Московский авиационный институт (национальный исследовательский университет)

Факультет информационных технологий и прикладной математики

Кафедра вычислительной математики и программирования

Лабораторная работа №1 по курсу «Компьютерная графика»
Тема: «Построение изображений 2D- кривых»

Студент: И. Д. Недосеков
Преподаватель: Чернышов Л. Н.
Группа: М8О-306Б-19

Дата:
Оценка:
Подпись:

Москва, 2021

Содержание

1 Постановка задачи

Лабораторная работа №1

Написать и отладить программу, строящую изображение заданной замечательной кривой

Вариант:
17. $x = aj - b * \sin j, y = a - b * \cos j, a < b, A \leq j \leq B$

2 Описание программы

Программа написана на Golang и Gtk для пользовательского интерфейса. Разметка интерфейса хранится в main.glade, все расчеты точек графика и отрисовки в render.go
Инструкция по установке:

- установить библиотеку GTK
- установить среду разработки Golang
- установить библиотеки для Golang (выполнить *go get -v {ссылка на репозиторий в github}*)
  - goGtk github.com/gotk3/gotk3/gtk
  - goCairo github.com/gotk3/gotk3/cairo
- скопировать файлы main.go, render.go, parsing.go, main.glade
- перейти в дирректорию проекта и запустить через команда *go run .*

3 Листинг программы

```
// main.go
package main

import (
"log"
"math"

"github.com/gotk3/gotk3/gtk"
)

func f(A, B, a, b, phi float64) (x, y float64) {
  x = a*phi - b*math.Sin(phi)
  y = a - b*math.Cos(phi)
  return
}

type globals struct {
  A, B, a, b             float64
  w                      *gtk.Window
  f                      func(A, B, a, b, phi float64) (x, y float64)
  count, start_i, segLen int
  calculated_points      []struct{ x, y float64 }
}

var vars globals

func main() {
  vars.f = f
  vars.count = 102
  vars.start_i = -50
  vars.segLen = 2

  // Инициализируем GTK.
  gtk.Init(nil)

  // Создаём билдер
  builder, err := gtk.BuilderNew()
  if err != nil {
    log.Fatal("Ошибка:", err)
  }

```

```
42      // Загружаем в билдер окно из файла Glade
43      err = builder.AddFromFile("main.glade")
44      if err != nil {
45        log.Fatal("Ошибка:", err)
46      }
47
48      // Получаем объект главного окна по ID
49      obj, err := builder.GetObject("window_main")
50      if err != nil {
51        log.Fatal("Ошибка:", err)
52      }
53
54      // Преобразуем из объекта именно окно типа gtk.Window
55      // и соединяем с сигналом "destroy" чтобы можно было закрыть
56      // приложение при закрытии окна
57      win := obj.(*gtk.Window)
58      win.Connect("destroy", func() {
59        gtk.MainQuit()
60      })
61      vars.w = win
62
63      obj, err = builder.GetObject("A_input")
64      if err != nil {
65        log.Fatal(err)
66      }
67      input := obj.(*gtk.Entry)
68      parse_A(input)
69      input.Connect("changed", parse_A)
70
71      obj, err = builder.GetObject("B_input")
72      if err != nil {
73        log.Fatal(err)
74      }
75      input = obj.(*gtk.Entry)
76      parse_B(input)
77      input.Connect("changed", parse_B)
78
79      obj, err = builder.GetObject("a_input")
80      if err != nil {
81        log.Fatal(err)
82      }
83      input = obj.(*gtk.Entry)
84      parse_a(input)
85      input.Connect("changed", parse_a)
```

```
86
87      obj, err = builder.GetObject("b_input")
88      if err != nil {
89        log.Fatal(err)
90      }
91      input = obj.(*gtk.Entry)
92      parse_b(input)
93      input.Connect("changed", parse_b)
94
95      obj, err = builder.GetObject("canvas")
96      if err != nil {
97        log.Fatal(err)
98      }
99      canvas := obj.(*gtk.DrawingArea)
100     canvas.Connect("draw", drawWhiteCanvas)
101     canvas.Connect("draw", drawAxis)
102     canvas.Connect("draw", renderGraphic)
103
104     // Отображаем все виджеты в окне
105     win.ShowAll()
106
107     // Выполняем главный цикл GTK (для отрисовки). Он остановится когда
108     // выполнится gtk.MainQuit()
109     gtk.Main()
110   }
111
```

```
1   // render.go
2   package main
3
4   import (
5   "log"
6   "math"
7   "strconv"
8
9   "github.com/gotk3/gotk3/cairo"
10  "github.com/gotk3/gotk3/gtk"
11  )
12
13  func drawWhiteCanvas(da *gtk.DrawingArea, cr *cairo.Context) {
14    cr.SetSourceRGB(255, 255, 255)
15
```

```
16    w, h := da.GetAllocatedWidth(), da.GetAllocatedHeight()
17
18    cr.Rectangle(0, 0, float64(w), float64(h))
19    cr.Fill()
20    log.Printf("Draw Canvas w=%d, h=%d", w, h)
21  }
22
23  func calculateArrowPoints(startX, startY, endX, endY, legth float64) (x1, y1, x2,
    ↪  y2 float64) {
24    const wingsAngel = math.Pi / 10
25    angle := math.Atan2(endY-startY, endX-startX) + math.Pi
26    x1 = endX + legth*math.Cos(angle-wingsAngel)
27    y1 = endY + legth*math.Sin(angle-wingsAngel)
28    x2 = endX + legth*math.Cos(angle+wingsAngel)
29    y2 = endY + legth*math.Sin(angle+wingsAngel)
30    log.Printf("Calculated x1=%f, y1=%f, x2=%f, y2=%f, ang1=%f, ang2=%f", x1, y1, x2,
      ↪  y2, math.Cos(angle-wingsAngel), math.Cos(angle+wingsAngel))
31    return
32  }
33
34  func drawArrows(cr *cairo.Context, startX, startY, endX, endY float64) {
35    cr.MoveTo(startX, startY)
36    cr.LineTo(endX, endY)
37
38    x1, y1, x2, y2 := calculateArrowPoints(startX, startY, endX, endY, 10)
39    cr.MoveTo(endX, endY)
40    cr.LineTo(x1, y1)
41    cr.MoveTo(endX, endY)
42    cr.LineTo(x2, y2)
43    cr.Stroke()
44  }
45
46  func drawNumirate(cr *cairo.Context, startX, startY, endX, endY float64) {
47
48    cr.SetSourceRGB(0, 0, 0)
49
50    dx, dy := (endX-startX)/float64(vars.count), (endY-startY)/float64(vars.count)
51    for posX, posY, i := startX+dx, startY+dy, vars.start_i; i < vars.count/2; posX,
      ↪  posY, i = posX+dx, posY+dy, i+1 {
52      cr.MoveTo(posX-float64(vars.segLen), posY)
53      cr.LineTo(posX+float64(vars.segLen), posY)
54      cr.MoveTo(posX, posY-float64(vars.segLen))
55      cr.LineTo(posX, posY+float64(vars.segLen))
56      cr.MoveTo(posX-7*float64(vars.segLen), posY+7*float64(vars.segLen))
```

```
57
58          if i%10 == 0 {
59          tmp_str_i := strconv.Itoa(i)
60          cr.ShowText(tmp_str_i)
61        }
62      }
63      log.Println("Axis enumerated")
64      cr.Stroke()
65
66      }
67
68      func drawAxis(da *gtk.DrawingArea, cr *cairo.Context) {
69      cr.SetSourceRGB(0, 0, 0)
70      cr.SetLineWidth(1)
71      w, h := float64(da.GetAllocatedWidth()), float64(da.GetAllocatedHeight())
72
73      horizontalPadding := w * 0.02
74      startX, startY := horizontalPadding, h/2
75      endX, endY := w-horizontalPadding, h/2
76
77      drawArrows(cr, startX, startY, endX, endY)
78      drawNumirate(cr, startX, startY, endX, endY)
79
80      verticalPadding := h * 0.05
81      startX, startY = w/2, h-verticalPadding
82      endX, endY = w/2, verticalPadding
83
84      drawArrows(cr, startX, startY, endX, endY)
85      drawNumirate(cr, startX, startY, endX, endY)
86
87      log.Printf("Axis rendered: w=%f h=%f hpad=%f\n", w, h, horizontalPadding)
88      }
89
90      func renderGraphic(da *gtk.DrawingArea, cr *cairo.Context) {
91      cr.SetSourceRGB(255, 0, 0)
92      cr.SetLineWidth(1)
93      w, h := float64(da.GetAllocatedWidth()), float64(da.GetAllocatedHeight())
94
95      horizontalPadding := w * 0.02
96      verticalPadding := h * 0.05
97      startX, endX := horizontalPadding, w-horizontalPadding
98      startY, endY := h-verticalPadding, horizontalPadding
99      dx, dy := (endX-startX)/float64(vars.count), (endY-startY)/float64(vars.count)
100
```

```
  var x, y float64
  cx, cy := float64(da.GetAllocatedWidth()/2), float64(da.GetAllocatedHeight()/2)
  cr.MoveTo(cx+vars.calculated_points[0].x*dx, cy+vars.calculated_points[0].y*dy)
  for i := 0; i < 100; i++ {
    x, y = vars.calculated_points[i].x, vars.calculated_points[i].y
    x, y = cx+x*dx, cy+y*dy
    cr.LineTo(x, y)
  }
  cr.Stroke()

  }

  func recalculate_points() {
  if vars.calculated_points == nil {
    vars.calculated_points = make([]struct{ x, y float64 }, 100)
  }
  var phi_value float64
  dist := vars.B - vars.A
  for i := 0; i < 100; i++ {
    phi_value = vars.A + float64(i)/100*dist
    vars.calculated_points[i].x, vars.calculated_points[i].y = vars.f(vars.A, vars.B,
    ↪  vars.a, vars.b, phi_value)
  }
  log.Printf("points recalculated")
  }
```

```
// parsing.go
package main

import (
"log"
"strconv"

"github.com/gotk3/gotk3/gtk"
)

func parse_a(entry *gtk.Entry) {
  str, err := entry.GetText()
  if err != nil {
    log.Fatal("Cant get text grom entry", err)
  }
  tmp, err := strconv.ParseFloat(str, 64)
```

```
    if err != nil {
      log.Print("Cant parse float", err)
      return
    }
    if tmp >= vars.b {
      log.Printf("Logic error a (get %f) must be less b=%f", tmp, vars.b)
      return
    }
    recalculate_points()
    vars.a = tmp
  }

  func parse_A(entry *gtk.Entry) {
    str, err := entry.GetText()
    if err != nil {
      log.Fatal("Cant get text grom entry", err)
    }
    tmp, err := strconv.ParseFloat(str, 64)
    if err != nil {
      log.Print("Cant parse float", err)
      return
    }
    if tmp > vars.B {
      log.Printf("Logic error A (get %f) must be less or equal B=%f", tmp, vars.B)
      return
    }
    vars.A = tmp
    recalculate_points()
    log.Printf("A %f", vars.A)
  }

  func parse_b(entry *gtk.Entry) {
    str, err := entry.GetText()
    if err != nil {
      log.Fatal("Cant get text grom entry", err)
    }
    tmp, err := strconv.ParseFloat(str, 64)
    if err != nil {
      log.Print("Cant parse float", err)
      return
    }
    if vars.a > tmp {
      log.Printf("Logic error a=%f must be less or equal b (get %f)", vars.a, tmp)
      return
```

```
61      }
62      vars.b = tmp
63      recalculate_points()
64      log.Printf("b %f", vars.b)
65    }
66
67    func parse_B(entry *gtk.Entry) {
68      str, err := entry.GetText()
69      if err != nil {
70        log.Fatal("Cant get text grom entry", err)
71      }
72      tmp, err := strconv.ParseFloat(str, 64)
73      if err != nil {
74        log.Print("Cant parse float", err)
75        return
76      }
77      if vars.A > tmp {
78        log.Printf("Logic error A=%f must be less or equal B (get %f)", vars.A, tmp)
79        return
80      }
81      vars.B = tmp
82      recalculate_points()
83    }
```

4 Тесты

1 Наборы тестов

1. $A = 0; B = 6.28318; a = 1.5; b = 22$
2. $A = -2; B = 6.28318; a = 17; b = 22$
3. $A = 0; B = 6.28318; a = 17; b = 40$

2 Визуализация тестов

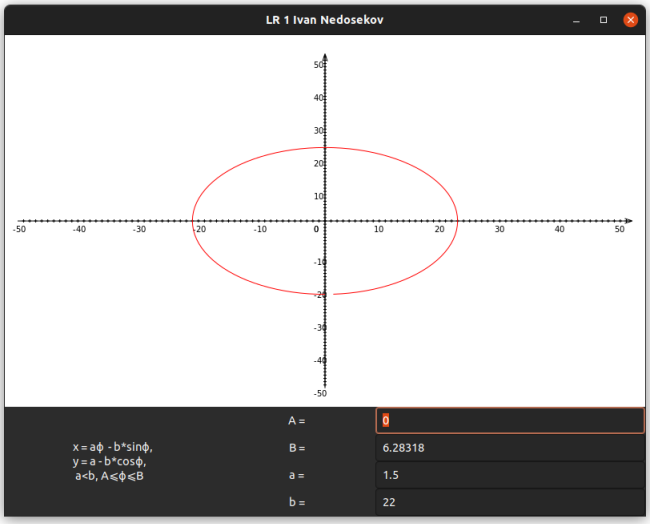

Рис. 1: 1ый тестовый набор

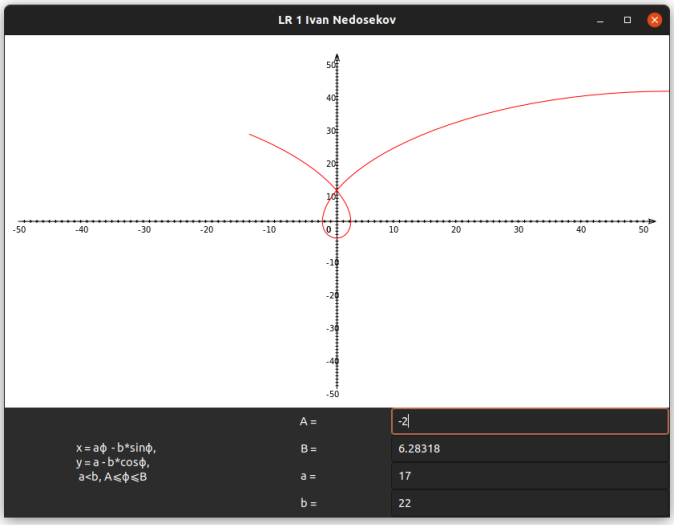

Рис. 2: 2ой тестовый набор

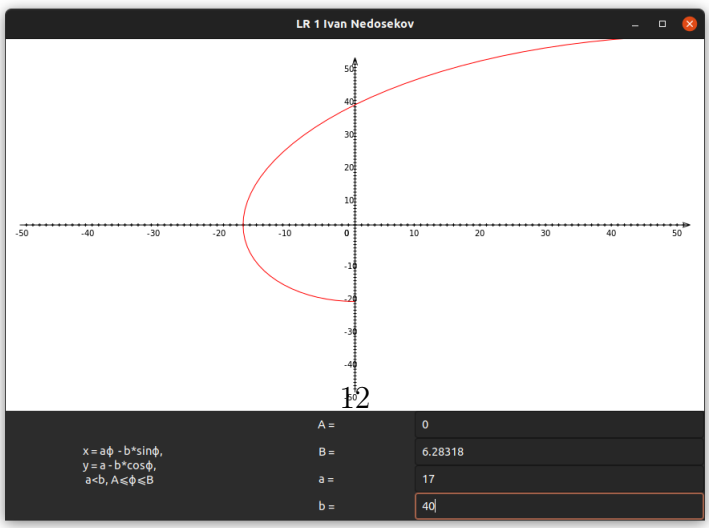

Рис. 3: 3ий тестовый набор

5 Выводы

Выполнив данную лабораторную работу я познакомился с фреймфорком для отрисовывания GUI GTK, а также изучил основы отрисовки и масштабирования графика.

Список литературы

[1] *GTK — официальная документация.*
URL: www.gtk.org

[2] *Golang — официальная документация.*
URL: golang.org