令和 元年度 卒業論文

# OpenModelica のシミュレーションを用いたモータ特性表自動生成ツールの試作

指導教員　片山 徹郎 教授

宮崎大学 工学部 情報システム工学科

原田　海人

**2020 年 1 月**

# 目次

# 第 1 章

# はじめに

本論文の構成は、以下の通りである。

第 2 章では、試作したモータ特性表自動生成ツールを開発するために必要となる前提知識について説明する。

第 3 章では、試作したモータ特性表自動生成ツールの機能について説明する。

第 4 章では、モータ特性表自動生成ツールの実装について説明する。

第 5 章では、試作したモータ特性表自動生成ツールの機能が正しく動作することを検証する。

第 6 章では、試作したモータ特性表自動生成ツールについて考察する。

第 7 章では、本論文のまとめと今度の課題を述べる。

# 第 2 章

# 研究の準備

本章では、本研究で必要となる前提知識を説明する。

## 2.1 モータ作成

### 2.1.1 仕様書

### 2.1.2 シミュレータの役割

## 2.2 モータ特性表

### 2.2.1 特性表の種類

### 2.2.2 特性表の要素

## 2.3 OpenModelica

## 2.4 modelica

# 第 3 章

# 機能

この章では、本研究で試作したモータ特性表自動生成ツールの機能について説明する。
モータ特性表自動生成ツールは、OpenModelica でモータのモデルをシミュレーションした時に出力される csv ファイルを読み込み、実行することによって、モータ特性表を生成する。

## 3.1 対応するモデル

今回試作したモータ特性表自動生成ツールでは以下の modelica モデルに対応する。

- モータ単体のモデル
- モータ単体のモデルを一つに統一したモデルを使用するモデル

以降、具体的に説明する。

### 3.1.1 モータ単体のモデル

モータ単体のモデルとは、

### 3.1.2 モータモデルを一つにまとめたモデル

## 3.2 特性表生成

# 第 4 章

# 実装

## 4.1 シミュレーション結果解析機能

## 4.2 特性表生成機能

# 第 5 章

# 適用例

本章では、本研究で作成した

## 5.1 モータ単体のモデル

## 5.2 パッケージ化されたモデル

# 第 6 章

# 考察

本論文では、ゲーム開発におけるテスト効率の向上を目的として、テスト支援ツールを試作した。テスト支援ツールの、オブジェクトとプレイヤーの重なり判定および自動リプレイ機能を利用することによって、ゲーム制作時の支援を行う。テスト支援ツールは、次に示す 10 個の機能を持つ。

- キャラクター情報入力機能
- オブジェクト増減機能
- オブジェクト情報入力機能
- オブジェクト位置確認機能
- ゲームスタートボタン機能
- リプレイボタン機能
- Warning 判定出力機能
- 入力キーとタイミングの記録機能
- リプレイ機能
- エラーメッセージ出力機能

本論文で試作したテスト支援ツールは、5 章の適用例で示したように正しく動作することが確認できた。以降、本論文で作成したテスト支援ツールについて考察する。

## 6.1 評価

### 6.1.1 評価方法

### 6.1.2 結果

本論文で試作したテスト支援ツールは、2D ゲームのプレイヤーと四角形のオブジェクトの重なり判定を行い、自動でファイルに出力できる。また、1 度プレイしたゲームを自動でリプレイできる。

テスト支援ツールを使用しない場合は、チーム開発の際のテストの報告時に報告書や口頭による説明では人為的なミスが生じてしまう可能性がある。また、1 度行ったテストを再現するために同じようにゲームをプレイするのは、手間と時間がかかる。

テスト支援ツールを使用した場合は、オブジェクトの重なり判定を自動で行い、自動でファイルに報告書として保存する。そのため、チームへのテスト結果を共有する際に、人為的なミスが生じる可能性を減らすことができる。それに加え、入力履歴を保存しているリプレイファイルをツールに読み込むことにより、自動で何度もゲームのリプレイが可能である。それにより、同じ条件にするために何度もプレイする手間がかからないので、時間の削減になる。

以上のことから、試作したテスト支援ツールを利用することによって、ゲーム制作時の支援を行うことができると言える。

## 6.2 関連研究

関連研究について述べる。

Selenium は、Web アプリケーションの画面操作を自動化するツールである [16]。Web ブラウザ上でのクリックやキーボード操作を記録し、一度記録しておけば再度実行するだけで新たに発生したエラーやバグを発見することができる。Selenium は Web ブラウザのみに対応しており、本研究の適用例で使用した exe ファイルを再生することはできないため、Windows 上で動作するゲームを対象としたテストは行えない。試作したテスト支援ツールは、exe ファイルを実行することができるので、ゲームのリプレイを行うことができる。

また、チェスにおいて、ピースの位置とピースの色、および種類を検出する研究がある [17]。検出方法には、テンプレートマッチングを用いている。この研究では、チェスに特化した手法を提案している。これに対して本研究では、1 つのゲームのみを対象としたものではなく、3 章で示した範囲のゲームであれば、ゲームに応じたキャラクターを設定することにより、複数のゲームに対して動作可能である。

## 6.3 ツールの問題点

以下に、今回作成したテスト支援ツールの問題点を示す。

- 実際に動かしたゲームとリプレイに誤差が生じる

  現段階では、ゲームをプレイする時とリプレイする時では、キャラクターの動きに誤差が生じてしまう (図 5.15、5.16 参照)。ゲームをプレイする際は、ウィンドウの取得、パーティクルフィルター、キー入力の取得などの処理を逐次実行しているため、キー入力の取得間隔が広くなってしまう。そのため、実際にキーを入力した時刻とキー入力を取得した時刻に誤差が生じてしまい、キャラクターの動きにも誤差が生じてしまうと考える。誤差を少なくする方法としては、ゲームをプレイする際のキー入力の取得処理を並列実行することにより、誤差を減らすことが可能と考える。

- オブジェクト毎の確認画面の未実装

  現段階では、オブジェクト全体の確認画面はあるが、1 つ 1 つのオブジェクトに対して確認する画面が未実装である。オブジェクト情報入力ウィンドウに新しくボタンを追加し、4.4 節の実装と同様の実装を行うことによって、実現可能と考える。

- Warning の種類が少ない

  現段階では、Warning として出力する対象は、キャラクターと動かない四角形のオブジェクトの重なり判定のみである。キャラクターと動くオブジェクトの重なり判定や、キャラクターの画像の種類の判別などは、現在のテスト支援ツールでは対応できていない。例えばキャラクターと動くオブジェクトの重なり判定を Warning として出力する場合は、パーティクルフィルターを複数対応させることによって、実現可能と考える。

- 移動速度の速いキャラクターを検出することが難しい

  現段階では、キャラクターの移動速度が速すぎるとパーティクルがキャラクターを検出するまでに時間がかかってしまう。そのため、重なり判定に影響が出てしまう。検出時間を短縮するためには、現在 300 個に設定しているパーティクル数を増やすことによって、実現可能だと考える。

- 複数キー入力に対応していない

  現段階では、入力ボタンとタイミングの記録機能およびリプレイファイルにおいて、複数キー入力に対応できていない。そのため、複数キーを同じタイミングで使うゲームには適用できない。複数キー入力に対応するためには、入力ボタンとタイミングの記録機能で複数キーの取得を行い、リプレイ機能もそれに応じた改良を行うことによって、実現可能だと考える。

# 第 7 章

# おわりに

本論文では、ゲーム開発におけるテスト効率の向上を目的として、テスト支援ツールを試作した。テスト支援ツールの、オブジェクトとプレイヤーの重なり判定および自動リプレイ機能を利用することによってゲーム制作時の支援を行う。テスト支援ツールは、次に示す 10 個の機能を持つ。

- キャラクター情報入力機能
- オブジェクト増減機能
- オブジェクト情報入力機能
- オブジェクト位置確認機能
- ゲームスタートボタン機能
- リプレイボタン機能
- Warning 判定出力機能
- 入力キーとタイミングの記録機能
- リプレイ機能
- エラーメッセージ出力機能

本論文で試作したテスト支援ツールは、5 章の適用例で示したように正しく動作することが確認できた。

本論文で試作したテスト支援ツールは、2D ゲームのプレイヤーと四角形のオブジェクトの重なり判定を行い、自動でファイルに出力できる。また、1 度プレイしたゲームを自動でリプレイできる。

テスト支援ツールを使用しない場合は、チーム開発の際のテストの報告時に報告書や口頭による説明では人為的なミスが生じてしまう可能性がある。また、1 度行ったテストを再現するために同じようにゲームをプレイするのは、手前と時間がかかる。

テスト支援ツールを使用した場合は、オブジェクトの重なり判定を自動で行い、自動でファイルに報告書として保存する。そのため、チームへのテスト結果を共有する際に、人為的なミスが生じる可能性を減らすことができる。それに加え、入力履歴を保存しているリプレイファイルをツールに読み込むことにより、自動で何度もゲームのリプレイが可能である。それにより、同じ条件にするために何度もプレイする手間がかからないので、時間の削減になる。

以上のことから、試作したテスト支援ツールを利用することによって、ゲーム制作時の支援を行うことができると言える。

以下に、テスト支援ツールの今後の課題を示す。

- 実際に動かしたゲームとリプレイに誤差への対応

  現段階では、ゲームをプレイする時とリプレイする時では、キャラクターの動きに誤差が生じてしまう (図 5.15、5.16 参照)。誤差を少なくする方法としては、ゲームをプレイする際のキー入力の取得処理を並列実行することにより、誤差を減らすことが可能と考える。

- オブジェクト毎の確認画面の実装

  現段階では、オブジェクト全体の確認画面はあるが、1 つ 1 つのオブジェクトに対して確認する画面が未実装である。オブジェクト情報入力ウィンドウに新しくボタンを追加し、4.4 節の実装と同様の実装を行うことによって、実現可能と考える。

- Warning の種類を増やす

  現段階では、Warning として出力する対象は、キャラクターと動かない四角形のオブジェクトの重なり判定のみである。キャラクターと動くオブジェクトの重なり判定や、キャラ

クターの画像の種類の判別などは、現在のテスト支援ツールでは対応できていない。例えばキャラクターと動くオブジェクトの重なり判定を Warning として出力する場合は、パーティクルフィルターを複数対応させることによって、実現可能と考える。

- 速度の速いプレイヤーを検出する

  現段階では、キャラクターの移動速度が速すぎるとパーティクルがキャラクターを検出するまでに時間がかかってしまう。検出時間を短縮するためには、現在 300 個に設定しているパーティクル数を増やすことによって、実現可能だと考える。

- 複数キー入力に対応する

  現段階では、入力ボタンとタイミングの記録機能およびリプレイ機能において、複数キー入力に対応できていない。複数キー入力に対応するためには、入力ボタンとタイミングの記録機能で複数キーの取得を行い、リプレイ機能もそれに応じた改良を行うことによって、実現可能だと考える。

# 謝辞

# 参考文献

[1] Martin Fowler(訳:児玉 公信, 友野 昌夫, 平澤 章, 梅沢 真史):”新装版 リファクタリング 既存のコードを安全に改善する”, オーム社 (2014).

[2] Craig Larman(訳:児高 慎治郎, 松田 直樹, 越智 典子):”初めてのアジャイル開発 スクラム、XP、UP、Evo で学ぶ反復型開発の進め方”, 日経 BP 社 (2004).

[3] IntelliJ IDEA - Wikipedia: https://ja.wikipedia.org/wiki/IntelliJ_IDEA, アクセス日:2020/01/05.

[4] プログラム構造インターフェース (PSI) / IntelliJ プラットフォーム SDK プラグイン開発ガイド: https://pleiades.io/intellij/sdk/docs/basics/architectural_overview/psi.html, アクセス日:2020/01/05.