МОСКОВСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ИМЕНИ
М. В. ЛОМОНОСОВА
Факультет вычислительной математики и кибернетики
Кафедра алгоритмических языков

Отчёт о выполнении задания практикума

«PING-PONG»

Индивидуальный вариант

«Расширенный интерфейс»

Студент 325 группы
Делба М.Г.

Москва, 2022

1 Постановка задачи

PONG — это классическая аркадная игра 1972 года, созданная Atari. Игровой процесс состоит в том, что два игрока отбивают мячик ракетками, пытаясь заработать как можно больше очков. Это происходит, если противник не может отбить мячик, и он улетает за ракетку. Если он ударяется о периметр игрового поля или ракетку, то его траектория изменяется под действием отскакивания.

PING-PONG — это измененная версия классической PONG: вместо двух игроков в игре участвует всего один, и его задача - отбивать мячик в стенку напротив. Игровое поле также ограничено стенами слева и справа, а ракетка движется в горизонтальной плоскости путем нажатия клавиш. При каждом отскоке мячика от ракетки игрок получает очко, а скорость движения мячика увеличивается.

Основная задача - набрать как можно больше очков. Как только игроку не удается отбить мячик, он проигрывает.

Игровое поле состоит из трех стенок: одной горизонтальной напротив игрока и двух вертикальных по бокам, а также ракетки - подвижной платформы, которой и управляет игрок.

1.1 Базовые требования

- Отрисовка в каждом кадре игрового мира со всеми его объектами: задним фоном (поверхностью поля), тремя стенками, ракеткой и перемещающимся по полю мячом.
- Симуляция элементарной физики: проверка столкновений мячика со стенками или ракеткой и их обработка: изменение траектории и скорости мячика
- Предоставление игроку возможности управлять ракеткой
- Подсчет и отображение набираемых игроком очков
- Определение проигрыша игрока с возможностью возобновить игру (старт игры - клавиша 'P')

1.2 Индивидуальная часть «Расширенный интерфейс»

Расширенный интерфейс позволяет запоминать имя игрока при запуске игры и записывает его лучший текущий результат в таблицу рекордов. Таблица рекордов сохраняет топ-5 лучших игроков.

- Игрок может получить доступ к таблице (открыть/скрыть) при нажатии на клавишу 'S'
- Таблица рекордов не доступна во время игры: только до ее начала или на экране проигрыша.

2 Модули проекта

Проект содержит следуюшие модули:

- **Main.hs** - содержит главную функцию main, запускающую игру
- **Consts.hs** - константы и типы, используемые в коде
- **GameIO.hs** - логика игрового цикла, взаимодействие с пользователем
- **Physics.hs** - симуляция физики
- **Rendering.hs** - отрисовка объектов в окне
- **Table.hs** - работа с таблицей рекордов/базой данных

А также используется файл конфигурации `config.yaml`.

В модуле **Confg.hs** описаны следующие константы и типы:

КОНСТАНТЫ:
`windowWidth :: Int` - ширина окна
`windowHeight :: Int` - высота окна
`paddleMaxX :: Float` - максимальное отклонение ракетки по оси X
`wallWidth :: Float` - толщина стенок
`paddleWidth :: Float` - толщина ракетки
`paddleLength :: Float` - длина ракетки
`paddleSpeed :: Float` - скорость ракетки (перемещение по оси X за кадр)
`ballRad :: Float` - радиус мячика
`fieldColor :: Color` - цвет игрового поля
`ballColor :: Color` - цвет мячика
`paddleColor :: Color` - цвет ракетки
`wallColor :: Color` - цвет верхней стенки
`borderColor :: Color` - цвет боковых стенок
`paddleColor :: Color` - цвет ракетки
`initBallPos :: Pos` - начальная позиция мячика
`initDir :: Dir` - начальная траектория мячика
`initSpeed :: Float` - начальное значение скорости мячика
`initPaddlePos :: Pos` - начальная позиция ракетки
`initialState :: Connection -> Username -> ScoresList -> GameState` - начальное игровое состояние

ТИПЫ:
`type Pos = (Float, Float)` - позиция по X и Y
`type Dir = (Float, Float)` - вектор направления
`type ScoresList = [(String, Int)]` - список (Имя игрока, наибольший счет игрока)
`type UserName = String` - имя пользователя
`type ScoreValue = Int` - набранные игроком очки

`data Move = MoveRight | MoveLeft | NoMovement deriving Show` - направление движения ракетки по горизонтали

```
data GameState = GS {
ballPos :: Pos,
ballDir :: Dir,
ballSpeed :: Float,
paddlePos :: Pos,
paddleMove :: Move,
score :: Int,
scoreBoardShow :: Bool,
gameStarted :: Bool,
gameOver :: Bool,
connection :: Connection,
scoresList :: ScoresList,
userName :: UserName
}
```

- структура - состояние игры

В модуле **GameIO.hs** описаны следующие функции:

`startGame :: IO` - запуск игры

`updateApp :: Float -> GameState -> IO GameState` - обновление состояния игры

`updateGS :: Float -> GameState -> GameState` - вспомогательная функция, вызываемая в `updateApp`

`checkArgs :: [String] -> String` - проверка аргументов командной строки на наличие имени игрока

`handleEventStart :: Event -> GameState -> IO GameState` - функция обработки событий - нажатий клавиш, меняющая состояние игры

`handleEvent :: Event -> GameState -> GameState` - вспомогательная функция, вызываемая в `handleEventStart`

В модуле **Physics.hs** описаны следующие функции:

`physicsBall :: Float -> GameState -> GameState` - функция перемещения мячика по игровому полю

`paddleControl :: GameState -> GameState` - функция управления ракеткой игроком

`physicsCollision :: GameState -> GameState` - функция обработки физики столкновений мячика с каким-либо объектом

`checkPaddleCollision :: Pos -> Pos -> Bool` - вспомогательная функция - проверка столкновения мячика с ракеткой, вызываемая в `physicsCollision`

`checkWallCollision :: Pos -> Bool` - вспомогательная функция - проверка столкновения мячика с верхней стенкой, вызываемая в `physicsCollision`

`checkBorderCollision :: Pos -> Bool` - вспомогательная функция - проверка столкновения мячика с боковыми стенками, вызываемая в `physicsCollision`

`ballMissed :: GameState -> GameState` - функция, обрабатывающая вылет мячика за пределы поля

В модуле **Rendering.hs** описаны следующие функции:

`render :: GameState -> IO Picture` - отрисовка всех объектов игрового мира на экране

`drawPaddle :: Pos -> Picture` - отрисовка ракетки

`drawBorder :: Float -> Picture` - отрисовка боковой стенки

`drawScoreBoard ::Int -> Int -> GameState -> [Picture]` - отрисовка таблицы рекордов

В модуле **Table.hs** описаны следующие функции:

`createID :: UserName -> IO ()` - создание в таблице users БД новой записи (ID игрока, имя игрока)

`writeScoreToTable :: ScoreValue -> Connection -> IO ()` - создание в таблице scores БД новой записи (ID игрока, счет игрока)

`getTop5 :: Connection -> IO ScoresList` - получение из БД по SQL-запросу списка 5 лучших игроков

3 Используемые библиотеки

При написании исходного кода игры использовались следующие библиотеки:

- **gloss** - простая графическая библиотека, обработка событий
- **sqlite-simple** - работа с базой данных Sqlite3

4 Сценарии работы с приложением

При запуске приложения игрок может выбрать себе имя, используя аргументы командной строки: $ stack run <name>. Если имени нет, или формат аргументов другой, будет автоматически выбрано имя Mike.

Игрок попадает на экран начала игры. Она запускается при нажатии клавиши 'P'. Также игрок может посмотреть таблицу рекордов, нажав клавишу 'S', но только если игра еще не запущена.

Во время игры видны игровое поле, стенки, ракетка и мячик. После начала игры он сразу летит в сторону ракетки. Сверху присутствует текущий счет игрока - количество набранных очков.

Игрок может двигать ракеткой из стороны в сторону, пытаясь отбивать мячик. Основная цель - набрать как можно больше очков, которые игрок получает при каждом успешном отбивании. Однако, при каждом ударе мячик немного ускоряется, и отбить его с каждым разом становится все сложнее. Если игроку это не удается, игра завершается, и выводится экран проигрыша с надписью LOST. Можно посмотреть таблицу рекордов или начать игру заново.