Московский государственный университет имени М. В. Ломоносова
Факультет вычислительной математики и кибернетики
Кафедра алгоритмических языков

Отчёт о выполнении задания практикума

«Разработка игры Arkanoid»

Студенты 324 группы
Б. С. Коротков
Н. О. Рожков

Москва, 2019

1 Постановка задачи, основной функционал приложения

1.1 Описание

Арканоид — многоуровневая игра в жанре аркады. Игрок контролирует платформу, которую можно передвигать горизонтально от одной стенки до другой, подставляя её под шарик, предотвращая его падение вниз. На другой стороне игрового поля находятся кирпичи, которые необходимо разбить шариком для успешного прохождения уровня. После того как все кирпичи на данном уровне уничтожены, происходит переход на следующий уровень, с новым набором кирпичей.
Если игрок роняет шарик, он вынужден начать прохождение уровня заново, потеряв при этом все свои очки. В любой момент игрок может перейти на следующий уровень, сохранив набранное им количество очков при помощи клавиши "n" . Переход по уровням при помощи цифр “1” - “7” обнуляет счётчик.

1.2 Базовая часть

Приложение:

1. Предоставляет возможность играть в игру с фиксированными уровнями, используя графический интерфейс;
2. Включает в себя 7 различных уровней и 4 бонуса (в том числе и секретный);
3. Фиксирует прохождение уровней, подсчитывает количество очков игрока, реализует корректный сценарий при поражении.

1.3 Индивидуальная часть (Коротков Борис)

К игровым возможностям добавлены:

1. Обновляющаяся в режиме реального времени **таблица рекордов**. Чтобы записать свой результат, пользователю необходимо нажать клавишу "r" , после чего он может увидеть изменение в таблице. Чтобы очистить всю таблицу - необходимо нажать клавишу "c" .
   Написаны:
   - Функция, которая записывает в файл профиля игрока его количество очков.
   - Функция, читающая изменения из файла и выводящяя их в таблицу рекордов.
   - Функция, очищающая таблицу рекордов.
2. **Сохранение и загрузка игровых сессий**. Для того, чтобы сохранить свой прогресс, пользователю необходимо нажать клавишу - "o" . Чтобы продолжить игру с точки сохранения - нажать клавишу "l" .
   Написаны:
   - Функция загрузки игровой сессии, которая читает из файла строку.
   - Функция разбора данной строки по разделительным символам.
   - Функция чтения из файла информации и заполнения параметров игрового состояния.

1.4 Индивидуальная часть (Рожков Никита)

К расширению игры добавлены:

1. **Ограничение прохождения уровня по времени.**

   - Для ограничения прохождения уровня по времени я добавил несколько полей в состояние игры, для того чтобы отследить момент когда нажали старт игры, и вторая чтобы смотреть сколько сейчас времени и из их разницы узнать сколько времени прошло. Так же добавил поле которое для каждого уровня хранит информацию от том сколько времени на этот уровень дано. Конкретных функций не реализовывал, а вносил изменения в функцию реагирующую на нажатие 's', на клавиши переключения уровней и т.д.

2. **Постепенное увеличение скорости движения шарика в рамках уровня.**

   - Для постепенного увеличения скорости я добавил поле в структуру, на которое умножатся положение шара.

3. **Бонусные баллы за уровень в зависимости от времени прохождения**

   - Для добавления бонусов за быстрое прохождение я воспользовался тем что уже добавил в пункте 1 и просто поставил условие когда нажимают клавишу 'n' что если время потраченное на прохождение меньше половины максимального времени = +5 балов, за половина времени +5 =3 баллам

2 Структура проекта

2.1 Модули, их взаимосвязь

Исходный код находится в следующий файлах:

- **Main.hs** - программа, получающая на вход данные от пользователя и запускающая игру.
- **MyProj.hs** - модуль, который отрисовывает игровое поле и отвечает за физику игры.
  - *drawPicture* - функция, отображающая какой-либо объект.
  - *drawBall* - функция, отображающая наш шар.
  - *render* - функция отображения интерфейса.
  - *drawScore* - функция, отображающая счёт.
  - *moveBall* - функция перемещения шара.
  - *Функции, в названия которых входит слово "Collision"* - реализуют отталкивание шара от поверхности.
  - *deletePlat* - функция удаления платформы.
  - *paddleBounce* - функция отталкивания шара от платформы игрока.
  - *ckeckGameOver* - функция проверки состояния поражения игрока.
  - *update* - функция обновления игрового состояния.
  - *runMyProj* - функция запуска игры.
- **Data.hs** - модуль с описаниями основных типов данных и начального состояния игры.
  - *data PongGame* - тип данных, описывающий состояние игры.
  - *initialState* - функция инициализации начального состояния игры.
  - *width, heigth* - размеры игровой области.
  - *background, window* - константы для функции playIO.
  - *Score, Radius, Position* - переименование типов.
- **HandleKeys.hs** - модуль для управления игрой при помощи клавиатуры.
  - *handleKeys* - функция управления игрой при помощи клавиатуры.
  - *parseStr* - функция разбора строки и её преобразования в список параметров игрового состояния.
  - *randVel* - функция получения случайного вектора скорости.
  - *buildPlatforms* - функция получения новых координат платформ.
  - Чтобы ознакомиться со всеми клавишами управления, откройте меню внутри игры.

- **ImgWork.hs** - модуль для работы с изображениями.
  - *data Images* - тип данных, хранящий все изображения.
  - *loadImages* - функция загрузки изображений.

Взаимосвязь модулей:

1. **Main** импортирует модуль MyProj и Img, чтобы загрузить картинки и пользовательский интерфейс.
2. **MyProj** - получает описание типов данных из модуля Data и функцию упрвления игрой при помощи клавиш из модуля HandleKeys.
3. **Data** и **HandleKeys** - являются вспомогательными модулями.

2.2 Проект

- В папке app - находится Main.hs. Папка src - содержит все необходимые модули.
- Папка saves - хранит все сохранённые игровые сессии пользователей.
- Папка images - содержит все картинки, использованные в игре.
- Файлы p*.txt - содержат игровую статистику (кол-во очков в таблице рекордов) пользователей.

3 Используемые библиотеки

- ***Graphics.Gloss*** - для отображения графического интерфейса.
- ***System.Random*** - для получения случайного направления полёта шарика и цвета платформ.
- ***Data.Time*** - для ограничения прохождения уровня по времени.
- ***System.IO*** - для работы с файлами.
- ***Text.Read*** - для преобразования типов.
- ***Graphics.Gloss.Juicy*** - для добавления изображений.
- ***System.Exit*** - для выхода по клавише "Esc" .

4 Сценарии работы с приложением и примеры использования

Проект можно собрать и запустить при помощи команды **stack build && stack exec project-template-exe**.

При запуске приложения игрок попадает на первый уровень.Чтобы начать игру он может нажать клавишу "s" или выбрать любой другой уровень при помощи цифр от "1" до "9" и также начать данную калвишу. После этого шар полетит в случайно направлении, и игроку необходимо будет отбивать его при помощи платформы, которой можно управлять, используя клавиши "a" и "d" .
После того, как игрок уничтожит все платформы - у него появится выбор: нажать "n" и перейти на следующий уровень со всеми набранными очками, или же используя цифры поменять уровень на любой другой.

Если игрок не успевает поймать шар, игра заканчивается. Он может переиграть уровень, нажав "s" или же перейти на любой другой.
Также, если выходит время, выделенное на прохождение уровня - игра заканчивается и на экране появляется информационное сообщение.
За быстрое прохождение предусмотрено вознаграждение, в виде дополнительных очков.

На некоторых уровнях есть бонусы, которые можно сбить и получить дополнительное преимущество.