提出日 2025 年 02 月 03 日

応用プログラミング最終レポート

学籍番号　202211905
所属　工学システム学類
学年　3 年
氏名　鈴木暁満

1 アプリケーションの概念設計

1. アプリ名称

SPASE SHOOTING

2. ユーザーは誰を想定したものか

スマートフォン、PC ゲームをプレイするユーザー

3. どのようなユーザ体験や機能を提供するか

- ゲームの目的

昨今、無料で遊べるいわゆる広告ゲームが多く存在し、話題になることや流行ることが多い。これら広告ゲームに共通する点として、手軽な操作かつゲーム内容が分かりやすい、ということが挙げられる。頭を使うことなく、ただ時間を浪費して“楽しい”という感情のみを得ようとする現代の若者をターゲットとしていると考えられる。本アプリは、このような広告ゲームの特徴を取り入れ、ユーザーが簡単に操作できる暇つぶしゲームを提供することを目的とする。

- 遊び方，ルール

PC 版では、

- 矢印キーで自機を操作
- スペースキーで弾を発射
- ゲームオーバーになるまで敵を倒す
- 敵を倒すとスコアが加算される
- 敵の弾に当たるとゲームオーバー
- ゲームオーバーになるとスコアが表示される
- ゲームオーバー後、リトライボタンを押すと再度ゲームが始まる

を繰り返すのみである。 スマホ版は未実装であるが、キーボード入力が全てボタン選択になる。

- どのように楽しんで貰いたいか

面白そう、をゲームへの入り口とし、簡単な操作で楽しめることを目指す。また、ゲームオーバー後にスコアが表示されることで、ユーザーが自己ベスト更新を目指し、繰り返しプレイすることを期待する。

2 アプリケーションの基本機能

- 使い方

ユーザーが矢印キーで自機を操作し、スペースキーで弾を発射すると、自機が弾を発射する。敵を倒すとスコアが加算される。敵の弾に当たるとゲームオーバーになる。

- UI（画面）の基本構成

タイトル場面には、ゲームタイトルと、図 1 のように各ボタンが配置される。ボタンはスタート、操作説明、終了の 3 つである。スタートすると画面中央のロケットが上昇する演出がある。

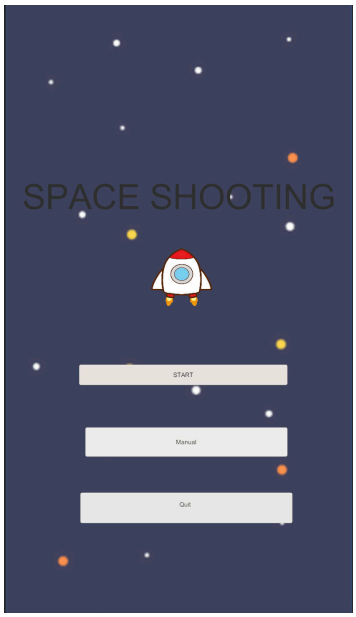

図 1: タイトル画面

次にゲーム画面の説明をする。ゲーム画面下部にプレイヤーが操作する自機が配置される。また、画面上部にはスコアが表示され、敵を倒すとスコアが加算される。敵は画面上部からランダムに出現する。これをプレイヤーが倒す、または避けることがゲームの基本的なルールである。
また、ランダムで 3 つのアイテムが出現する。それぞれ、スコアアップ、球数アップ、自機複製である。これを取得することがゲーム攻略の鍵である。図 2, 図 3
敵とアイテムはスコアに応じてランダムで出現するような設定になっており、スコアが高いほど難易度が上がる。

敵に衝突するとゲームオーバーになる。ゲームオーバーになると、スコアが表示される。スコアが表示された後、リトライボタンを押すと再度ゲームが始まる。 図 4

スコアが 1000 点になるとゲームクリアである。1000 点を超えると、自動的にゲームがストップし、リスタートとタイトルに戻るボタンが表示される。 図 5

図 2: ゲーム画面 1

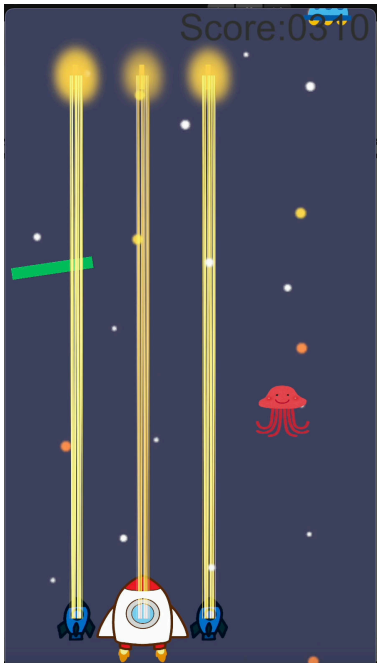

図 3: ゲーム画面 2

図 4: ゲームオーバー画面

図 5: ゲームクリア画面

3 システム仕様

編集を容易にし、また保守性を高めるためにクラスを分割してそれぞれに C# ファイルを作成した。以下に主要なクラスの概要とそれらのメソッドを示す。

3.1 RocketController.cs

ロケットの動きを制御するクラス

1. 主要な変数

- fallSpeed: ロケットの落下速度
- rotSpeed: ロケットの回転速度

2. 主要なメソッド

- Start(): 初期設定を行う
- Update(): ロケットの動きを更新する
- Explode(): 爆発エフェクトを生成し、爆発音を再生する

3.2 BulletController.cs

弾の動きを制御するクラス

1. 主要な変数

- speed: 弾の速度

2. 主要なメソッド:

- Start(): 初期設定を行う
- Update(): 弾の動きを更新する

3.3 RockController.cs

敵の動きを制御するクラス

1. 主要な変数

- speed: 弾の速度

2. 主要なメソッド

- Start(): 初期設定を行う
- Update(): 弾の動きを更新する

3.4 RockController.cs

敵の動きを制御するクラス

1. 主要な変数

- fallSpeed: 敵の落下速度
- rotSpeed: 敵の回転速度

2. 主要なメソッド

- Start(): 初期設定を行う
- Update(): 敵の動きを更新する

3.5 SceneController.cs

シーンの管理を行うクラス

1. 主要な変数

- RocketTransform: ロケットの Transform
- BottonSounds: ボタンの音

2. 主要なメソッド

- Start(): 初期設定を行う
- LoadGameScene(): ゲームシーンを読み込む
- LoadManualScene(): マニュアルシーンを読み込む
- LoadTitleScene(): タイトルシーンを読み込む
- QuitGame(): ゲームを終了する

3.6 UIController.cs

UI の管理を行うクラス

1. 主要な変数

- scoreText: スコアを表示するテキスト
- gameOverText: ゲームオーバーを表示するテキスト
- gameClearText: ゲームクリアを表示するテキスト
- score: 現在のスコア

2. 主要なメソッド

- Start(): 初期設定を行う
- AddScore(int points): スコアを追加する
- GameOver(): ゲームオーバーを表示する
- DisplayGameClear(): ゲームクリアを表示する
- UpdateScoreUI(): スコア UI を更新する

3.7 GameClearManager.cs

ゲームクリアの管理を行うクラス

1. 主要な変数

- gameClearScreen: ゲームクリア画面
- firstSelectedButton: 最初に選択されるボタン
- isGameClear: ゲームクリアの状態

2. 主要なメソッド

- Update(): ゲームクリアの条件をチェックする
- TriggerGameClear(): ゲームクリアをトリガーする

- IsGameClear(): ゲームクリアの状態を返す

3.8 GameOverManager.cs

ゲームオーバーの管理を行うクラス

1. 主要な変数

- gameOverScreen: ゲームオーバー画面
- firstSelectedButton: 最初に選択されるボタン
- isGameOver: ゲームオーバーの状態

2. 主要なメソッド

- Update(): ゲームオーバーの条件をチェックする
- TriggerGameOver(): ゲームオーバーをトリガーする
- IsGameOver(): ゲームオーバーの状態を返す