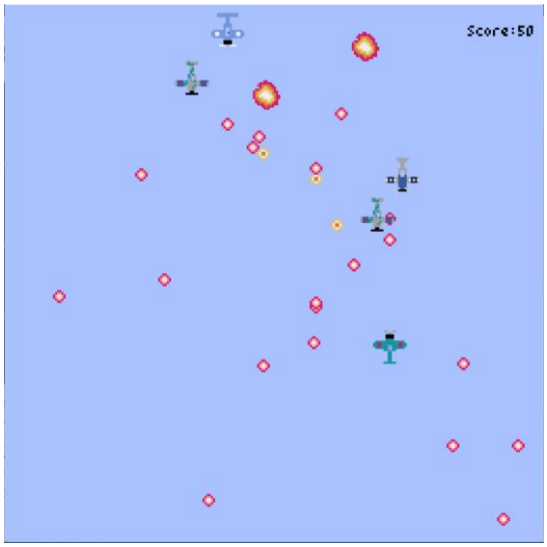
60 時間

5. ゲーム動作の主要な画面（4 コマ漫画などを使って説明する）



① 敵が上から流れてくるため，敵の弾に当たらないようによけながら発砲する．

③ ①②を繰り返し，自機が攻撃を受けたらゲームオーバーとなる．



② 発砲し敵に着弾したら，敵が爆破し，スコアが加算される．

※今回の本項目では，12 番目のプロトタイプのプログラムを実行し，その画面を使用した．

ゲーム作成で困難が予想される点．その理由やその解決方法．

- 敵の動きのプログラム（プレイヤーを追撃する敵や追撃の速さなど）

→一つひとつの敵の特性を簡単に設定できるように，変数を多く使用する．

- あたり判定（相手の弾に“少し”当たった場合との対処の違い）

→プレイヤーと敵，及び敵の弾の座標を取得し，できるだけメモリを使用しない効率的なプログラムを作成する．

- 背景の描写及び爆発の描写

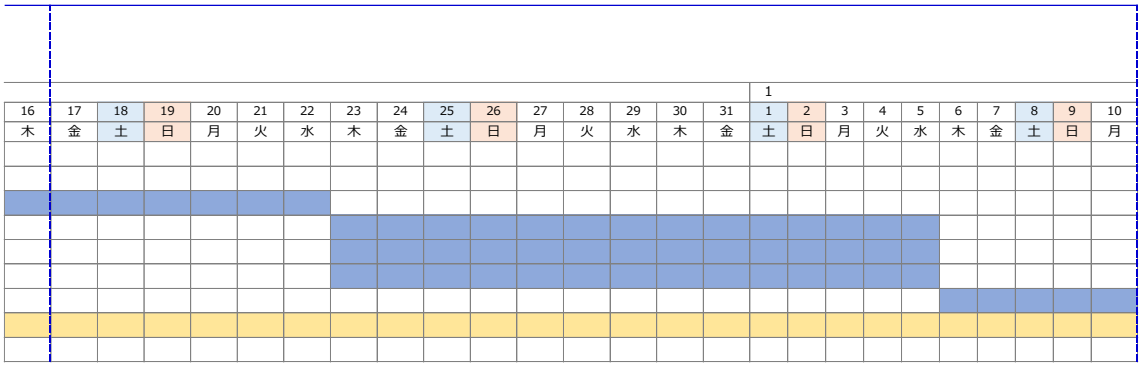
→専用のエディタを使用して，アニメーションとして描写できるように一つひとつ絵を作成し，描写する座標とその変化のルールを決めていく．

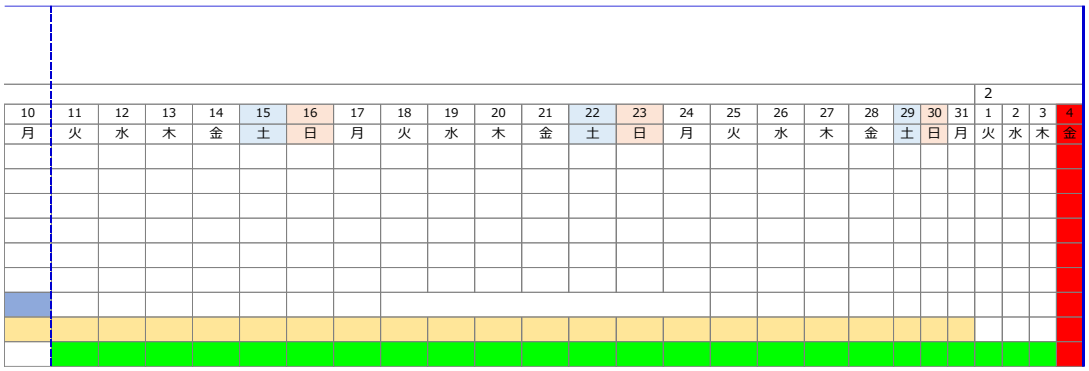
- タイトル画面及びゲームオーバー画面での処理

→専用のエディタを用いて一文字ずつ描写する．

プロジェクト名	Pyxelでつくるシューティングゲーム	開始日	2021年11月27日
		終了日	2021年2月4日

作業名	開始日	終了日	11				12																
			27	28	29	30	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15		
			土	日	月	火	水	木	金	土	日	月	火	水	木	金	土	日	月	火	水		
・敵のy軸方向の動作追加	11/27	11/30																					
・敵のランダム生成及び複数生成	12/1	12/15																					
・他の敵機の実装	12/16	12/22																					
・スキルアップアイテムの実装	12/23	1/5																					
・ゲームオーバー実装	12/23	1/5																					
・スコア及びゲームスタートの設定	12/23	1/5																					
・背景描写	1/6	1/10																					
・描写デザイン全般	11/27	1/31																					
・最終調整	1/11	2/3																					





5. プログラムリスト

以下に，今回作成したゲームのソースコードを掲載する．

```
# GOOD FLY 2


import pyxel

import random

import time


DISP_X = 256          # スクリーンの縦の長さ

DISP_Y = 256          # スクリーンの横の長さ

PLAYER_X = 0          # プレイヤーの x 座標

PLAYER_Y = 0          # プレイヤーの y 座標

PLAYER_SHOT_XY = []   # プレイヤーの弾の座標

PLAYER_SHOT_DISP = [] # プレイヤーの弾を表示させるかどうか

ENEMY_SHOT_XY = []    # 敵の弾の座標

ENEMY_SHOT_XY_LENGTH = [] # 敵の弾の幅

ENEMY_SHOT_FLAG = 0   # 敵の弾のフェードアウト用フラグ

ENEMY_NUMBER = 0      # 敵の数

ENEMY_DEAD_XY_FLAG = [] # 敵が死んだときの座標と爆発の段階

SKILL_UP_ITME_FLAG = 0 # スキルアップアイテムのフラグ







class Player:

    def __init__(self, player_x, player_y, player_speed, shot_speed, shot_rapid_time):
```

```
        global PLAYER_SHOT_XY
        global PLAYER_SHOT_DISP
        global ENEMY_SHOT_XY

        # プレイヤーの x 幅,y 幅,弾の x 幅,y 幅
        self.player_shot_length = [16, 16, 8, 8]
        # プレイヤーの初期座標
        self.player_x = player_x
        self.player_y = player_y
        # プレイヤーのスピード
        self.player_speed = player_speed
        # 弾の座標
        self.shot_xy = []
        # 弾の速さ
        self.shot_speed = shot_speed
        # 弾の連射速度
        self.shot_rapid_time = shot_rapid_time
        # チャタリング防止用の変数
        self.shot_disp = []
        self.shot_i = self.shot_dispi = -1
        self.shot_flag = 1
        self.out_shot = []
        # 被弾フラグ
        self.dead_flag = 0
        # 爆発の段階
        self.explosion = 5
        # スキルアップアイテム情報
        self.skill_up_item = []
        self.skill_up_texttime = 40



        if self.__class__.__name__ == "Player":
            PLAYER_SHOT_XY = self.shot_xy
            PLAYER_SHOT_DISP = self.shot_disp

    # 更新の関数まとめ
```

```
    def update(self):
        self.update_player()
        self.update_shot()
        self.update_shot_receive()

    # プレイヤーの座標の更新
    def update_player(self):
        global PLAYER_X
        global PLAYER_Y
        if pyxel.btn(pyxel.KEY_LEFT) or pyxel.btn(pyxel.GAMEPAD_1_LEFT):
            self.player_x = max(self.player_x - self.player_speed, 0)

        if pyxel.btn(pyxel.KEY_RIGHT) or pyxel.btn(pyxel.GAMEPAD_1_RIGHT):
            self.player_x = min(self.player_x + self.player_speed, pyxel.width -
self.player_shot_length[0])

        if pyxel.btn(pyxel.KEY_UP) or pyxel.btn(pyxel.GAMEPAD_1_UP):
            self.player_y = max(self.player_y - self.player_speed, 0)

        if pyxel.btn(pyxel.KEY_DOWN) or pyxel.btn(pyxel.GAMEPAD_1_DOWN):
            self.player_y = min(self.player_y + self.player_speed, pyxel.height -
self.player_shot_length[1])
        PLAYER_X = self.player_x
        PLAYER_Y = self.player_y

    # プレイヤーが発した弾の座標の更新
    def update_shot(self):
        if pyxel.btn(pyxel.KEY_E):
            self.shot_flag = 1
            self.shot_xy.append([self.player_x+(self.player_shot_length[0]/2-
self.player_shot_length[2]/2),¥
                self.player_y-self.player_shot_length[3]+4])

            if len(self.shot_disp) >= self.shot_rapid_time and self.shot_i == self.shot_dispi¥
                and 1 not in self.shot_disp[(self.shot_rapid_time)*-1: ]:
                self.shot_disp.append(1)
```

```
            elif self.shot_i != self.shot_dispi:
                self.shot_disp.append(1)
                self.shot_dispi = -1*self.shot_dispi
            else:
                self.shot_disp.append(0)


        elif not pyxel.btn(pyxel.KEY_E) and self.shot_flag == 1:
            self.shot_flag = 0
            self.shot_i =-1*self.shot_i


        del_index = 0
        for i in range (len(self.shot_xy)):
            #print(len(self.shot_xy))
            index = i-del_index
            self.shot_xy[index][1] -= self.shot_speed
            if self.shot_xy[index][1] <= -self.player_shot_length[3]:
                self.shot_xy.pop(index)
                self.shot_disp.pop(index)
                del_index+=1


    # 被弾したことを検知する関数(当たり判定)
    def update_shot_receive(self):
        global SKILL_UP_ITME_FLAG
        if len(ENEMY_SHOT_XY)!= 0:
            i = 0
            for ag_x, ag_y in ENEMY_SHOT_XY:
                if self.player_x-ENEMY_SHOT_XY_LENGTH[i][0]+4 <= ag_x <=
self.player_x+self.player_shot_length[0]-4¥
                and self.player_y-ENEMY_SHOT_XY_LENGTH[i][1]+4 <= ag_y <=
self.player_y+self.player_shot_length[1]-4 :
                        # print("out@@@!")
                        self.dead_flag = 1
                i+=1
        i = 0
        while i < len(self.skill_up_item):
```

```
            if self.player_x-8+4 <= self.skill_up_item[i][0] <=
self.player_x+self.player_shot_length[0]-4¥
                and self.player_y-8+4 <= self.skill_up_item[i][1] <=
self.player_y+self.player_shot_length[1]-4 :
                if self.skill_up_item[i][2] == 1:
                    self.player_speed += 0.4
                if self.skill_up_item[i][2] == 2:
                    self.shot_speed += 0.3
                if self.skill_up_item[i][2] == 3:
                    self.shot_rapid_time -= 2
                SKILL_UP_ITME_FLAG = self.skill_up_item[i][2]
                self.skill_up_item.pop(0)

            i += 1









class Enemy_1(Player):
    def __init__(self,lenghs, player_x, player_y, player_speed, shot_speed, shot_ins_time,
enemy_liner_type):
        super().__init__(player_x, player_y, player_speed, shot_speed, 0)
        self.player_shot_length = lenghs # [敵の縦, 敵の横, 弾の縦, 弾の横]
        self.shot_global = []
        self.shot_flag = 0
        self.shot_ins_time = shot_ins_time
        self.enemy_liner_type = enemy_liner_type
        self.score_plus = 0 # 弾が当たった時, スコア加算されたかのフラグ
        self.pre_shot_time = time.time()
        self.next_shot_sec = random.uniform(self.shot_ins_time, 2.0)



    def update_player(self):
        self.player_y += self.player_speed*0.5 # 敵のy方向の速さ
```

```
    def update_shot(self):
        global ENEMY_SHOT_XY
        global ENEMY_SHOT_XY_LENGTH
        # 弾の生成
        if self.dead_flag != 1:
            if self.pre_shot_time + self.next_shot_sec - time.time() <= 0.1¥
                and self.player_y <= DISP_Y-self.player_shot_length[1]:
                self.shot_xy.append([self.player_x+(self.player_shot_length[0]/2-
self.player_shot_length[2]/2),¥
                    self.player_y+self.player_shot_length[1]-4])
                ENEMY_SHOT_XY.append(self.shot_xy[-1])
                self.shot_global.append(len(ENEMY_SHOT_XY)-1)
                ENEMY_SHOT_XY_LENGTH.append(self.player_shot_length[2:4])
                # print(ENEMY_SHOT_XY_LENGTH[0][0])
                self.pre_shot_time = time.time()
                self.next_shot_sec = random.uniform(self.shot_ins_time, 2.0)

    # ---敵-弾だけ進む-- #
    def progress_shot(self):

        global ENEMY_SHOT_FLAG
        del_index = 0

        for i in range (len(self.shot_xy)):
            index = i-del_index
            self.shot_xy[index][1] -= self.shot_speed

            ENEMY_SHOT_XY[self.shot_global[index]] = self.shot_xy[index]
            if self.shot_xy[index][1] > DISP_Y and self.shot_global[index] == 0:
                self.shot_xy.pop(0)
                self.shot_global.pop(0)
                ENEMY_SHOT_FLAG+=1
                ENEMY_SHOT_XY.pop(0)
                del_index+=1
```

```
                self.shot_global = list(map(lambda x:x-1, self.shot_global))




    def update_shot_receive(self):
        global ENEMY_DEAD_XY_FLAG
        if len(PLAYER_SHOT_XY)!= 0:
            i = 0
            while i < len(PLAYER_SHOT_XY):
                if self.player_x-8 <= PLAYER_SHOT_XY[i][0] <=
self.player_x+self.player_shot_length[0] ¥
                and self.player_y +self.player_shot_length[1]-2 >= PLAYER_SHOT_XY[i][1] and
self.player_y <= PLAYER_SHOT_XY[i][1]+8¥
                and PLAYER_SHOT_DISP[i] == 1 and self.dead_flag != 1:
                    # print("out!!!!!!!!!")
                    self.dead_flag = 1
                    if self.enemy_liner_type != 6:
                        ENEMY_DEAD_XY_FLAG.append([self.player_x, self.player_y, 5, 3]) # 爆発の処
理
                    else :
                        ENEMY_DEAD_XY_FLAG.append([self.player_x, self.player_y, 6, 3]) # 爆発の処
理


                    PLAYER_SHOT_XY.pop(i)   # プレイヤーの弾を削除
                    PLAYER_SHOT_DISP.pop(i)
                i += 1


class Enemy_2(Enemy_1):
    def update_player(self):
        if self.player_y < PLAYER_Y:
            if self.player_x < PLAYER_X:
                self.player_x = self.player_x + self.player_speed


            if self.player_x > PLAYER_X:
                self.player_x = self.player_x - self.player_speed
```

```
            if self.player_x == PLAYER_X:
                pass

        self.player_y += self.player_speed*0.5 # 敵のy方向の速さ

class Enemy_3(Enemy_1):
    def __init__(self,lenghs, player_x, player_y, player_speed, shot_speed, shot_ins_time,
enemy_liner_type):
        super().__init__(lenghs, player_x, player_y, player_speed, shot_speed, shot_ins_time,
enemy_liner_type)
        self.pre_move_time = time.time()
        self.next_move_sec = random.uniform(2.0, 4.0)
        self.move_lr = random.choice([1, -1])
        self.aim_player_xy = []

    def update_player(self):
        if self.pre_move_time + self.next_move_sec - time.time() <= 0:
            self.move_lr *= -1
            self.pre_move_time = time.time()
            self.next_move_sec = random.uniform(2.0, 4.0)
        if self.player_x <= 0:
            self.move_lr = 1
        if self.player_x + self.player_shot_length[0] >= DISP_X:
            self.move_lr = -1



        if self.move_lr == 1:
            self.player_x = self.player_x + self.player_speed
        elif self.move_lr == -1:
            self.player_x = self.player_x - self.player_speed

        self.player_y += self.player_speed*0.5 # 敵のy方向の速



class Enemy_4(Enemy_3):
    def __init__(self,lenghs, player_x, player_y, player_speed, shot_speed, shot_ins_time,
enemy_liner_type):
```

```
        super().__init__(lenghs, player_x, player_y, player_speed, shot_speed, shot_ins_time,
enemy_liner_type)
        self.player_x_now = []


    def update_shot(self):
        global ENEMY_SHOT_XY
        global ENEMY_SHOT_XY_LENGTH
        # 弾の生成
        if self.dead_flag != 1:
            if self.pre_shot_time + self.next_shot_sec - time.time() <= 0.1¥
                and self.player_y <= DISP_Y-self.player_shot_length[1]:
                self.shot_xy.append([self.player_x+(self.player_shot_length[0]/2-
self.player_shot_length[2]/2),¥
                    self.player_y+self.player_shot_length[1]-4])
                ENEMY_SHOT_XY.append(self.shot_xy[-1])
                self.shot_global.append(len(ENEMY_SHOT_XY)-1)
                ENEMY_SHOT_XY_LENGTH.append(self.player_shot_length[2:4])
                self.pre_shot_time = time.time()
                self.next_shot_sec = random.uniform(self.shot_ins_time, 2.0)
                self.player_x_now.append(PLAYER_X)


    # ---敵-弾だけ進む-- #
    def progress_shot(self):


        global ENEMY_SHOT_FLAG
        del_index = 0


        for i in range (len(self.shot_xy)):
            index = i-del_index
            if self.player_x_now[index]+4 > self.shot_xy[index][0]:
                if self.player_x_now[index]+4 - self.shot_xy[index][0] > abs(self.shot_speed):
                    self.shot_xy[index][0] -= self.shot_speed*0.7
                else :
                    self.shot_xy[index][0] = self.player_x_now[index]+4
            elif self.player_x_now[index]+4 < self.shot_xy[index][0]:
                if  self.shot_xy[index][0] - self.player_x_now[index]+4 > abs(self.shot_speed):
```

```
                    self.shot_xy[index][0] += self.shot_speed*0.7
                else :
                    self.shot_xy[index][0] =self.player_x_now+4
            elif self.player_x_now[index]+4 == self.shot_xy[index][0]:
                pass

            self.shot_xy[index][1] -= self.shot_speed



            ENEMY_SHOT_XY[self.shot_global[index]] = self.shot_xy[index]
            if self.shot_xy[index][1] > DISP_Y and self.shot_global[index] == 0:
                self.shot_xy.pop(0)
                self.shot_global.pop(0)
                self.player_x_now.pop(0)
                ENEMY_SHOT_FLAG+=1
                ENEMY_SHOT_XY.pop(0)
                del_index+=1
                self.shot_global = list(map(lambda x:x-1, self.shot_global))




class App:
    def __init__(self):
        global PLAYER_X
        global PLAYER_Y
        pyxel.init(DISP_X, DISP_Y,scale=3, caption="GOOD FLY 2")
        pyxel.load("my_resource.pyxres")

        # ******自由に設定できる******* #
        # 初期位置 x, y, 動きの速さ(値が大きいほど早い), 弾の速さ(値が大きいほど早い), 連射速度(値が大きいほど
遅い)
        self.player = Player(120, 220, 3, 2, 20)
        self.enemy_ins_time_min = 0.3           # 敵を次に生成する時間の最小値
        self.enemy_ins_time_max = 2.0           # 敵を次に生成する時間の最大値
        #    ******ここまで******    #
```

```
        self.player_speed = self.player.player_speed
        self.get_score = 0
        self.score_value = [10, 15,20, 25, 30, 25]
        self.shot_type = []
        self.enemy = []
        self.next_enemy_ins = 200
        self.next_skill_up_ins = 150
        self.pre_time = 0
        self.next_sec = 0
        self.pre_time_cloud = time.time()
        self.next_sec_cloud = random.uniform(1, 3)
        PLAYER_X = self.player.player_x
        PLAYER_Y = self.player.player_y
        self.gamemode = 0
        self.back_ground_mode = 0
        self.cloud = []
        self.star = []
        for i in range(0, 11):
            self.cloud.append([random.randint(-48, DISP_X), random.randint(-48, DISP_Y),
random.randint(1, 4)])

        star_type = list(range(1, 4))
        w = [0.9, 0.05, 0.05]
        for i in range(0, 20):
            next_star_type = random.choices(star_type, k = 1, weights=w)
            self.star.append([random.randint(-8, DISP_X), random.randint(-8, DISP_Y),
next_star_type[0]])

        pyxel.run(self.draw, self.updata)

    # ---敵のインスタンス--- #
    def enemy_instance(self):
        global ENEMY_NUMBER
        enemy_type = list(range(1, 7))
        w = [0.15, 0.15, 0.2, 0.2, 0.1, 0.2] # 生成確率
```

```
        next_enemy_type = random.choices(enemy_type, k = 1, weights=w)

        # Enemy_n([敵の縦, 敵の横, 弾の縦, 弾の横], 生成場所_x, 生成場所_y, 進む速さ, 弾の速さ, 弾の生成時間の下限(sec), (敵の種類))
        if next_enemy_type[0] == 1:
            self.enemy.append(Enemy_1([16, 16, 8, 8],random.randint(-8, DISP_X-8), -8, 1, -2, 1, 1) )
            self.shot_type.append(1)
        elif next_enemy_type[0] == 2:
            self.enemy.append(Enemy_2([16, 16, 8, 8],random.randint(-8, DISP_X-8), -8, 1, -2, 1, 2) )
            self.shot_type.append(1)
        elif next_enemy_type[0] == 3:
            self.enemy.append(Enemy_3([16, 16, 8, 8],random.randint(-8, DISP_X-8), -8, 1, -2, 1, 3) )
            self.shot_type.append(1)
        elif next_enemy_type[0] == 4:
            self.enemy.append(Enemy_4([16, 16, 8, 8],random.randint(-8, DISP_X-8), -8, 1, -2, 1, 4) )
            self.shot_type.append(2)
        elif next_enemy_type[0] == 5:
            self.enemy.append(Enemy_1([16, 16, 5, 10],random.randint(-8, DISP_X-8), -8, 5, -6, 0.2, 5) )
            self.shot_type.append(3)
        elif next_enemy_type[0] == 6:
            self.enemy.append(Enemy_1([32,24 , 8, 12],random.randint(-8, DISP_X-8), -8, 0.5, -3, 1, 6) )
            self.shot_type.append(4)



        ENEMY_NUMBER += 1

    # ---敵の弾のインデックス更新--- #
    def enemy_shot_index(self, a):
        for i in range(ENEMY_NUMBER):
            if i != a:
```

```
                self.enemy[i].shot_global = list(map(lambda x:x-ENEMY_SHOT_FLAG,
self.enemy[i].shot_global))


    # ---初期化--- #
    def initiarize(self):
        global PLAYER_SHOT_XY, PLAYER_SHOT_DISP, ENEMY_SHOT_XY, ENEMY_SHOT_FLAG, ENEMY_NUMBER,
ENEMY_DEAD_XY_FLAG, PLAYER_X, PLAYER_Y, SKILL_UP_ITME_FLAG

        PLAYER_SHOT_XY = []   # プレイヤーの弾の座標
        PLAYER_SHOT_DISP = [] # プレイヤーの弾を表示させるかどうか
        ENEMY_SHOT_XY = []    # 敵の弾の座標
        ENEMY_SHOT_FLAG = 0   # 敵の弾のフェードアウト用フラグ
        ENEMY_NUMBER = 0      # 敵の数
        ENEMY_DEAD_XY_FLAG = [] # 敵が死んだときの座標と爆発の段階
        SKILL_UP_ITME_FLAG = 0

        self.player = Player(120, 220, 3, 2, 20)
        self.enemy = []
        self.shot_type = []
        self.next_enemy_ins = 200
        self.next_skill_up_ins = 150
        self.pre_time = time.time()
        self.next_sec = random.uniform(self.enemy_ins_time_min, self.enemy_ins_time_max)
        PLAYER_X = self.player.player_x
        PLAYER_Y = self.player.player_y
        self.pre_time_cloud = time.time()
        if self.back_ground_mode == 0 or self.back_ground_mode == 1:
            self.next_sec_cloud = random.uniform(0.2, 2.0)
        elif self.back_ground_mode == 2 :
            self.next_sec_cloud = random.uniform(0, 0.2)


        self.get_score = 0


    # ---プレイヤー，敵の更新--- #
    def updata(self):
```

```
        global ENEMY_NUMBER
        global ENEMY_SHOT_FLAG
        global ENEMY_SHOT_XY_LENGTH



        # ゲーム終了
        if pyxel.btnp(pyxel.KEY_Q):
            pyxel.quit()



        # タイトル画面
        if self.gamemode == 0:
            self.pre_time = time.time()
            self.next_sec = random.uniform(self.enemy_ins_time_min, self.enemy_ins_time_max)
            if pyxel.btnp(pyxel.KEY_SPACE):
                self.gamemode = 1
            if pyxel.btnp(pyxel.KEY_TAB):
                self.back_ground_mode = (self.back_ground_mode + 1) % 3




        # ゲーム中
        elif self.gamemode == 1:

            # スペースを押している間はゆっくり動く
            if pyxel.btn(pyxel.KEY_SPACE):
                self.player.player_speed = 1
            else :
                self.player.player_speed = self.player_speed


            # -敵-生成
            if self.pre_time + self.next_sec - time.time() <= 0:
                self.pre_time = time.time()
                self.next_sec = random.uniform(self.enemy_ins_time_min,
self.enemy_ins_time_max)
                self.enemy_instance()
```

```
        # -プレイヤー-更新
        self.player.update()
        i = 0

        # -敵-被弾してなかったら更新
        while i < ENEMY_NUMBER:
            if self.enemy[i].dead_flag != 1:
                self.enemy[i].update()
            elif self.enemy[i].score_plus == 0 :
                type = str(self.enemy[i].__class__.__name__)
                t = int(type[-1])-1
                if t == 5:
                    t = self.enemy[i].enemy_liner_type
                #print(self.get_score[t])
                self.get_score += self.score_value[t]
                self.enemy[i].score_plus = 1
            self.enemy[i].progress_shot() # 弾だけ進む
            # -敵-弾がフェードアウトしたらすべての弾のインデックスを更新
            if ENEMY_SHOT_FLAG > 0:
                self.enemy_shot_index(a = i)
                ENEMY_SHOT_XY_LENGTH.pop(0)
                ENEMY_SHOT_FLAG = 0
            i+=1

        # 敵が０じゃなかったら
        if len(self.enemy)!=0 :
            # -敵-弾が放たれず，かつフェードアウトしたら，削除（打たれても実態はあり）
            if len(self.enemy[0].shot_global) == 0 and self.enemy[0].player_y > DISP_Y+30:
                self.enemy.pop(0)
                self.shot_type.pop(0)
                ENEMY_NUMBER -= 1

        # スコアによる敵の排出率の増加
        if self.get_score >= self.next_enemy_ins:
            if self.enemy_ins_time_min -0.1 > 0:
```

```
                self.enemy_ins_time_min -= 0.1
            if self.enemy_ins_time_min -0.1 > 1:
                self.enemy_ins_time_max -= 0.1

            self.next_enemy_ins += 200

        # スキルアップアイテムの生成及び削除
        if self.get_score >= self.next_skill_up_ins:
            r = random.randint(1, 3)
            if r == 1 and self.player.player_speed <= 6 ¥
            or r == 2 and self.player.shot_speed <= 5 ¥
            or r == 3 and self.player.shot_rapid_time >= 5:
                self.player.skill_up_item.append([random.randint(0, DISP_X-8), -8, r])
            self.next_skill_up_ins += 150

        i = 0
        while i < len(self.player.skill_up_item):
            self.player.skill_up_item[i][1] += 2
            if self.player.skill_up_item[i][1] >= DISP_Y+8:
                self.player.skill_up_item.pop(0)
                i -= 1
            i += 1

    # 雲の生成
    if self.gamemode == 0 or self.gamemode == 1:
        if self.back_ground_mode == 0 or self.back_ground_mode == 1:
            if self.pre_time_cloud + self.next_sec_cloud - time.time() <= 0:
                self.pre_time_cloud = time.time()
                self.next_sec_cloud = random.uniform(0.2, 2)
                self.cloud.append([random.randint(-48, DISP_X), -48, random.randint(1, 4)])
            j = 0
            while j < len(self.cloud):
                if self.cloud[j][1] > DISP_Y+48:
                    self.cloud.pop(j)
                    j -= 1
                self.cloud[j][1] += 1
```

```
            j += 1

    # 星の生成
        if self.back_ground_mode == 2:
            star_type = list(range(1, 4))
            w = [0.9, 0.05, 0.05]

            if self.pre_time_cloud + self.next_sec_cloud - time.time() <= 0:
                self.pre_time_cloud = time.time()
                self.next_sec_cloud = random.uniform(0, 0.2)
                next_star_type = random.choices(star_type, k = 1, weights=w)
                self.star.append([random.randint(-8, DISP_X), -8, next_star_type[0]])

            j = 0
            while j < len(self.star):
                if self.star[j][1] > DISP_Y+8:
                    self.star.pop(j)
                    j -= 1
                self.star[j][1] += 5
                j += 1

    # ゲームオーバー
    elif self.gamemode == 2:
        self.pre_time = time.time()
        self.next_sec = random.uniform(0.1, 2.0)
        if pyxel.btnp(pyxel.KEY_SPACE):
            self.initiarize()
            self.gamemode = 1
        if pyxel.btnp(pyxel.KEY_ENTER):
            self.initiarize()
            self.gamemode = 0


    # -プレイヤー-被弾したらゲームオーバー #
    if self.player.dead_flag == 1:
        self.gamemode = 2
```

```
def draw(self):
    if self.back_ground_mode == 0:
        pyxel.cls(6)
    if self.back_ground_mode == 1:
        pyxel.cls(9)
    if self.back_ground_mode == 2:
        pyxel.cls(0)

    global ENEMY_DEAD_XY_FLAG
    global SKILL_UP_ITME_FLAG

    # 雲を描写
    if self.back_ground_mode == 0 or self.back_ground_mode == 1:
        for i in range(len(self.cloud)):
            if self.cloud[i][2] == 1:
                pyxel.blt(self.cloud[i][0], self.cloud[i][1], 2, 0, 0, 64, 40, 0)
            if self.cloud[i][2] == 2:
                pyxel.blt(self.cloud[i][0], self.cloud[i][1], 2, 0, 40, 40, 32, 0)
            if self.cloud[i][2] == 3:
                pyxel.blt(self.cloud[i][0], self.cloud[i][1], 2,32, 64, 32, 32, 0)
            if self.cloud[i][2] == 4:
                pyxel.blt(self.cloud[i][0], self.cloud[i][1], 2,0, 96, 48, 32, 0)
    # 星を描写
    elif self.back_ground_mode == 2:
        for i in range(len(self.star)):
            if self.star[i][2] == 1:
                pyxel.blt(self.star[i][0], self.star[i][1], 2, 0, 72, 8, 8, 0)
            if self.star[i][2] == 2:
                pyxel.blt(self.star[i][0], self.star[i][1], 2, 8, 72, 8, 8, 0)
            if self.star[i][2] == 3:
                pyxel.blt(self.star[i][0], self.star[i][1], 2, 0, 80, 8, 8, 0)
```

```
        if self.gamemode == 0:
            pyxel.blt(50, 50, 1, 0, 0, 32, 32, 0)
            pyxel.blt(83, 50, 1, 32, 0, 32, 32, 0)
            pyxel.blt(116, 50, 1, 32, 0, 32, 32, 0)
            pyxel.blt(149, 50, 1, 0, 32, 32, 32, 0)
            pyxel.blt(70, 84, 1, 32, 32, 32, 32, 0)
            pyxel.blt(101, 84, 1, 32, 64, 32, 32, 0)
            pyxel.blt(125, 84, 1, 0, 96, 32, 32, 0)
            pyxel.blt(167, 82, 1, 32, 96, 32, 32, 0)
            color = 0
            if self.back_ground_mode == 2:
                color = 7
            pyxel.text(100, 200, "Press Space Key", color)




        # 弾を描写
        for i in range(len(self.player.shot_xy)):
            if self.player.shot_disp[i] == 1:
                pyxel.blt(self.player.shot_xy[i][0], self.player.shot_xy[i][1], 0, 16, 0,
self.player.player_shot_length[2], self.player.player_shot_length[3], 1)

        # 戦闘機を描写
        pyxel.blt(self.player.player_x, self.player.player_y, 0, 0, 0,
self.player.player_shot_length[0], self.player.player_shot_length[1], 1,)

        for i in range(len(self.enemy)):
            for j in range(len(self.enemy[i].shot_xy)):
                if self.shot_type[i] == 1:
                    pyxel.blt(self.enemy[i].shot_xy[j][0], self.enemy[i].shot_xy[j][1], 0, 24, 0,
self.enemy[i].player_shot_length[2], self.enemy[i].player_shot_length[3], 1)
                if self.shot_type[i] == 2:
                    pyxel.blt(self.enemy[i].shot_xy[j][0], self.enemy[i].shot_xy[j][1], 0, 16, 9,
self.enemy[i].player_shot_length[2], self.enemy[i].player_shot_length[3], 1)
```

```
                if self.shot_type[i] == 3:
                    pyxel.blt(self.enemy[i].shot_xy[j][0], self.enemy[i].shot_xy[j][1], 0, 23,
13, self.enemy[i].player_shot_length[2], self.enemy[i].player_shot_length[3], 1)
                if self.shot_type[i] == 4:
                    pyxel.blt(self.enemy[i].shot_xy[j][0], self.enemy[i].shot_xy[j][1], 0, 20,
27, self.enemy[i].player_shot_length[2], self.enemy[i].player_shot_length[3], 1)



        # 敵を描写
        for i in range(len(self.enemy)):
            if self.enemy[i].dead_flag != 1:
                if  self.enemy[i].enemy_liner_type == 1:
                    pyxel.blt(self.enemy[i].player_x, self.enemy[i].player_y, 0, 0, 32,
self.enemy[i].player_shot_length[0], self.enemy[i].player_shot_length[1], 1,)
                elif self.enemy[i].enemy_liner_type == 2:
                    pyxel.blt(self.enemy[i].player_x, self.enemy[i].player_y, 0, 0, 16,
self.enemy[i].player_shot_length[0], self.enemy[i].player_shot_length[1], 1,)
                elif self.enemy[i].enemy_liner_type == 3:
                    pyxel.blt(self.enemy[i].player_x, self.enemy[i].player_y, 0, 0, 48,
self.enemy[i].player_shot_length[0], self.enemy[i].player_shot_length[1], 1,)
                elif self.enemy[i].enemy_liner_type == 4:
                    pyxel.blt(self.enemy[i].player_x, self.enemy[i].player_y, 0, 0, 64,
self.enemy[i].player_shot_length[0], self.enemy[i].player_shot_length[1], 4,)
                elif self.enemy[i].enemy_liner_type == 5:
                    pyxel.blt(self.enemy[i].player_x, self.enemy[i].player_y, 0, 0, 80,
self.enemy[i].player_shot_length[0], self.enemy[i].player_shot_length[1], 4,)
                elif self.enemy[i].enemy_liner_type == 6:
                    pyxel.blt(self.enemy[i].player_x, self.enemy[i].player_y, 0, 0, 96,
self.enemy[i].player_shot_length[0], self.enemy[i].player_shot_length[1], 4,)



        # 敵の爆破の描写
        i = 0
        while i < len(ENEMY_DEAD_XY_FLAG) :


            if ENEMY_DEAD_XY_FLAG[i][2] > 0:
```

```
                if ENEMY_DEAD_XY_FLAG[i][2] == 6:
                    pyxel.blt(ENEMY_DEAD_XY_FLAG[i][0]+4, ENEMY_DEAD_XY_FLAG[i][1]+4, 0, 32, 96,
24, 24, 1)
                else :
                    pyxel.blt(ENEMY_DEAD_XY_FLAG[i][0]+8, ENEMY_DEAD_XY_FLAG[i][1]+8, 0, 32, (5-
ENEMY_DEAD_XY_FLAG[i][2])*16, 16, 16, 1)
                ENEMY_DEAD_XY_FLAG[i][3] -= 1
                if ENEMY_DEAD_XY_FLAG[i][3] == 0:
                    ENEMY_DEAD_XY_FLAG[i][2] -= 1
                    ENEMY_DEAD_XY_FLAG[i][3] = 3

            if ENEMY_DEAD_XY_FLAG[i][2] == 0:
                ENEMY_DEAD_XY_FLAG.pop(0)
                i-=1
            i+=1

        # スキルアップアイテムの描写
        for i in range(len(self.player.skill_up_item)):
            if self.player.skill_up_item[i][2] == 1:
                pyxel.blt(self.player.skill_up_item[i][0], self.player.skill_up_item[i][1], 0,
48, 0, 8, 8, )
            if self.player.skill_up_item[i][2] == 2:
                pyxel.blt(self.player.skill_up_item[i][0], self.player.skill_up_item[i][1], 0,
56, 0, 8, 8, )
            if self.player.skill_up_item[i][2] == 3:
                pyxel.blt(self.player.skill_up_item[i][0], self.player.skill_up_item[i][1], 0,
48, 8, 8, 8, )

        # スキルアップアイテムを得た描写
        if SKILL_UP_ITME_FLAG > 0:
            if SKILL_UP_ITME_FLAG == 1:
                pyxel.text(10, 10, "Player Speed UP!!!!", 2)
            if SKILL_UP_ITME_FLAG == 2:
                pyxel.text(10, 10, "Shot Speed UP!!!!", 3)
            if SKILL_UP_ITME_FLAG == 3:
                pyxel.text(10, 10, "Shot Rapid Speed UP!!!!", 1)
```

```
            if self.player.skill_up_texttime == 0:
                self.player.skill_up_texttime = 40
                SKILL_UP_ITME_FLAG = 0
            self.player.skill_up_texttime-=1



        # スコアの描写
        if self.gamemode != 0:
            color = 0
            if self.back_ground_mode == 2:
                color = 7
            pyxel.text(210, 10, "Score:{}".format(self.get_score), color)

        # プレイヤーの爆破
        if self.gamemode == 2 :
            pyxel.blt(PLAYER_X, PLAYER_Y, 0, 32, 0, 16, 16, 1)
            pyxel.blt(63, 50, 1, 96, 0, 32, 32, 0)
            pyxel.blt(96, 50, 1, 64, 32, 32, 32, 0)
            pyxel.blt(129, 50, 1, 96, 32, 32, 32, 0)
            pyxel.blt(161, 50, 1, 64, 64, 32, 32, 0)
            pyxel.blt(63, 84, 1, 64, 0, 32, 32, 0)
            pyxel.blt(96, 84, 1, 96, 64, 32, 32, 0)
            pyxel.blt(129, 84, 1, 64, 64, 32, 32, 0)
            pyxel.blt(161, 84, 1, 0, 64, 32, 32, 0)
            color = 0
            if self.back_ground_mode == 2:
                color = 7
            pyxel.text(105, 150, "Your Score : ", color)
            pyxel.text(155, 150, str(self.get_score), 8)
            pyxel.text(89, 180, "Home : Press Enter Key", color)
            pyxel.text(85, 190, "Retry : Press Space Key", color)
            pyxel.text(89, 200, "quit : Press ¥"q¥" Key", color)
App()
```

リスト 1. 本作品のソースコード