Московский авиационный институт (национальный исследовательский университет)

Факультет информационных технологий и прикладной математики

Кафедра вычислительной математики и программирования

Лабораторная работа №7 по курсу «Компьютерная графика»
Тема: Построение плоских полиномиальных кривых.

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

Дата:
Оценка:
Подпись:

Москва, 2021

Содержание

1 Постановка задачи

Лабораторная работа №7

Написать программу, строящую полиномиальную кривую по заданным точкам. Обеспечить возможность изменения позиции точек и, при необходимости, значений касательных векторов и натяжения.

Вариант:

10. B-сплайн. n = 6, k = 3. Узловой вектор равномерный.

2 Описание программы

Программа написана на Golang[2] и OpenGL[1] все расчеты точек графика и отрисовки в vertex_calculator.go. В compile.go операции по работе с шейдерами.
Инструкция по установке:

- установить среду разработки Golang
- установить библиотеки для Golang (командой такого вида) `go get -v {репозиторий github}`
  - github.com/go-gl/gl/v3.3-core/gl
  - github.com/go-gl/glfw/v3.3/glfw
  - github.com/go-gl/mathgl/mgl32
  - github.com/inkyblackness/imgui-go/v4
- скопировать файлы main.go, vertex_calculator.go, compile.go
- перейти в директорию проекта и запустить через команду `go run .`

3 Листинг программы

main.go

```
package main

import (
"log"

"fmt"

g "github.com/AllenDang/giu"
math "github.com/chewxy/math32"
"github.com/go-gl/mathgl/mgl32"
)

var (
count_dots  = 100
N           []func(float32) float32
r_vec       func(z float32) mgl32.Vec2
l, r        float32
nodes       []mgl32.Vec2
linedata    []float64
n_lines     [][]float32
kek_lines   [][]float64
n_plots     []g.PlotWidget
dot_sliders []g.Widget
)

func FindMinMax(t []float32) (min_el, max_el float32) {
  if len(t) == 0 {
    log.Default().Print("Error zerro leght FindMinMax")
  }
  for _, el := range t {
    min_el = math.Min(el, min_el)
    max_el = math.Max(el, max_el)
  }
  return
}

func divided_difference(f func(float32) func(float32) float32, t ...float32)
↪  func(float32) float32 {

  if len(t) == 2 {

```

```
    res := func(z float32) float32 {
      if t[1]-t[0] == 0 {
        return 0
      }
      return (f(t[1])(z) - f(t[0])(z)) / (t[1] - t[0])

    }
    return res
  } else if len(t) < 2 {
    log.Fatal("len < 2")
  }
  n := len(t) - 1
  d1 := divided_difference(f, t[1:n]...)
  d2 := divided_difference(f, t[0:n-1]...)

  return func(z float32) float32 {
    return (d1(z) - d2(z)) / (t[n] - t[0])
  }
}

func truncated_power_function(n int, t float32) func(float32) float32 {
  n_float := float32(n)
  return func(z float32) float32 {
    if z <= t {
      return 0.
    }

    return math.Pow((z - t), n_float)
  }

}

func truncated_power_function_fixed_n(n int) func(float32) func(float32) float32 {

  return func(t float32) func(float32) float32 {
    return truncated_power_function(n, t)
  }
}

func calculate_N(n, k int) (float32, float32, []func(float32) float32) {

  t_max := n - k + 2
  var t []float32
  N := make([]func(float32) float32, n+1)
```

```
85
86    for i := 1; i < k; i++ {
87      t = append(t, 0)
88    }
89
90    for i := 0; i <= t_max; i++ {
91      t = append(t, float32(i))
92    }
93
94    for i := 1; i < k; i++ {
95      t = append(t, float32(t_max))
96    }
97    sigma := truncated_power_function_fixed_n(k - 1)
98    norm := float32(t_max - 0)
99    for i := range N {
100     tmp_i := i
101     d := divided_difference(sigma, t[tmp_i:tmp_i+k+1]...)
102     N[i] = func(z float32) float32 {
103       return norm * d(z)
104     }
105   }
106
107   return 0, float32(t_max), N
108
109 }
110
111 func format_r(node []mgl32.Vec2) func(z float32) mgl32.Vec2 {
112
113   res_f := func(z float32) mgl32.Vec2 {
114     var res mgl32.Vec2
115     for i, ri := range node {
116       j := i
117       tmp_ri := ri
118       res = res.Add(tmp_ri.Mul(N[j](z)))
119     }
120     return res
121   }
122
123   return res_f
124
125 }
126
127 func calculate_line(l, r float32, f func(float32) mgl32.Vec2) []mgl32.Vec2 {
128   res := make([]mgl32.Vec2, count_dots)
```

```
  for i := range res {
    t := (r - l) * float32(i) / float32(count_dots)
    res[i] = f(t)
  }
  return res
}

func kekw(a []mgl32.Vec2) []float64 {
  res := make([]float64, len(a))
  for i, el := range a {
    res[i] = float64(el.Y())
  }
  return res
}

func calculateNLines() [][]float32 {
  res := make([][]float32, len(N))
  for j := range N {
    res[j] = make([]float32, count_dots)
  }
  for i := range res[0] {
    t := (r - l) * float32(i) / float32(count_dots)
    for j, f := range N {
      res[j][i] = f(t)

    }
  }
  return res
}

func init() {
  nodes = []mgl32.Vec2{
    {0, 1},
    {0, -1},
    {0, 1},
    {0, -1},
    {0, 1},
    {0, -1},
    {0, 1},
  }
  l, r, N = calculate_N(6, 3)
  r_vec = format_r(nodes)
  linedata = kekw(calculate_line(l, r, r_vec))
  n_lines = calculateNLines()
```

```
}

func loop() {
  dot_sliders = make([]g.Widget, len(nodes))
  for i := range nodes {
    dot_sliders[i] = g.SliderFloat(
    fmt.Sprintf("node %d", i),
    &nodes[i][1],
    -10,
    10,
    )

  }
  linedata = kekw(calculate_line(l, r, r_vec))
  dotsX := make([]float64, len(nodes))
  dotsy := make([]float64, len(nodes))

  for i, n := range nodes {
    dotsX[i] = float64(n.X())
    dotsy[i] = float64(n.Y())
  }
  g.SingleWindow().Layout(
  g.Plot("-spline  = 6, K = 3,  = (z - )).AxisLimits(-0.5, 5.5, -10, 10,
  ↪  g.ConditionOnce).Plots(
  g.PlotLine("spline", linedata).XScale(float64(r-l)/100),
  g.PlotScatterXY("Nodes", dotsX, dotsy),
  ),

  g.Plot(" basis").AxisLimits(-0.5, 5.5, -10, 10, g.ConditionOnce).Plots(n_plots...),
  dot_sliders[0],
  dot_sliders[1],
  dot_sliders[2],
  dot_sliders[3],
  dot_sliders[4],
  dot_sliders[5],
  dot_sliders[6],
  )
}

func main() {
  kek_lines = make([][]float64, len(N))
  for i := range kek_lines {
    kek_lines[i] = make([]float64, count_dots)
  }
```

```
  for i := range nodes {
    nodes[i][0] = float32(i) / float32(len(nodes)) * (r - l)
  }
  for i, line := range n_lines {
    for j, el := range line {
      kek_lines[i][j] = float64(el)
    }
    n_plots = append(n_plots, g.PlotLine(fmt.Sprintf("N[%d]", i),
↪   kek_lines[i]).XScale(float64(r-l)/100))
  }

  wnd := g.NewMasterWindow("Lab 7 Nedosekov ", 1000, 800, 0)
  wnd.Run(loop)
}
```

4 Тесты

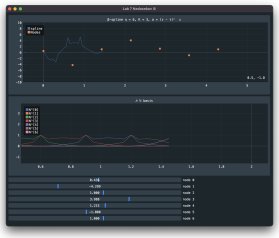

Рис. 1: 1ый тестовый набор

Рис. 2: 2ой тестовый набор

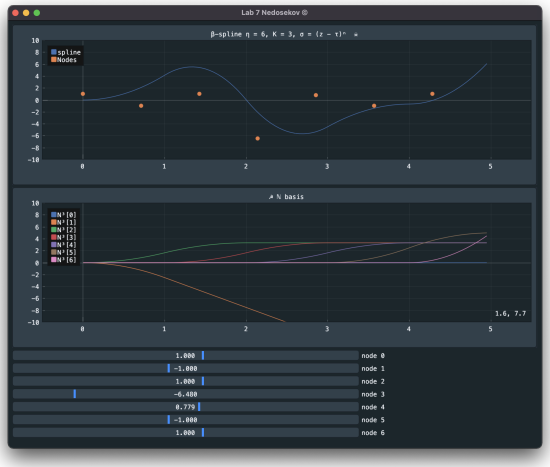

Рис. 3: 3ой тестовый набор

5 Выводы

Выполнив данную лабораторную работу, я познакомился с OpenGl где есть более богатый встроенный инструментарий для отрисовки примитивов.

Список литературы

[1] *Go bindings to various OpenGL*. URL: https://github.com/go-gl/gl (дата обр. 27.10.2021).

[2] *Golang — официальная документация*. URL: https://golang.org/ (дата обр. 27.10.2021).