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

近年、デジタルゲームの開発人口が増加傾向にあり [1]、ゲーム市場の競争率が上がっている [2] ことから、デジタルゲーム開発の大規模化、複雑化が進んでいる [3]。このような背景から、開発したゲームに対するアドホックテスト [4] を、開発チームとは別の複数人のテスターが行う場合が多い [5]。しかし、開発チームとテスターが異なる場合、以下の 2 つの問題点が存在する。

- テスターがバグを発見した際の、報告書や口頭による開発チームへの説明は人為的なミスが入り込む可能性がある
- テスターが発見したバグを開発チームが再現するために、テスターが行った入力を再現することは手間と時間がかかってしまう

本研究では、上記 2 つの問題点を解決することを目的として、ゲーム開発におけるテスト支援ツールを試作する。

試作するテスト支援ツールは、前者の問題点を解決するために、テストの報告書を自動生成する。また、後者の問題点を解決するために、入力履歴を自動で保存し、その保存した入力履歴を用いて自動で再現する。これら 2 つの解決策により、ゲーム開発における、テスト効率の向上を図る。試作するテスト支援ツールは、具体的に以下の 2 つの特徴を持つ。

- キャラクターとオブジェクトの重なり判定を自動で行い、報告書としてファイルに出力
- プレイしたゲームの自動リプレイ

ここで、キャラクターとは、ゲームにおいてユーザーが操作できる画像であり、オブジェクトとは、ユーザーが選択できるゲーム画面上の任意の大きさの四角形である。

本論文の構成は、以下の通りである。

第 2 章では、試作したテスト支援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、試作したテスト支援ツールの概観と機能について説明する。

第 4 章では、テスト支援ツールの実装について説明する。

第 5 章では、試作したテスト支援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、試作したテスト支援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、テスト支援ツールの概観とその機能について説明する。テスト支援ツールは、以下の 2 つの特徴をもつ。

- キャラクターとオブジェクトの重なり判定を自動で行い、報告書としてファイルに出力
- プレイしたゲームの自動リプレイ

なお、今回対象とするゲームは、Unity で制作した 1 画面のみで完結する 2D ゲームで、同時に複数のキーを入力しないものとする。今回対象とするキーは「A キー、S キー、W キー、D キー、スペースキー、左矢印キー、上矢印キー、右矢印キー、下矢印キー」である。また、実装の都合上、今回設定できるオブジェクトの数を 4 つまでとする。動作環境は Windows で、設定できるキャラクターの画像の拡張子は、png か jpeg とする。なお、設定するキャラクターの画像に似た画像を使用するゲームの場合は、今回は対象外とする。これは、重なり判定にパーティクルフィルターを用いることが原因で、誤判定が生じてしまうためである。

3.1 対応するモデル

テスト支援ツールの概観を、図 3.1 に示す。テスト支援ツールは、以下のエリアからなる。

- キャラクター情報設定部
- オブジェクト追加部

- データ入力部
- 実行部
- エラーメッセージ出力表示部

3.1.1 モータ単体のモデル

キャラクター情報設定部は、重なり判定させたいキャラクター名の表示と、そのキャラクターのキャラクター情報入力ウィンドウを新しく生成し、表示する。キャラクター情報設定部は、以下の 2 つの要素をもつ。キャラクター情報設定ボタンを押下すると、3.2.1 節に示すキャラクター情報入力機能を利用できる。

- キャラクター名
- キャラクター情報設定ボタン

3.1.2 パッケージ化されたモデル

キャラクター情報入力ウィンドウの概観を、図 3.2 に示す。キャラクター情報入力ウィンドウはキャラクター情報入力部と、キャラクター情報エラーメッセージ出力表示部で構成している。キャラクター情報入力部は、ゲームで追跡させたいキャラクターの画像と名前を入力できる。キャラクター情報エラーメッセージ出力表示部は、キャラクター情報入力部で設定していない情報があった場合、エラーメッセージを表示する。キャラクター情報入力部は、以下の 3 つの要素をもつ。

3.2 特性表生成

第 4 章

実装

テスト支援ツールの 10 個の機能である、「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」の実装について説明する。また、「Warning 判定出力機能」の実装のために、「ゲーム実行時のゲーム画面リアルタイム取得」、「キャラクターとオブジェクトの重なり判定」の実装について説明する。

4.1 シミュレーション結果解析機能

「キャラクター情報入力機能」の実装方法について説明する。キャラクター情報入力ウィンドウのキャラクター情報入力部では、キャラクター画像、キャラクター名、キャラクター情報保存を配置している。キャラクター画像と、キャラクター情報保存の処理を以下に示す。

4.2 特性表生成機能

「オブジェクト増減機能」の実装方法について説明する。オブジェクト追加部のオブジェクト増加ボタンを押下すると、オブジェクト名とオブジェクト情報設定ボタンをオブジェクト追加部に表示する。表示の処理を、以下に示す。また、オブジェクト名とオブジェクト情報設定ボタンを表示した状態を、図 4.1 に示す。

1. オブジェクト増加ボタンを押下すると、オブジェクト名とオブジェクト情報設定ボタンを生成する。この操作は 4 回まで行える。

2. オブジェクト名とオブジェクト情報設定ボタンの場所、サイズ、名前を設定する。

3. オブジェクト名とオブジェクト情報設定ボタンを 1 つ目以外を全て非表示にする。

4. 1.～3. はオブジェクト増加ボタンを最初に押下した時にのみ処理を行う。2 回目以降の場合は 5.、6. の処理を行う。

5. オブジェクト増加ボタンを押下すると、生成したオブジェクト名とオブジェクト情報設定ボタンを表示する。

6. オブジェクト減少ボタンを押下すると、表示しているオブジェクト名とオブジェクト情報設定ボタンを非表示にする。

「オブジェクト情報入力機能」の実装方法について説明する。オブジェクト増減機能で表示したオブジェクト情報設定ボタンを押下すると、オブジェクト情報入力ウィンドウが表示できる。オブジェクト情報入力部では、オブジェクト座標入力テキストボックス、オブジェクト名入力テキストボックス、オブジェクト情報保存ボタンを配置している。オブジェクト情報保存ボタンの処理を、以下に示す。

第 5 章

適用例

本章では、本研究で作成したテスト支援ツールが正しく動作することを検証する。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

この中の、「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」、「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」、「エラーメッセージの出力機能」の 8 個の機能について検証を行う。

なお、「ゲームスタートボタン機能」については、「Warning 判定出力機能」と「入力キーとタイミングの記録機能」を実行する前に押下するボタンの機能のため、これらが機能することで、正しく動作したとする。また、「リプレイボタン機能」については、「リプレイ機能」を実行する前に押下するボタンの機能のため、「リプレイ機能」が機能することで、正しく動作したとする。

5.1 モータ単体のモデル

テスト支援ツールにおいて「キャラクター情報入力機能」が機能することを検証するために、「キャラクター情報を入力する前のキャラクター情報入力ウィンドウ」と、「キャラクター情報を入力した後のキャラクター情報入力ウィンドウ」のキャラクター情報入力ウィンドウを比較する。キャラクター情報を入力する前のキャラクター情報入力ウィンドウを図 5.1 に、キャラクター情報を入力した後のキャラクター情報入力ウィンドウを図 5.2 に、それぞれ示す。また、入力したキャラクター名とキャラクターのファイルパスを保存している csv ファイルを、図 5.3 に示す。

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.