Московский государственный университет имени М. В. Ломоносова
факультет вычислительной математики и кибернетики
кафедра алгоритмических языков

Отчёт о выполнении задания практикума

Игра "Арканоид"

Выполнил: студент 325 группы
Толкачев Денис Александрович

Москва 2021

Содержание

1 Постановка задачи

Требуется реализовать базовую версию классической игры "Арканоид"при помощи средств языка Haskell и графической библиотеки Gloss.

1.1 Правила игры

Игрок контролирует небольшую платформу-ракетку, которую можно передвигать горизонтально от одной стенки до другой, подставляя её под шарик, предотвращая его падение вниз. Удар шарика по кирпичу приводит к уменьшению здоровья кирпича. Изначально кирпич имеет 3 единицы здоровья, поэтому ровно 3 удара мяча необходимо для полного разрушения кирпича. Игра будет окончена после того, как все кирпичи (24 шт.) будут уничтожены. Удар шарика об дно игрового поля означает поражение.

1.2 Базовые требования

Необходимо реализовать следующие игровые элементы и механики:

1. Отрисовка игрового поля,
2. Отрисовка сетки из кирпичей,
3. Отрисовка игрового мяча,
4. Отрисовка платформы-ракетки,
5. Обработка ввода с клавиатуры для управления игровой платформой и для завершения/перезапуска игрового процесса,
6. Обработка движения платформы,
7. Обработка столкновения мяча с кирпичами,
8. Обработка столкновения мяча с игровым полем,
9. Обработка столкновения мяча с ракеткой,
10. Подсчет очков,
11. Определение проигрыша и успешного прохождения игры,
12. Выход из игры,
13. Перезапуск игры.

2 Модули проекта

Проект состоит из следующих модулей:

- BallLib.hs - функции, связанные с движением мяча;
- BricksLib.hs - отрисовка кирпичей и обработка столкновения с ними мяча;

- Constants.hs - модуль с константами;
- GamePlay.hs - вспомогательные функии;
- PlatformLib.hs - обработка движения платформы и столкновения с ней мяча;
- Run.hs - основной цикл игры;
- Structures.hs - пользовательские типы данных.

В модуле BallLib.hs реализованы следующие функции:

- flipDirectionVerticaly - отражает направление движения мяча относительно оси OY;
- flipDirectionHorizontaly - отражает направление движения мяча относительно оси OX;
- isCollideWalls - проверяет столкновение мяча со стенами;
- resolveCollide - обрабатывает столкновение мяча с разными объектами;
- applyPosition - изменяет позицию мяча, применяя новое направление;
- changeBallPosition - главная функция, использующая все указанные выше функции для комплексной обработки столкновения мяча с игровыми объектами и подсчёта новой позиции на игровом поле.

В модуле BricksLib.hs реализованы следующие функции:

- initBricks - инициализирует начальную сетку кирпичей;
- drawBricks - отрисовывает сетку кирпичей;
- getColorOfBrick - вычисляет цвет кирпича в зависимости от остатка его здоровья;
- flipDirectionVerticalyBrick - отражает направление движения мяча относительно оси OY. Отличие от функции из модуля BallLib.hs заключается в типах принимаемых и возвращаемых параметров;
- flipDirectionHorizontalyBrick - отражает направление движения мяча относительно оси OX. Отличие от функции из модуля BallLib.hs заключается в типах принимаемых и возвращаемых параметров;
- collideBrick - проверка столкновения мяча с конкретным кирпичом;
- insertBrick - добавляет кирпич в список кирпичей. Принимает на вход аргумент типа Maybe Brick, поэтому умеет обрабатывать ситуацию, когда на вход пришло Nothing;
- checkBricks - перебор каждого кирпича из сетки и проверка их функцией collideBrick;

- resolveBricksCollision - главная функция, использующая все указанные выше функции для комплексной обработки столкновения мяча с сеткой кирпичей.

В модуле Constants.hs определены функции для получения константных значений. Для удобства будем указывать название функции и то, что она возвращает:

- getBallColor - цвета мяча;
- getPlatformColor - цвет платформы;
- platformWidth - длина платформы;
- platformWidthDiv2 - половины длины платформы (такое использование обусловлено оптимизацией вычислений);
- platformHeight - высота платформы;
- platformHeightDiv2 - половина высоты платформы;
- platformSpeed - скорость движения платформы;
- ballRadius - радиус мяча;
- ballSpeed - скорость мяча;
- brickWidth - длина кирпича;
- brickWidthDiv2 - половина длины кирпича;
- brickHeight - высота кирпича;
- brickHeightDiv2 - половина высоты кирпича.

В модуле GamePlay.hs объявлена всего одна функция:

- gameOverFunc - обрабатывает ситуацию проигрыша.

В модуле PlatformLib.hs определены следующие функции:

- drawPlatform - отрисовывает платформу;
- changePlatformPosition - изменяет позицию платформы;
- calculateDirection - вычисляет новое направление мяча после столкновения с платформой;
- collidePlatform - проверка столкновения мяча с платформой;

- resolvePlatformCollision - главная функция, обрабатывающая столкновение мяча с платформой и вычисляющая новое направление при помощи вышеуказанных функций из этого модуля.

В модуле Run.hs определены следующие функции:

- initState - инициализирует базовое состояние игры;
- drawApp - отрисовка всех игровых элементов;
- handleEvent - обработка нажатий клавиш;
- updateFrame - обработка каждого кадра;
- run - основной цикл игры.

В модуле Structures.hs определены следующие типы данных:

- Direction - направление движения платформы;
- CollideObject - типа объекта, с которым столкнулся мяч;
- Brick - объект для кирпича, содержащий поля:
  - position - позиция кирпича;
  - hp - количество здоровья у кирпича.
- GameState - объект игрового состояния, содержащий поля:
  - score - счёт;
  - ballPosition - позиция мяча;
  - ballDirection - направление движения мяча;
  - platformPosition - позиция платформы;
  - bricks - список кирпичей;
  - pause - флаг паузы;
  - gameOver - флаг проигрыша;
  - gameStarter - флаг начала игры;
  - gameFinished - флаг окончания игры;
  - leftKeyPressed - флаг нажатия левой стрелки;
  - rightKeyPressed - флаг нажатия правой стрелки.
- HitBrick - вспомогательные объекты, использующиеся при проверке на столкновение с кирпичом. Содержат поля:
  - hit - флаг столкновения;

  - direction - направление после столкновения;
  - brick - содержит данные о кирпиче.

- CollideBricksReturn - вспомогательный тип данных, использующийся при проверке на столкновение с сеткой кирпичей. Содержит поля:
  - newBricks - список кирпичей после проверки на столкновение (кирпич после столкновения может разрушиться);
  - state - состояние игры после столкновений.

3 Используемые библиотеки

При реализации использовались следующие библиотеки:

- gloss - графический интерфейс;
- Cabal - система сборки проектов, написанная на Haskell;
- yaml - чтение конфигурационных файлов.

4 Сценарий игры

При первоначальном запуске игрок получает 0 очков. На экране отрисовываются игровые объекты (сетка кирпичей, платформа, мяч). Мяч и платформа зафиксированы в стартовой позиции. См. *Рис. 1*

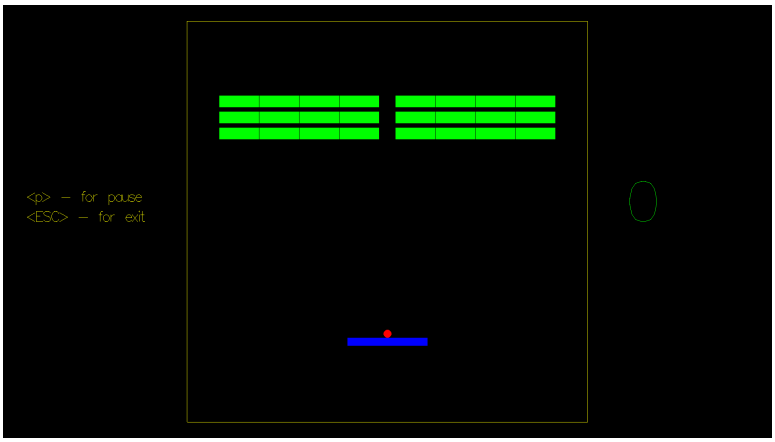

Рис. 1: Начало игры

Для начала игры необходимо нажать на клавишу <Пробел>. Мяч оттолкнется от платформы и полетит. В случае необходимости, игру можно поставить на паузу, нажав на клавишу <p> - от англ. *Pause*. По ходу игры пользователь сможет повлиять на траекторию полета мяча при помощи платформы, подставляя её под мяч. Для перемещения платформы влево нужно использовать на клавиатуре клавишу <Левая стрелка>, а для

перемещения вправо - <Правая стрелка>. Мяч, ударяясь о кирпич, отнимает у него единицу здоровья (у кирпича 3 единицы здоровья). Окраска кирпича зависит от его остатка здоровье (3 ед. - зеленый, 2 ед. - жёлтый, 1 ед. - красный). Если у кирпича заканчиваются единицы здоровья, то он исчезает, добавляя к сумме очков игрока единицу. См. *Рис. 2*

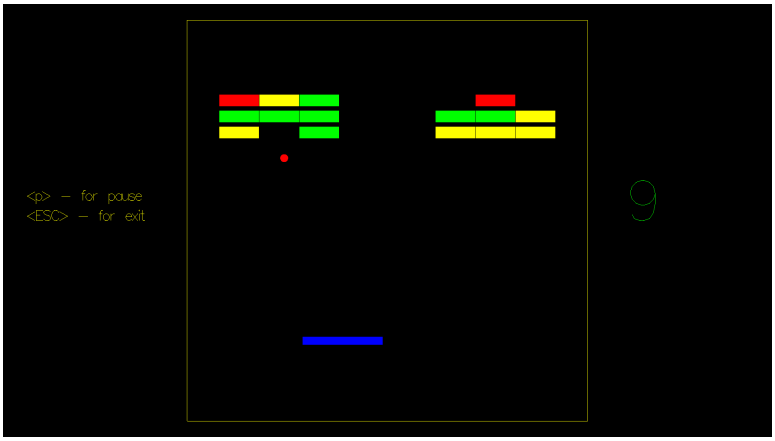

Рис. 2: Игровой процесс

Если игрок допустит удар мяча об дно игрового поля, то игровой процесс будет остановлен. Игрок увидит на экране сообщение о поражении. См. *Рис. 3*

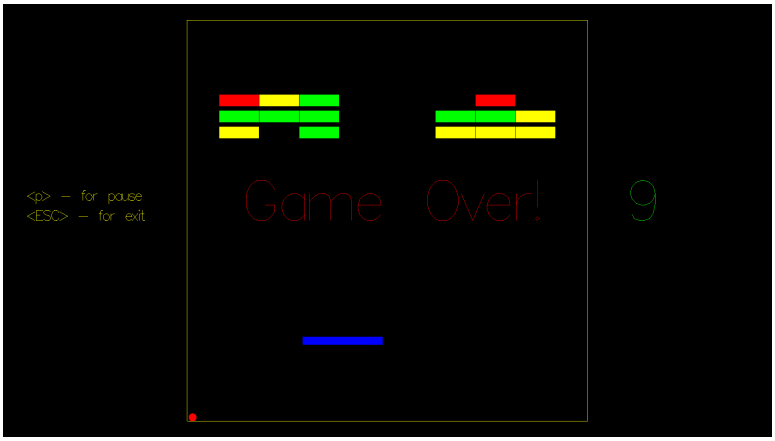

Рис. 3: Проигрыш

В том случае, если игроку удастся уничтожить все кирпичи и набрать 24 очка, то игра считается успешно пройденной. Игрок увидит сообщение о завершении игры на экране. См. *Рис. 4*

Рис. 4: Выигрыш

Стоит заметить, что игрок в любой момент может перезапустить игру, нажав на клавишу <r> - от англ. *Restart*.

На этом описание игрового процесса можно считать **законченным**.