Московский государственный университет имени
М. В. Ломоносова

Факультет вычислительной математики и кибернетики

Кафедра алгоритмических языков

Отчет о выполнении задания практикума

«Однопользовательский платформер»

Студент 325 группы
Г. В. Хорошилов

Москва, 2019

1 Постановка задачи, функционал игры

1.1 Описание игры

Платформер — это жанр компьютерных игр, в которых основной чертой игрового процесса является прыгание по платформам, лазанье по лестницам, собирание предметов, обычно необходимых для завершения уровня.

Конечный платформер — несколько уровней с препятствиями, после успешного прохождения которых игроку показывается экран с оповещением об успешном завершении игры. При непрохождении препятствий персонаж умирает, и уровень начинается заново.

1.2 Базовая часть

- 5 уровней с возрастающей сложностью;
- отрисовка игрового процесса;
- управление персонажем (перемещение/прыжок);
- определение момента поражения и прохождения уровня.

1.3 Помимо базовой части реализовано

- анимация персонажа при движении;
- привязка разных анимаций к направлению движения;
- разная гравитация (скорость падения персонажа);
- ускорение персонажа при передвижении (до лимита).

Последние два пункта влияют на прохождение игры.

2 Структура проекта

Подавляющее большинство кода проекта располагается в директории `/src` и поделено на три части: движок игры, уровни и рендерер. В этой же директории находится файл `Orphne.hs`, объединяющий все модули.

- **Engine** — директория, содержащая движок игры:
  - `CommonTypes.hs` — описание основных типов;
  - `Constants.hs` — константы, относящиеся к движку;
  - `Engine.hs` — сам движок;
- **Levels** — директория, содержащая уровни:
  - `Level[1|2|3|4|5].hs` — описание начального состояния уровня;
- **Renderer** — директория, содержащая графический движок игры:
  - `Renderer.hs` — графический движок игры;
  - `TextureLoader.hs` — загрузка текстур;
  - `VisualConstants.hs` — константы, относящиеся к визуальной части игры;
- `Orphne.hs` — файл с вызовом всех основных функций из других модулей.

В свою очередь код файла `Orphne.hs` вызывается в файле `Main.hs`, располагающемся в другой директории (`/app`). В директории `/img` находятся изображения, загружаемые в качестве текстур объектов.

2.1 Директория Engine

В файле `CommonTypes.hs` содержатся следующие типы:

- `Position` — описывает точку в 2D пространстве игры;
- `PlatformSize` — размер хитбокса платформы;
- `GameState` — состояние игрового мира в каждый момент времени;
- `Platform` — описывает платформу;
- `Goal` — позиция двери (конца уровня);
- `PlatformType` — вспомогательный тип для платформы, задает характеристику для дальнейшего взаимодействия с игроком:
  - `Pltfrm` — обычная платформа, запрыгнуть можно только сбоку;
  - `Earth` — земля также является платформой;
  - `Spike` — шипы, при столкновении с которыми персонаж умирает;
  - `Cloud` — платформа, можно запрыгнуть с любой стороны;
- `Player` — описывает персонажа в каждый момент времени игры.

Также в этом файле содержится функция `pConstruct` преобразования последовательности данных в объект типа `Platform`.

В файле `Constants.hs` содержатся следующие константы:

- `maxLVL` — количество уровней в игре;
- `gravity` — коэффициент ускорения при падении;
- `jumpVel` — начальная скорость прыжка;
- `velocity` — ускорение при движении вбок;
- `maxSideSpeed` — верхняя граница скорости персонажа;
- `sideSpeed` — начальная скорость движения вбок;

– `playerImgSize` — хитбокс персонажа;

– `playerHeight` — высота хитбокса;

– `playerWidth` — ширина хитбокса;

– `halfPlayerHeight` — половина высоты хитбокса персонажа;

– `constPlatformThick` — высота хитбокса платформы;

– `constPlatformWidth` — ширина хитбокса персонажа;

– `earthLineAbsPos` — линия земли;

– `defaultWall` — границы перемещения персонажа;

– `thresholdTop` — на сколько необходимо персонажу сместиться вверх, чтобы «камера» поменяла свое положение;

– `thresholdBottom` — на сколько необходимо персонажу сместиться вниз, чтобы «камера» поменяла свое положение;

– `goalSize` — хитбокс двери.

В файле `Engine.hs` содержатся следующие функции:

– `handleInput` — обработка нажатия клавиш;

– `evalStats` — вспомогательная функция, возвращает 1 если оба переданных значения истинны, иначе 0;

– `playerMovement` — назначение действий на клавиши;

– `motion` — изменение позиций объектов каждый момент времени;

– `movePlayer` — изменение позиции персонажа;

– `collideObj` — проверка столкновений персонажа с объектами уровня;

– `isGrounded` — проверка приземления персонажа на платформу;

– `checkDeath` — проверка условия смерти (столкновение с шипами);

– `checkGoal` — проверка условия победы;

– `takeIf` — вспомогательная функция, возвращает второй переданный параметр если первый – истина, иначе 0;

– `update` — функция перехода на новый тик игрового мира.

2.2 Директория Levels

В каждом из файлов этой директории содержатся следующие данные:

– `levelInit` — описание начального состояния каждого уровня;

– `initEarth` — инициализация земли;

– `initHero` — инициализация персонажа;

– `initGoal` — инициализация цели.

Опционально добавление следующих элементов:

– `initPlatforms` — список платформ;

– `initSpikes` — список шипов;

– `initClouds` — список платформ второго типа.

2.3 Директория Renderer

В файле `Render.hs` содержатся следующие элементы:

– `newWindow` — задает окно;

– `atmosphere` — задаёт фон;

– `drawGoal` — прорисовка цели;

– `drawPlayer` — прорисовка персонажа;

– `platToPic` — преобразовывает объект типа `Platform` в картинку;

– `platformGenerator` — итерируется по списку платформ, создавая массив картинок;

– `endGameScreen` — экран смерти;

– `goodGameScreen` — экран конца игры;

– `walkAnimationControlR` — анимация движения вправо;

– `walkAnimationControlUPR` — анимация прыжка вправо;

– `walkAnimationControlL` — анимация движения влево;

– `walkAnimationControlUPL` — анимация прыжка влево;

– `render` — отрисовка всего, преобразовывает отдельные картинки в одну;

– `scroll` — передвижение «камеры» вверх/вниз;

– `changeY` — вспомогательная функция, меняет координаты объектов при движении «камеры».

В файле `TextureLoader.hs` происходит загрузка текстур. В комментариях к коду в этом файле указано, какие текстуры для чего используются, поэтому я счел ненужным расписывать это повторно.

В файле `VisualConstants.hs` содержатся следующие константы:

– `windowName` — название окна;

– `initialWindowDimensions` — размеры окна;

– `initialWindowPosition` — расположение окна на экране;

– `backgroundColor` — цвет фона окна;

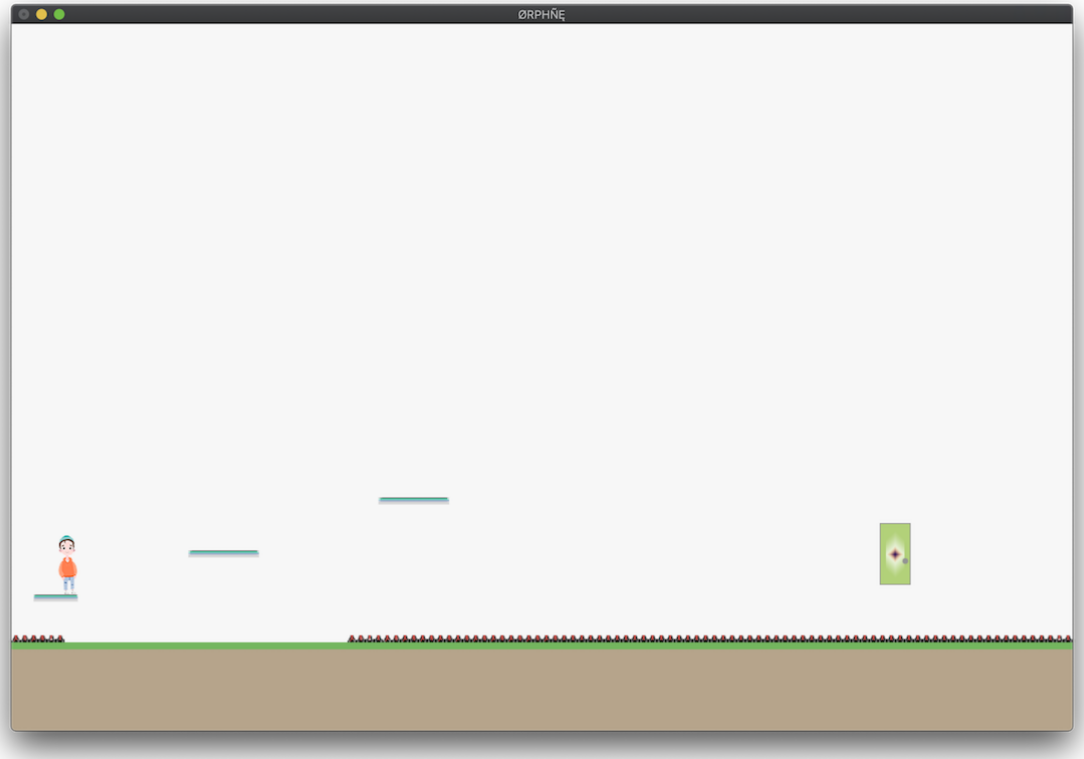
– `fps` — количество кадров в секунду.

3 Используемые библиотеки

При реализации проекта использовались следующие библиотеки:

- `gloss` — отрисовка объектов и обработка внешних событий;
- `yaml` — чтение конфигурационных файлов в формате YAML.

4 Работа с приложением



Проект можно собрать и запустить при помощи команд `stack build && stack exec orphne`.

После успешного запуска игрок попадает на первый уровень. Цель — дойти до зеленой двери. Управление персонажем осуществляется на кнопки **A**, **W** и **D**. Во время игрового процесса, для того чтобы начать уровень заново, можно нажать кнопку **R**. Также предусмотрен выбор желаемого уровня в любой момент игрового процесса на клавиши **1**, **2**, **3**, **4** и **5**.

Для завершения приложения необходимо нажать кнопку **esc**.