解説ドキュメント

平成 31 年 4 月 8 日

氏名：小島 一也

〒 214-0014
住所:神奈川県川崎市多摩区登戸 1082-2 メゾンルソレーユ 204 号
大学/学科/専攻:明治大学大学院/理工学研究科/情報科学専攻
電話番号:080-6962-0835
E-mail:kazu@cs.meiji.ac.jp

1 成果物のファイル/ディレクトリリスト

- src/Makefile 実行ファイルを生成する Makefile
- src/breakout.cpp 今回の提出作品
- document.pdf 解説ドキュメント

2 成果物について

2.1 概要

画像処理ライブラリの OpenCV を使ってブロック崩しを作成しました。画面は 2 値画像で表示し、操作はキー入力から行うことができます。

2.2 実行環境

- ubuntu 16.04
- g++
- OpenCV 2.4.11

2.3 実行手順

1. OpenCV 2 以上をインストールする
2. src ディレクトリ内で'make breakout' コマンドを実行して実行ファイルを生成する
3. './breakout' コマンドでゲームが起動する
4. 端末上で's' キーを入力するとゲームが開始する

3 作成環境

3.1 OS

ubuntu 16.04

3.2 エディタ

Emacs

3.3 言語

C++

4 作成期間

2 週間

5 作成メンバー数と作業分担

個人で制作しました。

6 作業担当箇所の解説

まず、作成したクラスの説明をします。今回、ブロック崩しを実装するために、Object クラス、Player クラス、Block クラス、Ball クラスを作成しました。Object クラスは、オブジェクトの幅、高さ、位置を変数に持ち、初期位置を決定するコンストラクタと描画関数を持つ各クラスの親クラスです。これらのメンバは、どのクラスでも必要であるため、各クラスはこれを継承して作ります。Player クラスには、親クラスのメンバ以外にキー入力を受けてからの処理を加えます。Block クラスは、クリア判定に用いるためにブロックが消されたかどうかの情報を持ちます。Ball クラスは、ボールのベクトル情報や衝突判定の関数を持ちます。これらのクラスを使ってメイン関数で実装します。まず、初期画面を生成したら、スタートボタンの入力待ち状態になります。スタートボタンを押すとゲームが始まり、ボールやキー入力によってゲーム内の情報が変化し、それに応じてゲーム画面を更新します。ボールが画面の下部に接触してしまった場合はゲームオーバー、ブロックをすべて消すことができた場合はゲームクリアとなります。

7 制作する上で参考にしたものの列挙

- C++リファレンス (https://ja.cppreference.com/w/cpp)
- Unix 環境におけるリアルタイムキー入力 (http://i2blog.matrix.jp/index.php?UID=1479357418)

8 成果物に対しての自身の評価

良く言えば堅実な、悪く言えばオリジナリティに乏しい作品となってしまったと思います。今回、プログラムで一からゲームを作るのが初めてであったこともあり、バグに気を付けながら丁寧に作成しました。そのため、他者が読んでも理解しやすいプログラムを書くことができたと考えています。また、衝突判定やそれに伴うベクトルの決定などの実装は不自然のないように意識したのですが、その結果全体的に普遍的な実装となってしまい、面白みにかけるゲームとなってしまったと感じました。今回は時間が足りなくなってしまって実装できませんでしたが、せっかく OpenCV を使っているので、ゲームオーバー時にモルフォロジー処理をかけて徐々に画面を真っ暗にするなどして、もっと演出にこだわるべきであったと思いました。また、今回作成したクラスは、次にゲームを作成する際に再び必要になる可能性が非常に高いため、クラスごとにファイルを分けて include できるように再利用性を高める必要があったと感じました。

9　成果物に対しての周りの評価

今回、この作品を研究室の同期や後輩に遊んでもらったところ、操作に慣れるまで時間がかかるという指摘を多くいただきました。この作品では、右移動を開始するボタン、移動を停止するボタン、左移動を開始するボタンの 3 種類を設定していましたが、右移動の ON/OFF ボタン、左移動の ON/OFF ボタンの 2 種類のほうが、ボタンの数も少なく、UFO キャッチャーと似ていることもあり馴染みやすかったかもしれないと思いました。

また、ボールのベクトルの角度が、水平に対して 45 度で一定であるため、どうしても同じような軌道を描きやすく、プレイヤー側の工夫が反映できるような仕組みのほうが面白そうだという指摘をいただきました。今後は、45 度方向だけでなく、30 度や 60 度方向にも飛ばせるようにしたいです。また、角度の変化はプレイヤーとボールが触れたタイミングのプレイヤーの動きから反映させたいと考えています。