令和6年度 メディア情報学プログラミング演習
グループプログラミング レポート
料理提供ゲーム「MiniCook」

2025年2月17日

学科	情報理工学域
クラス	J1
グループ番号	26
2210259	米谷祐希
2210730	鈴木早紀
2210743	吉田陽音

1 概要説明

このゲームは、レストランで働くプレイヤーが、制限時間内に料理を作るゲームである。以下の料理提供までの手順を繰り返すことでポイントを獲得し、制限時間終了時にスコアとランクが表示される。

1. オーダーの確認

   まず、画面上部にランダムにオーダーが提示される。オーダーには、使う食材と調理方法が記載されている。各オーダーにはそれぞれ制限時間が設定されており、残り時間はオーダー上のゲージにリアルタイムに表示される。

2. 食材の調理

   次に、オーダーに記載されている食材を、各食材ボックスから取り出す。各食材を持ったまま、各調理器具の前でアクションボタンを押すことで、食材が加工される。

3. 料理の完成と提供

   料理は、加工された食材とお皿を組み合わせることで完成する。それらを組み合わせて料理ができあがれば、提供口に置くことで提供となり、オーダーと一致しているか判定される。一致していれば加点、間違っていれば減点となる。

また、ゲームは 3 画面に分かれており、スタート画面、ゲーム画面、リザルト画面がある。また、各画面や各動作には BGM や効果音がついている。操作はキーボードの A,S,D,W,J,K,Space キーを用いている。

作業は GitHub を用い保存・共有を行った。米谷が Model と全体の管理、鈴木が View、吉田が Controller を主に担当したが、最終的には各自の担当領域を超えて協力しながら取り組んだ。

2 設計方針

図 1 にクラス図を示す。MVC モデルで設計した。

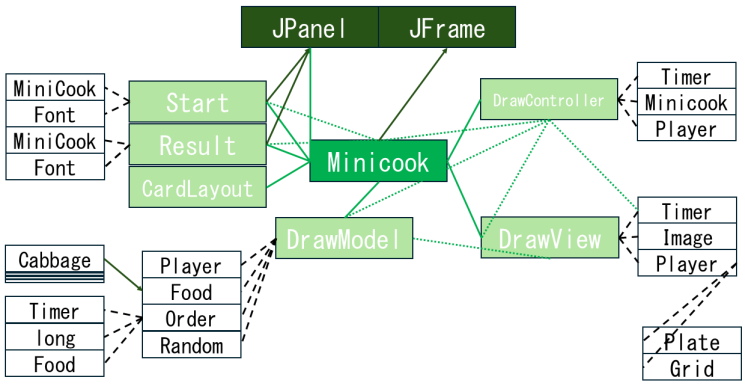


図 1: クラス図

3 プログラムの説明

以下にクラスとその説明を示す。

- MiniCook
- Model
  - Food
  - Order
- View
  - Timer
  - Image
  - Player
    - Plate
    - Grid
- Controller
- Start
- Result
- CardLayout
- AudioManager

4 実行例

スタート画面

実行すると始めにこの画面 (a) が現れる。スタートボタンを押すとゲーム画面：スタート時 (c) になる。

リザルト画面

ゲーム終了後はこのリザルト画面 (b) になる。スコアによってランクが星の数で表される。

ゲーム画面：スタート時

スタート時の画面 (c) では、食材などは何もなく、オーダーが 1 つ入るところから開始される。上部にはオーダー、中央にはゲーム部分、下部にはスコアと制限時間を表示している。

ゲーム画面：オーダー

画面上部のオーダー (d) では、完成品、必要な食材、加工方法、残り時間が示されている。

ゲーム画面：加工前

加工前の食材 (e) をボックスから取り出す。

ゲーム画面：加工後

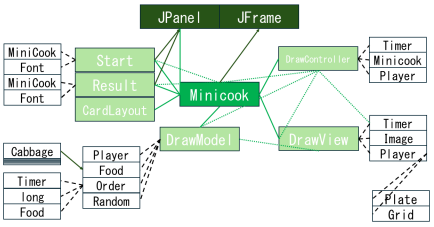
調理器具でアクションを行うと加工される。

ゲーム画面：組み合わせ

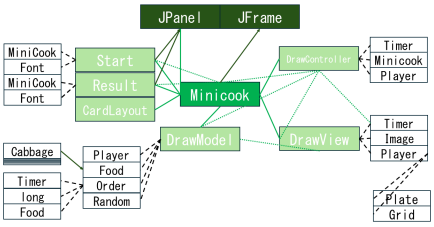
皿の上に各食材を載せると画像がそれに伴い完成品となる。

ゲーム画面：提供

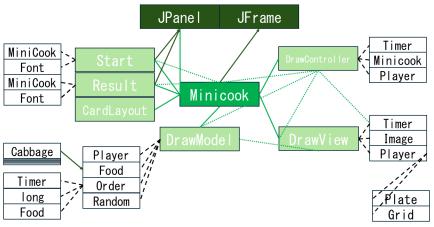
完成した料理を提供口に置くと、ホールスタッフが取りに来る。



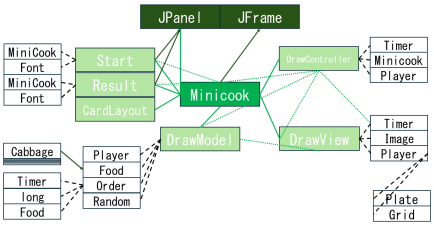
(a) スタート画面



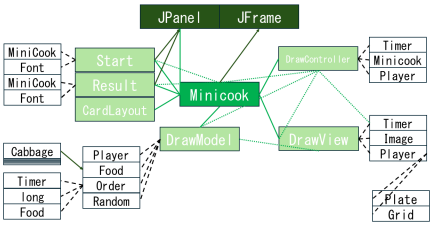
(b) リザルト画面



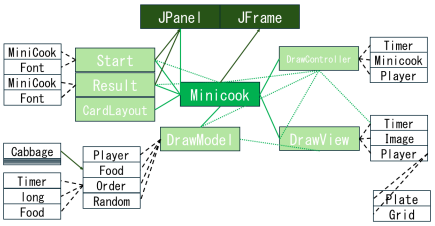
(c) ゲーム画面：スタート時



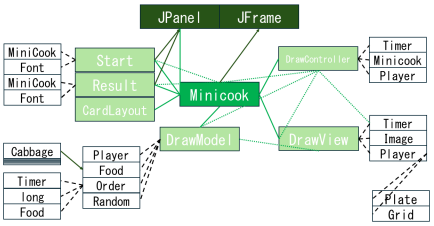
(d) ゲーム画面：オーダー



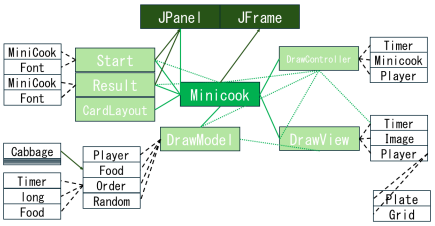
(e) ゲーム画面：加工前



(f) ゲーム画面：加工後



(g) ゲーム画面：組み合わせ後



(h) ゲーム画面：提供

5 考察

予定していた以上のものが完成した。

6 感想

(米谷祐希)

(鈴木早紀)

(吉田陽音)

付録 1：操作マニュアル

(ストーリー)

キミはレストランのキッチンで働いているぞ！制限時間内にオーダー通りの料理を作れ！目指せ高得点 !!

(実行方法)

「Java MiniCook」でゲームが開始する。

(操作方法)

このゲームはキーボードでキャラクターを操作する。図 2 にキー操作を示す。W,S,A,D で上下左右を操作し、J で取る、K で置く、スペースキーでアクションを行う。

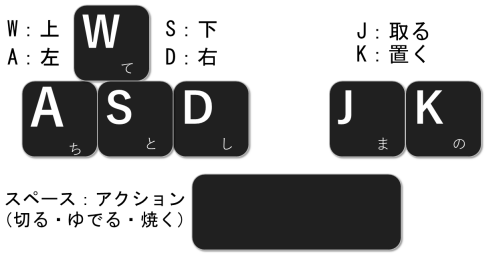


図 2: キーボード操作方法

(遊び方)

1. スタート

   スタートボタンを押すとゲームが開始する。

2. オーダーの確認

   まず、画面上部にランダムにオーダーが提示される。オーダーには、使う食材と調理方法が記載されている。各オーダーにはそれぞれ制限時間が設定されており、残り時間はオーダー上のゲージにリアルタイムに表示される。

3. 食材の調理

   次に、オーダーに記載されている食材を、各食材ボックスから取り出す。各食材を持ったまま、各調理器具の前でアクションボタンを押すことで、食材が加工される。

4. 料理の完成と提供

   料理は、加工された食材とお皿を組み合わせることで完成する。それらを組み合わせて料理ができあがれば、提供口に置くことで提供となり、オーダーと一致しているか判定される。一致していれば加点、間違っていれば減点となる。

5. リザルト

制限時間がなくなるとリザルト画面に遷移する。スコアとランクが表示される。リザルトを押せばもう一度ゲームが開始する。

- メニュー一覧
  - マグロ握り
  - イカ握り
  - 海鮮丼
  - カッパ巻
  - 鉄火巻き
  - サラダ
- 調理器具一覧
  - 包丁
  - 鍋
- 食材一覧
  - マグロ
  - イカ
  - 米
  - 海苔
  - キャベツ
  - トマト
  - キュウリ

付録 2：プログラムリスト

以下にプログラムリスト全体を記述する。

```
import javax.swing.*;
import java.awt.*;

class MiniCook extends JFrame {
   DrawModel model;
   DrawView view;
   DrawController cont;
   AudioManager audio;
   Result resultScreen;

   private CardLayout cardLayout;
   private JPanel cardPanel;

   public MiniCook() {
      System.out.printf("\n---Start---\n\n"); 見やすいように//
      model = new DrawModel();
      view = new DrawView(model);
      cont = new DrawController(model, view, this);
      audio = new AudioManager();

      model.getPlayer().setController(cont);
      model.getPlayer().setView(view);
      view.setController(cont);
      view.addKeyListener(cont);

      this.setBackground(Color.WHITE);
      this.setTitle("MiniCookey");
      this.setSize(1016, 950);
      this.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
      setLocationRelativeTo(null);

      // カードレイアウトの設定
      cardLayout = new CardLayout();
      cardPanel = new JPanel(cardLayout);

      // 各画面の追加
      Start startScreen = new Start(this);
      resultScreen = new Result(this);

      cardPanel.add(startScreen, "start");
      cardPanel.add(resultScreen, "result");

      // ゲーム画面
      JPanel gamePanel = new JPanel(new BorderLayout());
      gamePanel.add(view, BorderLayout.CENTER);

      cardPanel.add(gamePanel, "game");

      add(cardPanel);
      cardLayout.show(cardPanel, "start");
   }

   // スタート画面からゲーム画面に切り替える
   public void startGame() {
      cardLayout.show(cardPanel, "game");
      cont.startGame();
      //audio.playBGM("./sound/music_background2.wav");

      // キーボード入力を受け取るためにフォーカスを設定
      view.requestFocusInWindow();
   }

   // ゲーム終了時にリザルト画面を表示する
   public void showResult() {
      audio.stopBGM();
      System.out.println リザルト画面を表示します。("");
      resultScreen.updateScore(model.score);
      cardLayout.show(cardPanel, "result");
   }

   // リザルト画面からもう一度プレイ
   public void restartGame() {
      audio.playBGM("./sound/music_background2.wav");
      model.reset(); // ゲームデータをリセット（必要なら実装）
      startGame(); // ゲームを開始
   }

   public static void main(String[] args) {
      new MiniCook().setVisible(true);
   }
}
```