Rubyを用いたシューティングゲームの作成

松谷　一希　渡邊　蒼来

１．研究概要

今回、私たちは、HTML と Ruby の知識を深めるとともに、ゲーム開発の経験を積むことを目標とし､ブラウザ上で動く 2D 横スクロールシューティングゲームを作成した。昨年、プログラミング技術の授業でゲーム作成を行ったが失敗に終わったため、前回の反省を生かし、比較的、制作難易度の低いシューティングゲームを制作することにした。

シューティングゲームを作る上で、私たちは､”宇宙”をテーマにゲーム作りを行った。

先生からスターソルジャーというゲームを紹介してもらい、このようなゲームを制作してみたいと考えたからである。

２．研究の具体的内容

(１)Ruby と Dxopal について

Ruby は日本発のオブジェクト指向スクリプト言語で、シンプルな文法と高い生産性を特徴とする。就職後、Ruby を使用するため、学習を兼ねて Ruby にした。

Dxopal はその Ruby をブラウザ上でゲーム開発に活用できるようにしたライブラリで、Ruby のコードを JavaScript に変換して動作させる仕組みを持っている。

写真１　Ruby のロゴ

(２)シューティングゲームについて

シューティングゲームとは、弾丸等の飛び道具を打ち出し、敵を撃つコンピューターゲームの一種である。また、私たちが作った 2D シューティングゲームはその名の通りに 2D（二次元的視点）のシューティングゲームである。私たちが作ったシューティングゲームは、スコアを稼ぎながらボスの撃破を目指すシューティングゲームである。

(３)ゲーム作成について

ゲーム作成は、表１のような計画でゲームを作成した。

表１　年間計画

３月-４月	Ruby の学習・ゲームの設計
５月	プログラム作成
６月	プログラム作成・企画発表
７月-10 月	プログラム作成
11 月	デバッグ・文化祭発表
12 月	デバッグ・報告書作成
１月	課題研究発表会

(ア)Ruby の学習

私たちは Ruby に触れたことがなかったので、まず Ruby の知識を深めるために、インターネットや図書館の本を参考に学習を行った。

(イ)ゲームの設計

このシューティングゲームは弾を何発でも発射することができる。また、様々な動きをする敵機が出現する。敵機は４種類いる。自機は上下左右に動けるようにした。ゲームのクリア条件はボスを倒すことにした。WASD キーか矢印キーを押したら移動できるようにした。それは、ユーザアクセシビリティの向上のためである。

(ウ)デザイン

テーマに合わせて素材を無料のサイトから引用した。

(エ)プログラム作成

友人や先生からアドバイスをいただき、以下の順番でシューティングゲームを制作した。

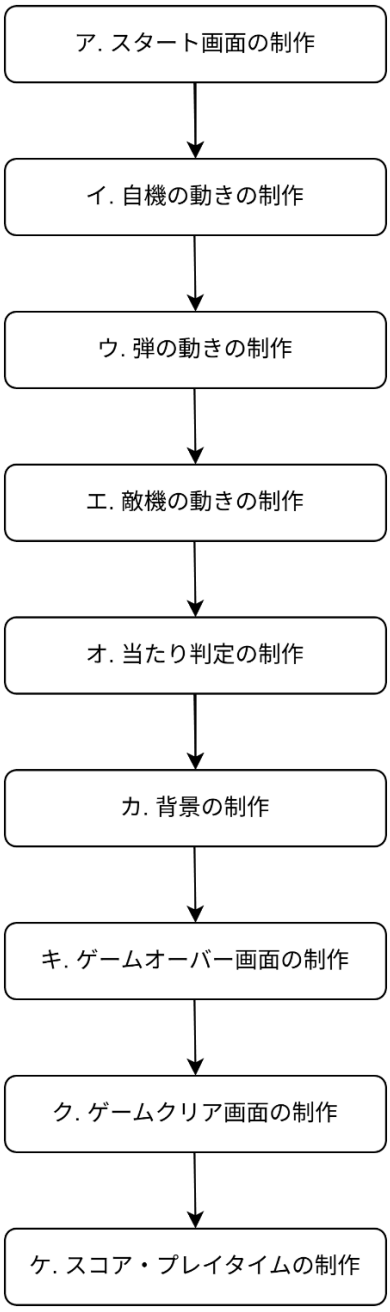

図１　プログラム作成の流れ

(エ)デバッグ

デバッグでは、完成したシューティングゲームの不具合を修正した。

(オ)その他

企画発表では、ゲーム作成の企画を発表し、文化祭では、成果を展示してプレイしてもらった。

(４)各処理の制作

(ア)スタート画面の制作

図２　スタート画面の画像

スタート画面は、実習の内容を活かしてHTML で制作を行った。なぜ HTML にしたのかというと、dxopal で作成したページを表示できなかったので、静的ページからのリンクをたどるようにしたためである。（図２）

(イ)自機の動きの制作

図３　自機の画像

```
def move
  @player_x -= 10 if Input.key_down?(K_LEFT) || Input.key_down?(K_A)
  @player_x += 10 if Input.key_down?(K_RIGHT) || Input.key_down?(K_D)
  @player_y -= 10 if Input.key_down?(K_UP) || Input.key_down?(K_W)
  @player_y += 10 if Input.key_down?(K_DOWN) || Input.key_down?(K_S)
  @player_x = [[@player_x, 50].max, Window.width - @size - 260].min
  @player_y = [[@player_y, -45].max, Window.height - @size - 160].min

  if Input.key_push?(K_SPACE)
    @bullets << Bullet_player.new(@player_x + @size / 2 + 143, @player_y + 84, @bullet_image)
  end

  @bullets.each(&:move)
  @bullets.reject!(&:out_of_bounds?)
end
```

図４　自機のプログラミングの一部

ゲームを開始すると自機は左側中央の定められた位置に出現しゲームがスタートされる。

それは、自機が一定時間の間、攻撃されない状況を作るためである。

自機のスピードについては、初期の段階で班員以外の友人にプレイしてもらう機会があり、自機のスピードが速すぎる、という意見を頂いたため思い当たる最適な速さを模索しつつ、何度も調整を行った。具体的には、スピードを変える変数に 20 の値を入れると一瞬で画面端に行ってしまったため少しずつ値を減らし最適な速さに調整した。(図４)

(ウ)弾の動きの制作

具体的には、敵機を倒してもその敵機が発射した弾は消えないようにしたり、弾を大きくして見えやすいようにした。

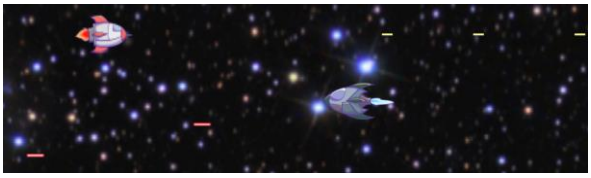

図５　弾の画像

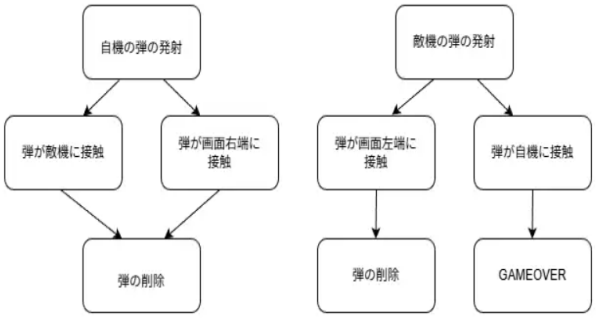

図６　弾の処理の流れ図

(エ)敵機の動きの制作

敵機は以下の表２の通りに実装した。

図７　enemy１

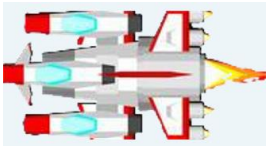

図８　enemy２

図９　enemy３

図 10　boss

表２　敵機の説明

名前	移動量		出現タイミング	HP	備考
	x	y	Score		
enemy1	-5	-2	0	1	
enemy2	sin(t)		100	3	t は加算され続ける
enemy3	-8	0	200	100	出現後３秒停止
boss	0	±5	500	200	

(オ)当たり判定の制作

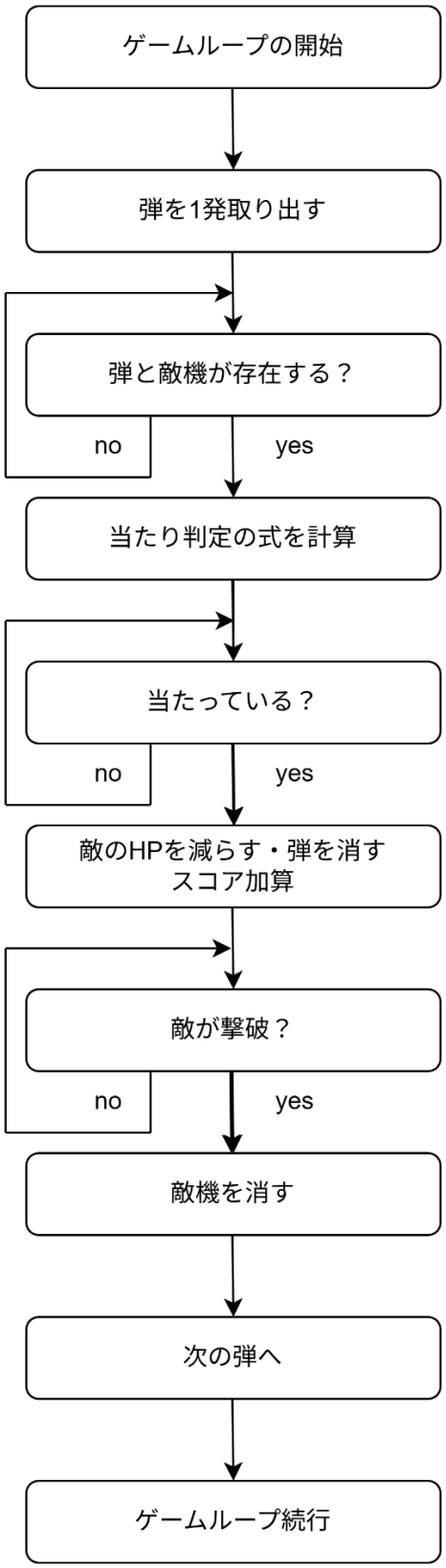

図 11　当たり判定の流れ

当たり判定を自機と敵機と弾につけている。画像と判定範囲が一致するように、値を何度も変え、少しめり込んだら当たるという判定になるように微調整を行った。自機が敵機や敵弾に当たるとゲームオーバー画面に遷移し、自機の弾が敵機に当たると敵機が消滅してスコアが加算されるようにした。

（カ）背景の制作

図 12　背景の画像

```
Window.draw(x, y, haikei_image)
Window.draw(x + haikei_image.width, y, haikei_image)
x = 0 if x <= -haikei_image.width
if !game_over && !game_clear
  enemy_bullets.each(&:move)
  enemy_bullets.reject!(&:out_of_bounds?)
  enemy_bullets.each { |b1| b1.draw}
  x -= 2
```

図 13　背景のプログラム画像

宇宙のテーマと合うように適切な画像を無料画像サイトから選別した。また先生から、背景がスクロールするとよりゲームらしくなるという、アドバイスを頂いたので、２枚の背景画像を繋げ、２枚目の背景画像に入ると、１枚目の背景画像に戻る条件を入れることで、この機能を実現した。さらに背景画像の移動を早くしすぎると、友人からプレイ中に酔ってしまうという意見をもらい、何度も修正・実行し、背景の移動をしつつ、酔わない程度の速さを見つけることができた(図 13)。

(キ)ゲームオーバー画面の制作

図 14　ゲームオーバー画面の制作

ゲームオーバー画面には GAMEOVER という文字と、スコア、プレイタイムを表示し、そして、エンターキーを押すとリスタートする機能を実装した(図 14)。

リスタートするときスコアとプレイタイムが引き継がれないように初期化した。リスタートしたときは HTML のスタート画面からゲーム開始した時とは違い、すぐにゲームが始まるようにしている。これはプレイアビリティ(快適に遊べるかの指標)の向上のためである。

(ク)ゲームクリア画面の制作

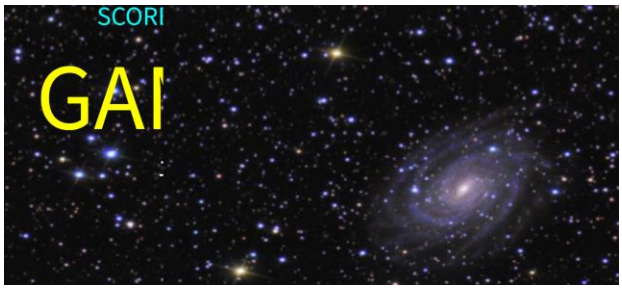

図 15　ゲームクリア画面の制作

ゲームオーバー画面と見た目はほとんど同じであるが、このゲームクリア画面にはエンディングロールがある(図 15)。

(ケ)スコア、プレイタイムの制作

図 16　スコア、プレイタイム画像

スコア、プレイタイムはゲームプレイ中は画面の左上に表示され、ゲームオーバー画面・ゲームクリア画面の時は画面の上の中央に表示される。ゲームプレイ中はあまり目立つようにはせずゲームが終了したときに、大きさや色を変えて見やすくしている(図 16)。

３．研究のまとめ

（渡邊）

今回の課題研究を通して、二人で協力して計画通りシューティングゲーム制作を完了させることの大変さを知ることができた。一人でプログラムをするのであれば動けばなんでもいいと思っていたが、二人で作るときは相手に自分がどんなコードを作っているのか理解してもらう必要があるので、コメント文を使ったり、わかりやすい変数名にする必要があった。制作作業当初は、相手に理解してもらえず、命名を工夫することに苦労した。

（松谷）

今回の課題研究では、初めて触れる言語で慣れないゲーム開発を共同で完成させるということもあり、困難な壁に何度も悩まされた。ゲームを作るという目的だけでは、どこまでのクオリティーの物を作るべきかわからなくなり、ゴールが見えなくなってしまいがちだが、共同開発している渡邊や、友人達、先生方からアドバイスをもらい、よりシューティングゲームらしくなるように、機能の追加・改善をしてきた。その結果、簡単にはクリアできない丁度いい難易度のシューティングゲ

ームを完成させることが出来た。

４．参考文献

「画像」

・いらすとや

https://www.irasutoya.com

・オープンゲームアート

https://opengameart.org/

「学習」

・dxopal の demo

https://yhara.github.io/dxopal/doc/api/index.html

・Rails ガイド

https://railsguides.jp/getting_started.html