МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

федеральное государственное бюджетное образовательное учреждение
высшего образования
«ИРКУТСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»
(ФГБОУ ВО «ИГУ»)

Институт математики и информационных технологий

Кафедра информационных технологий

ОТЧЕТ

о курсовой работе

«Разработка приложения на языке функционального программирования»

Студента 3 курса группы 02321-ДБ
направления 01.03.02 «Прикладная математика и информатика»
Ковалева Степана Алексеевича

Руководитель:
Черкашин Евгений Александрович,
преподаватель
Оценка
_____

Иркутск – 2022

Оглавление

Введение: 3
1. Теоретическая часть: 3
2. Практическая часть: 4
2.1 Архитектура 4
2.2 Реализация 4
2.3 Интерфейс 4
Заключение: 10
Список литературы: 10

Введение:

Язык программирования Haskell, названный в честь Хаскелля Брукса Карри, чьи работы в области математической логики служат основой функциональных языков - это стандартизированный, чистый функциональный язык программирования общего назначения, довольно отличающийся от остальных языков программирования, также часто называемый “ленивым” языком программирования. Haskell основан на лямбда-исчислении, поэтому лямбда используется как логотип. Язык специально разработан, чтобы оперировать широким спектром функций, от численных до символических. Поэтому Haskell имеет ясный синтаксис и богатое разнообразие встроенных типов данных, включая длинные целые и рациональные числа, а также обычные целые числа, числа с плавающей запятой и булевые типы.

1. Теоретическая часть:

Haskell — это язык функционального программирования, который был специально разработан для обработки символьных вычислений и обработки списков. Функциональное программирование основано на математических функциях. Помимо Haskell, некоторые из других популярных языков, которые следуют парадигме функционального программирования, включают: Lisp, Python, Erlang, Racket, F #, Clojure и т.д.

В традиционном программировании инструкции принимаются как набор объявлений в определенном синтаксисе или формате, но в случае функционального программирования все вычисления рассматриваются как комбинация отдельных математических функций.

Тема:

Разработать приложение на языке функционального программирования: солнце на canvas. В зависимости от времени нужно нарисовать солнце над горизонтом и траву. Пользователь может двигать солнце стрелками.

Для изображения реализованных в программном коде на языке Haskell фигур будет использоваться библиотека Gloss, которая даст удобный

интерфейс для использования OpenGl для рисования векторной графике на canvas.

2. Практическая часть:

2.1 Архитектура

У нас есть type World, в нем лежат 2 числа с плавающей запятой (Float), ими мы полностью можем описать мир, первое значение это значение поворота от изначального положения солнца, а второе значение - это описание, куда и как быстро будет поворачиваться наши солнце и луна. Программу можно разбить на 3 основных логических модуля:

drawingFunc - рисуются фигуры

inputHandler - обрабатываются события ввода каких-либо символов

updateFunc - наш мир обновляется, точнее обновляется значение поворота

2.2 Реализация

Все фигуры, включая фон и саму землю, это круги, одни статичны (фон, земля), другие вращаются вокруг земли, создавая впечатления смены дня и ночи. Специально делаем вращение не вокруг центра земли, и тогда положение луны и солнца будет в зените, будет выше чем когда будет уходить или выходить. Чтобы у нас как можно больше времени на экране были фигуры, мы сделали 2 солнца и 2 луны. Они чередуются 0 градусов солнце, 90 луна, 180 солнце, 270 луна. Луна - это два круга один - это тело луны, другой круг поменьше и у него цвет фона. Таким образом, мы и получаем месяц. Изначально все фигуры движутся по часовой стрелке, но мы можем это изменить, если нажмем на стрелочку влево, и фигуры начнут двигаться в обратном направлении. Чтобы вернуть обратно, нужно нажать стрелочку вправо, если несколько раз нажать, то можно ускорить вращение. Максимально можно ускорить вращение 2 раза. На SPACE мир перестает изменять свои значения, и фигуры застывают.

2.3 Интерфейс

Разберем **drawingFunc** на примере фона. Мы рисуем круг в центре:

```
// ставим фигуру в (0,0)

// видим что фон это большой круг, но цветом backgroundColor times:

 background

        = Pictures

            [Translate 0 0

            $ Color (backgroundColor times) (circleSolid 800)]
```

times - это насколько четвертей окружности повернулись наши фигуры от изначального положения. Нужен для того, чтобы понимать от какого цвета к какому идет градиент.

```
backgroundColor times = getColorGradient color1 color2 times

  where

    numberHalvesPi = floor (abs times) :: Integer // берем целую часть поворот pi / 2

    color1 = if even numberHalvesPi // если оно четно

      then (42, 231, 245) // значит сейчас у нас солнце заходит и цвет идет от более теплых

      else (20, 53, 97)

    color2 = if even numberHalvesPi

      then (20, 53, 97) // к более холодным

      else (42, 231, 245) // и наоборот


lerp :: Float -> Float -> Float -> Float

lerp x0 x1 p = x0 + (x1 - x0) * p // линейная интерполяция


type GradientColor = (Float, Float, Float)

getColorGradient :: GradientColor -> GradientColor-> Float -> Color

// вычисляем цвет
```

```
        //получив интерполяцию от каждого канала цвета RGB
        getColorGradient (rhsR, rhsG, rhsB) (lhsR, lhsG, lhsB) progress =
makeColor curR curG curB 1
          where
            fract = abs progress - fromIntegral (floor (abs progress)) :: Float
            curR = lerp rhsR lhsR fract / 256
            curG = lerp rhsG lhsG fract / 256
            curB = lerp rhsB lhsB fract / 256
```

Следующий модуль реализует пользовательский ввод:

```
  inputHandler :: Event -> World -> World
  inputHandler (EventKey (SpecialKey KeyRight) Down _ _) (x, y) = (x, dy)
    where
      dy = if y > 0 then min (y + 0.5) 2 else 1
  inputHandler (EventKey (SpecialKey KeyLeft) Down _ _) (x, y) = (x, dy)
    where
      dy = if y < 0 then max (y - 0.5) (-2) else -1
  inputHandler (EventKey (SpecialKey KeySpace) Down _ _) (x, y) = (x, 0)
  -- disable other input handler
  inputHandler _ w = w
```

Первые 3 обработчика мы выставляем при нажатии стрелок и SPACE как быстро и в каком направлении будут вращаться фигуры, если y < 0 значит вращение против часовой, y > 0 по часовой. Когда y = 0, то мир не изменяется. Последний обработчик заставляет все остальные события ввода игнорироваться.

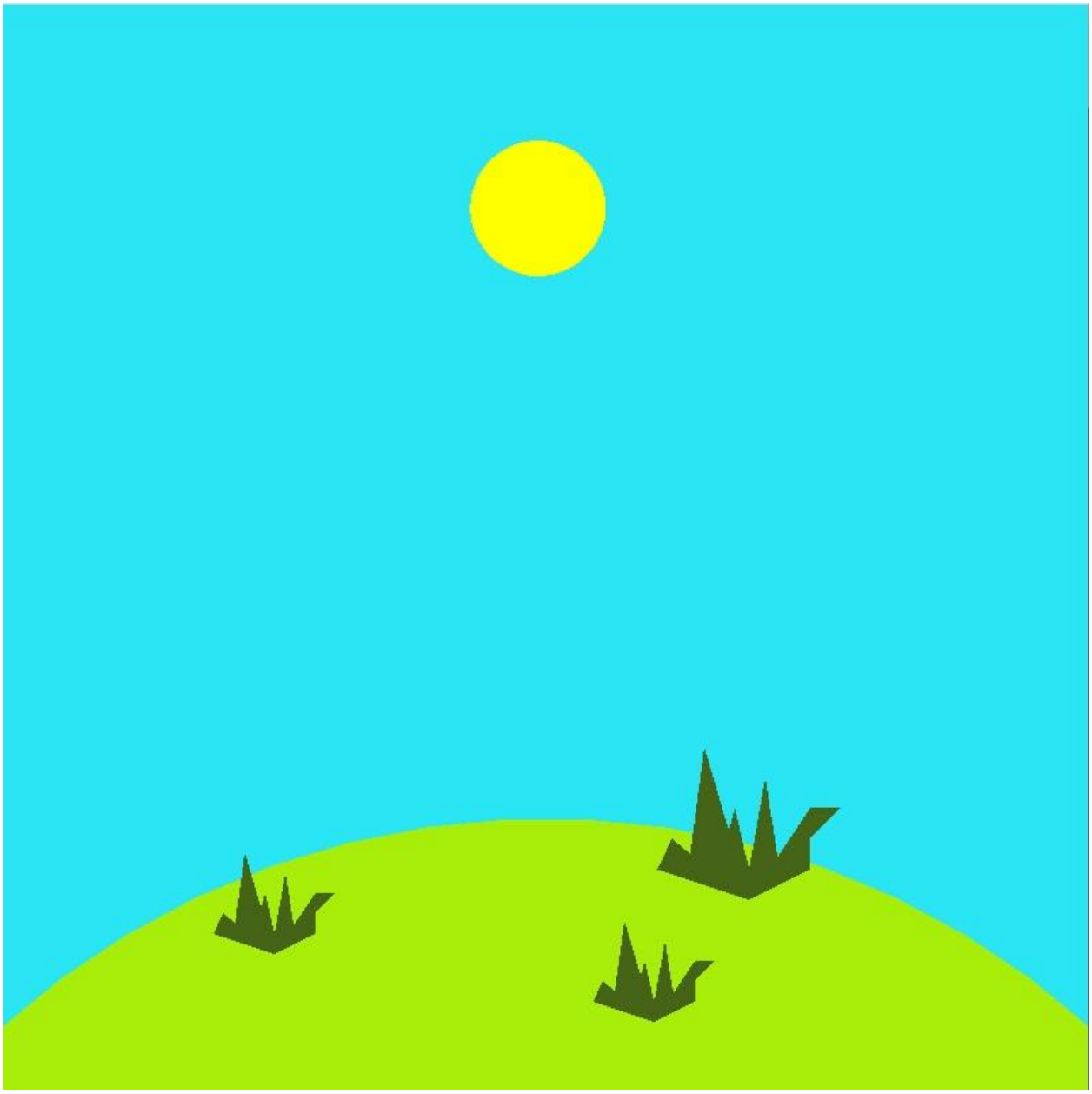
Последний модуль обновление мира:

```
  updateFunc :: Float -> World -> World
  updateFunc _ (x, y) = (\x y -> (x + y * 0.25, y)) x y
```

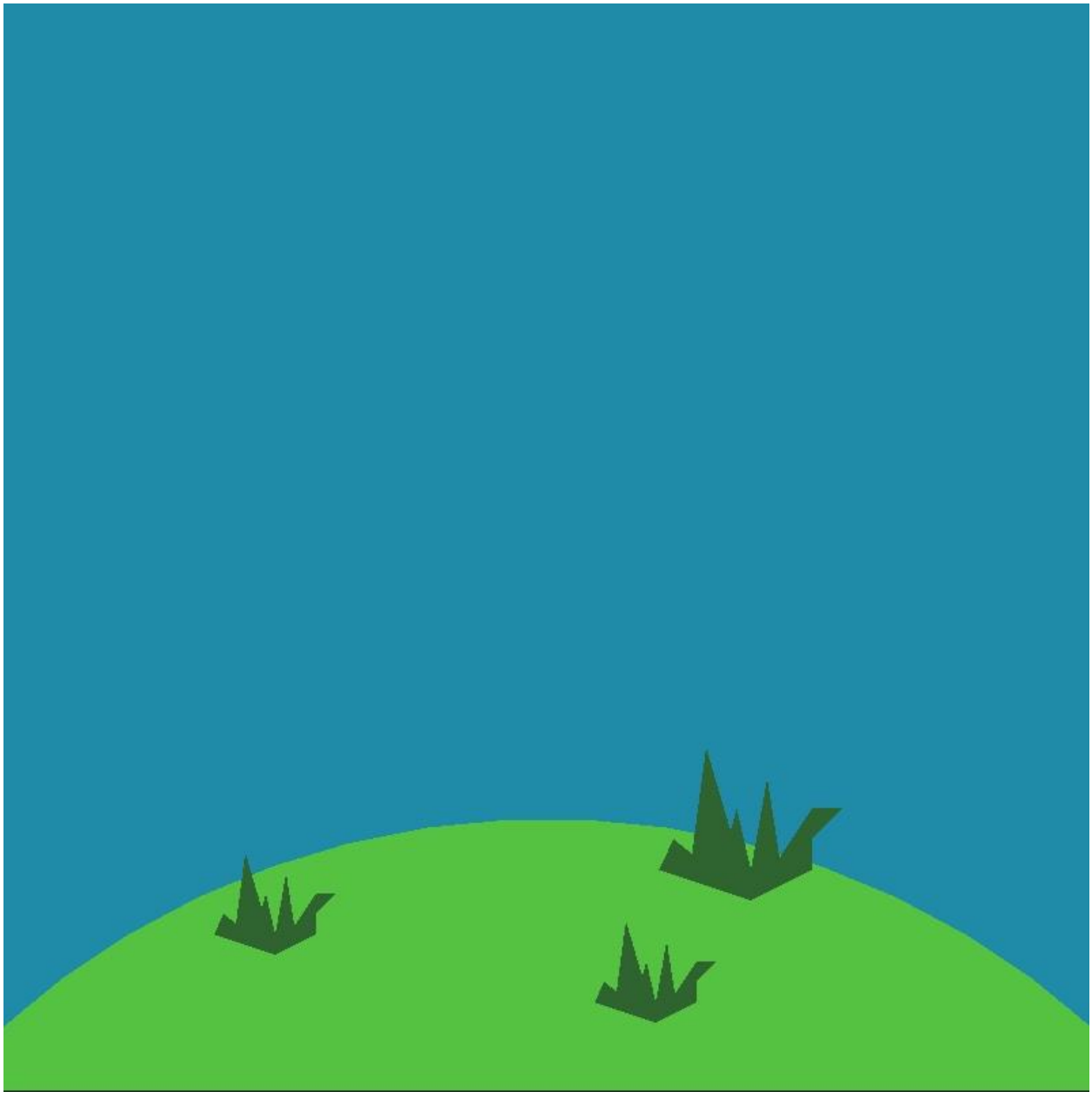
увеличиваем поворот на характеристику поворота умноженную на 0.25 (на 1 выходит слишком быстрое вращение).

2.4 Демонстрация

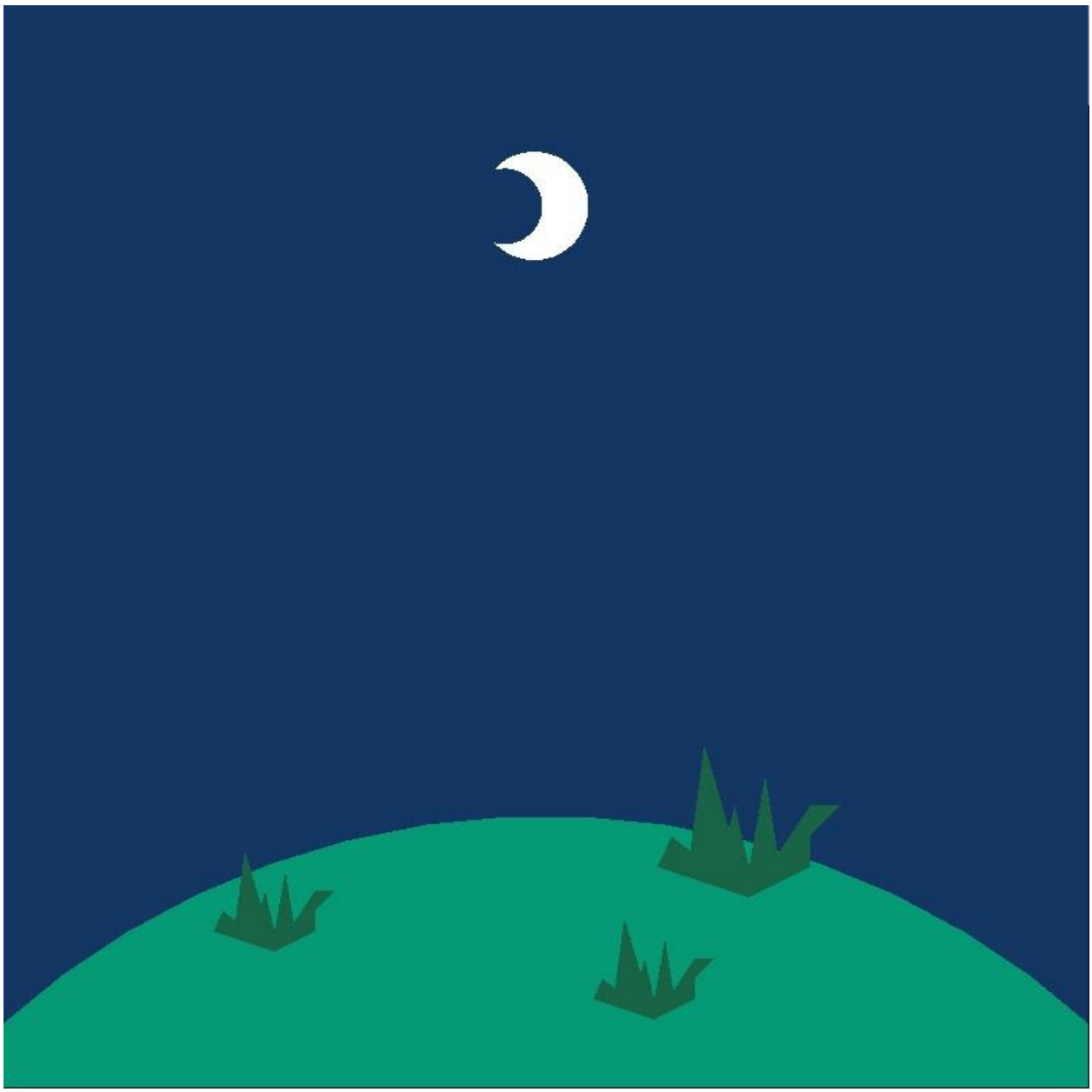
Солнце в зените



Солнце зашло



Луна в зените



Заключение:

В результате проделанной работы было получено приложение, солнце на canvas. Для решения поставленной задачи был разработан и реализован алгоритм, который в зависимости от времени рисует солнце над горизонтом. Для реализации приложения была использована среда разработки языка Haskell.

Список литературы:

1. Миран Липовача. Изучай Haskell во имя добра! / Пер. с англ. Леушина Д., Синицына А., Арсанукаева Я.– М.: ДМК Пресс, 2012. – 490 с.: ил. ISBN 978-5-94074-749-9
2. Haskell URL: https://zvon.org/other/haskell/Outputglobal/index.html
3. Directory listing for gloss-1.13.2.1 documentation URL: https://hackage.haskell.org/package/gloss-1.13.2.1/docs/