Министерство образования и науки Российской Федерации
Новосибирский государственный технический университет
Кафедра прикладной информатики

Практическое задание №2
по дисциплине
«Объектно-ориентированное программирование»

Классы

Факультет: ПМИ
Группа: ПМИ-42
Студент: Кислицын И. О.
Преподаватель: Тракимус Ю. В.

Новосибирск
2015

Условие задачи

Модифицировать программу, разработанную в лабораторной работе №1, так чтобы в ней был определён класс, реализующий понятие треугольника в графической системе.

Анализ задачи

Для работы с графикой использовалась свободная реализация OpenGL API (пакет libgl1-mesa-dev версии 11.0.4-1), для работы с окном и обработки ввода с клавиатуры – библиотека SDL2 (пакет libsdl2-dev версии 2.0.2). Был разработан класс `Display`, предназначенный для работы с графикой.

Входные данные

На вход программе подаётся файл в формате:

$(x_1, y_1)(x_2, y_2)(x_3, y_3)$ r g b
$(x_4, y_4)(x_5, y_5)(x_6, y_6)$

где:

- $(x_1, y_1)(x_2, y_2)(x_3, y_3)$ – координаты внешнего треугольника, типа `float`;
- r g b – цвет треугольника (`float`);
- $(x_4, y_4)(x_5, y_5)(x_6, y_6)$ – координаты внутренего треугольника.

Выходные данные

- Если треугольник выходит за границы окна, программа выведет в консоль сообщение об ошибке и завершится.
- Если треугольники не вложены друг в друга, программа выведет в консоль предупреждение и не отреагирует на команду вывода «вычтенных» треугольников.
- В противном случае программа выведет треугольник заданного цвета с заданными координатами.

С помощью клавиш `1, 2 и 3` можно переключать режимы отображения:

- 1 соответствует отображению контура;

- 2 соответствует отображению заполненной фигуры;
- 3 соответствует «вычитанию» треугольников.

Внутреннее представление

- Класс `Display` отвечает за создание окна и работу с графикой. При неполадках, связанных с инициализацией графики, выбрасывает соответствующие исключения.
- Интерфейс `Drawable` описывает объекты, которые могут быть отрисованы с помощью класса `Display`.
- Класс `Triangle` реализует интерфейс `Drawable` и описывает треугольник.

Текст программы

Листинг 1: main.h

```
/* main.h
**
** Here are macros and third-side library includes located.
** Local macros are located where appropriate.
*/

#ifndef _LAB2_DEFS
#define _LAB2_DEFS

#include <cmath>
#include <cfloat> //for FLT_MAX
#include <cstdlib>
#include <iostream>
#include <fstream>
#include <sstream>
#include <string>
#include <utility>
#include <vector>
#include <algorithm>
#include <exception>
#include <SDL2/SDL.h>
#include <GL/gl.h>
#include <GL/glu.h>
#include <glm/glm.hpp>
#include <glm/gtx/matrix_transform_2d.hpp>
```

```
#define WIN_WIDTH   800
#define WIN_HEIGHT    600
#define WIN_TITLE   "Lab␣2"
#define COLOR_BLACK   0.f, 0.f, 0.f
#define COLOR_WHITE   1.f, 1.f, 1.f
#define COLOR_RED   1.f, 0.f, 0.f
#define COLOR_GREEN   0.f, 1.f, 0.f
#define COLOR_BLUE    0.f, 0.f, 1.f
#define COLOR_BGND    0.f, 0.f, 0.f

void terminate(std::string msg);

#endif
```

Листинг 2: main.cc

```
#include "main.h"
#include "Display.h"
#include "Exceptions.h"
#include "Triangle.h"

void terminate(std::string msg)
{
  std::cerr << "[ERROR]␣" << msg << '\n';
  exit(1);
}

std::vector<Triangle*> input(std::string filename, unsigned
   width, unsigned height)
{
  if(filename == "") terminate("Filename␣is␣not␣specified.");
  std::ifstream in(filename, std::ifstream::in);
  std::vector<Triangle*> res;
  res.push_back(new Triangle());
  try
  {
    in >> *res[0];
  }
  catch(TriangleInputException &ex)
  {
    terminate(ex.what());
  }
  float color[3];
  for(int i = 0; i < 3; ++i)
    in >> color[i];
  res[0]->setColor(color[0], color[1], color[2]);
  res.push_back(new Triangle());
  try
  {
```

```
    in >> *res[1];
  }
  catch(TriangleInputException &ex)
  {
    terminate(ex.what());
  }

  if( res[0]->p[0][0] < 0 || res[0]->p[0][0] > width  ||
    res[0]->p[1][0] < 0 || res[0]->p[1][0] > width  ||
    res[0]->p[0][1] < 0 || res[0]->p[0][1] > height ||
    res[0]->p[1][1] < 0 || res[0]->p[1][1] > height )
  {
    terminate("Triangle's␣out␣of␣the␣window.");
  }
  return res;
}

int main(int argc, char **argv)
{
  std::string filename = argc > 1 ? std::string(argv[1]) : "";
  std::vector<Triangle*> tri = input(filename, WIN_WIDTH,
   WIN_HEIGHT);

  Display app;
  try
  {
    app.create(WIN_TITLE, WIN_WIDTH, WIN_HEIGHT,
   SDL_WINDOW_OPENGL | SDL_WINDOW_SHOWN);
  }
  catch(std::exception &ex)
  {
    terminate(ex.what());
  }

  bool run = true;
  SDL_Event event;
  const Uint8* keystate = SDL_GetKeyboardState(NULL);

  tri[0]->setInner(tri[1]);
  app.add(tri[0]);

  double vx = 0, vy = 0, vr = 0;
  while(run)
  {
    app.draw();

    if(SDL_PollEvent(&event) != 0)
    {
      switch(event.type)
```

```
{
  case SDL_KEYDOWN:
    switch(event.key.keysym.sym)
    {
      case SDLK_LEFT:
      case SDLK_a:
        vx = -1;
      break;
      case SDLK_RIGHT:
      case SDLK_d:
        vx = 1;
      break;
      case SDLK_UP:
      case SDLK_w:
        vy = -1;
      break;
      case SDLK_DOWN:
      case SDLK_s:
        vy = 1;
      break;
      case SDLK_q:
        vr = -0.01;
      break;
      case SDLK_e:
        vr = 0.01;
      break;
      case SDLK_1:
        tri[0]->setMode(TRI_LINE);
      break;
      case SDLK_2:
        tri[0]->setMode(TRI_FULL);
      break;
      case SDLK_3:
        tri[0]->setMode(TRI_NESTED);
      break;
    }
  break;
  case SDL_KEYUP:
    switch(event.key.keysym.sym)
    {
      case SDLK_LEFT:
      case SDLK_RIGHT:
      case SDLK_a:
      case SDLK_d:
        vx = 0;
      break;
      case SDLK_UP:
      case SDLK_DOWN:
      case SDLK_w:
```

```
            case SDLK_s:
              vy = 0;
            break;
            case SDLK_q:
            case SDLK_e:
              vr = 0;
            break;
          }
        break;
        case SDL_QUIT:
          run = false;
        break;
      }
    }
    SDL_PumpEvents();

    tri[0]->translate(vx, vy);
    tri[0]->rotate(vr);
  }
}
```

Листинг 3: Drawable.h

```
/* Drawable.h
 *
 * For objects that can be drawn.
 */

#ifndef _LAB2_DRAWABLE
#define _LAB2_DRAWABLE

class Drawable
{
  public:
    Drawable() {}
    virtual ~Drawable() {}
    virtual void draw() {}
};

#endif
```

Листинг 4: Display.h

```
/* Display.h
 *
 * Display class realizes every interaction between SDL and
   OpenGL.
 */

#ifndef _LAB2_DISPLAY
#define _LAB2_DISPLAY
```

```
#include "main.h"
#include "Exceptions.h"
#include "Drawable.h"

class Display
{
  public:
    Display();

    /*
     * Constructor that calls create(...) function.
     */
    Display(std::string _title, unsigned int _w, unsigned int
   _h, Uint32 flags) throw(SDLInitException,
   SDLWindowCreateException, GLContextCreateException);
    virtual ~Display();

    /*
     * Creates window and GL context.
     *
     * Args:
     *  string _title - title of the window.
     *  unsigned int _w - width of the window.
     *  unsigned int _h - height of the window.
     *  Uint32 flags - flags of the window (sheck out
   SDL_CreateWindow(...) 6th argument)
     *
     * Throws:
     *  SDLInitException
     *  SDLWindowCreateException
     *  GLContextCreateException
     */
    void create(std::string _title, unsigned int _w, unsigned
   int _h, Uint32 flags) throw(SDLInitException,
   SDLWindowCreateException, GLContextCreateException);

    /*
     * Clears buffers, prepares window for drawing.
     */
    void clear();

    /*
     * Swaps buffers.
     */
    void swap();

    /*
     * Draws everything.
```

```
     */
    void draw();

    /*
     * Add drawable element.
     */
    void add(Drawable *elem);

  private:
    unsigned int w, h;         // window's width and height
    std::string title;         // window's title
    SDL_Window *win = NULL;      // SDL's window object
    SDL_GLContext context = NULL; // SDL's GL context
    std::vector<Drawable*> dlist; // list of elements to draw
};

#endif
```

Листинг 5: Display.cc

```
#include "main.h"

#include "Display.h"
#include "Exceptions.h"

Display::Display() {}

Display::Display(std::string _title, unsigned int _w, unsigned
   int _h, Uint32 flags) throw(SDLInitException,
   SDLWindowCreateException, GLContextCreateException)
{
  create(_title, _w, _h, flags);
}

Display::~Display()
{
  SDL_GL_DeleteContext(context);
  SDL_Quit();
}

void Display::create(std::string _title, unsigned int _w,
   unsigned int _h, Uint32 flags) throw(SDLInitException,
   SDLWindowCreateException, GLContextCreateException)
{
  w = _w; h = _h; title = _title;
  if(SDL_Init(SDL_INIT_EVERYTHING) != 0)
   throw(SDLInitException());
  SDL_GL_SetAttribute(SDL_GL_DOUBLEBUFFER, 1);

  win = SDL_CreateWindow(title.c_str(), SDL_WINDOWPOS_CENTERED,
```

```
   SDL_WINDOWPOS_CENTERED, w, h, flags);
  if(win == NULL) throw(SDLWindowCreateException());

  context = SDL_GL_CreateContext(win);
  if(context == NULL) throw(GLContextCreateException());

  SDL_GL_SetSwapInterval(1);    //enables vertical syncronization
  glClearColor(COLOR_BGND, 0);

  //initializing 2D
  glMatrixMode(GL_PROJECTION);
  glLoadIdentity();
  glOrtho(0, w, h, 0, 0, 1);
  glMatrixMode(GL_MODELVIEW);
  glDisable(GL_DEPTH_TEST);
  glLineWidth(3.f);
}

void Display::clear()
{
  glClear(GL_COLOR_BUFFER_BIT);
  glLoadIdentity();
}

void Display::swap()
{
  if(!win) terminate("Can't␣swap␣Display␣object!␣Forgot␣to␣
   initialize␣it?");
  SDL_GL_SwapWindow(win);
}

void Display::draw()
{
  clear();
  std::for_each(dlist.begin(), dlist.end(), [](Drawable *elem)
    {
      elem->draw();
    });
  swap();
}

void Display::add(Drawable *elem)
{
  dlist.push_back(elem);
}
```

Листинг 6: Triangle.h

```
#ifndef _LAB2_TRIANGLE
#define _LAB2_TRIANGLE
```

```
#include "main.h"
#include "Drawable.h"
#include "Exceptions.h"

enum tri_mode
{
  TRI_LINE,
  TRI_FULL,
  TRI_NESTED
};

enum sides
{
  LEFT,
  RIGHT,
  UP,
  DOWN
};

class Triangle : public Drawable
{
  public:
    Triangle();
    Triangle(glm::vec2 x, glm::vec2 y, glm::vec2 z);
    void draw();
    void setColor(float r, float g, float b);
    void setInner(Triangle *in);
    void rotate(double a);
    void translate(double x, double y);
    void setMode(tri_mode m);

    friend std::istream &operator>>(std::istream &stream,
   Triangle &obj) throw(TriangleInputException);
    friend std::vector<Triangle*> input(std::string filename,
   unsigned width, unsigned height);
  private:
    glm::vec2 p[3];
    float color[3];       //color in RGB order
    Triangle *inner;      //nested triangle
    tri_mode mode;
    glm::vec2 trnsl;      //coordinates of transfer vector
    float angle;        //for rotation
    bool blocked[4];      //side that's blocked
    glm::vec3 blocked_vtx;    //vertex that's blocked (rotated,
   but not translated!)
    void sort_inner();
};
```

```
#endif
```

Листинг 7: Triangle.cc

```
#include "main.h"
#include "Triangle.h"
#include "Exceptions.h"

bool sgn(float x)
{
  return x > 0.f ? true : false;
}

Triangle::Triangle(): inner(NULL), mode(TRI_FULL), angle(0.0)
{
  trnsl[0] = 0.0;
  trnsl[1] = 0.0;
  for(int i = 0; i < 4; ++i) blocked[i] = false;
}

Triangle::Triangle(glm::vec2 x, glm::vec2 y, glm::vec2 z):
   inner(NULL), mode(TRI_FULL), angle(0.0)
{
  p[0] = x;
  p[1] = y;
  p[2] = z;
  trnsl[0] = 0.0;
  trnsl[1] = 0.0;
  for(int i = 0; i < 4; ++i) blocked[i] = false;
}

void Triangle::draw()
{
  static glm::vec3 old_render[3];
  glColor3f(color[0], color[1], color[2]);

  glPushMatrix();

  glm::mat3 m_trans(1.0);
  glm::mat3 m_rot(1.0);

  if(0.0 != trnsl[0] || 0.0 != trnsl[1]) m_trans =
   glm::translate(glm::mat3(1.0), trnsl);
  if(0.0 != angle)
  {
    glm::vec2 c((p[0][0]+p[1][0]+p[2][0])/3,
   (p[0][1]+p[1][1]+p[2][1])/3);
    m_rot =
   glm::translate(glm::rotate(glm::translate(glm::mat3(1.0), c),
   angle), -c);
```

```
}

glm::vec3 render[3];
for(int i = 0; i < 3; ++i) render[i] = m_trans * m_rot *
 glm::vec3(p[i][0], p[i][1], 1);

/* Check if the triangle is blocked by window's border */
for(int i = 0; i < 3; ++i)
{
  if(render[i][0] <= 0)
  {
    blocked[LEFT] = true;
    blocked_vtx = m_rot * glm::vec3(p[i][0], p[i][1], 1);
  }
  if(render[i][0] >= WIN_WIDTH)
  {
    blocked[RIGHT] = true;
    blocked_vtx = m_rot * glm::vec3(p[i][0], p[i][1], 1);
  }
  if(render[i][1] <= 0)
  {
    blocked[UP] = true;
    blocked_vtx = m_rot * glm::vec3(p[i][0], p[i][1], 1);
  }
  if(render[i][1] >= WIN_HEIGHT)
  {
    blocked[DOWN] = true;
    blocked_vtx = m_rot * glm::vec3(p[i][0], p[i][1], 1);
  }
}

if(mode == TRI_LINE) glBegin(GL_LINES);
else glBegin(GL_TRIANGLES);

for(short int i = 0; i < 3; ++i)
{
  glVertex2i(render[i][0], render[i][1]);
  if(mode == TRI_LINE) glVertex2i(render[(i+1)%3][0],
 render[(i+1)%3][1]);
}
if(mode == TRI_NESTED)
{
  if(inner != NULL)
  {
    glm::vec3 nest_rend[3];
    for(int i = 0; i < 3; ++i) nest_rend[i] = m_trans * m_rot
 * glm::vec3(inner->p[i][0], inner->p[i][1], 1);
    glColor3f(COLOR_BGND);
    for(short int i = 0; i < 3; ++i)
```

```
        glVertex2i(nest_rend[i][0], nest_rend[i][1]);
    }
  }
  glEnd();
  glPopMatrix();
}

void Triangle::setColor(float r, float g, float b)
{
  color[0] = r;
  color[1] = g;
  color[2] = b;
}

void Triangle::setInner(Triangle *in)
{
  inner = in;
  float a[3], b[3];   //ax + b for all three sides of outter
   triangle
  bool one_side[9];   //one_side[i*3 + j], i ~ point of tri_in,
   j ~ line
  for(int i = 0; i < 3; ++i)
  {
    a[i] = (float)(p[(i+1)%3][1] -
   p[i][1])/(float)(p[(i+1)%3][0] - p[i][0]);
    b[i] = (float)(p[i][1]*p[(i+1)%3][0] -
   p[(i+1)%3][1]*p[i][0])/(float)(p[(i+1)%3][0] - p[i][0]);
  }
  /* Checking if all three points of the second triangle is into
   the first one */
  for(int i = 0; i < 3; ++i)
  {
    for(int j = 0; j < 3; ++j)
    {
      float y_0 = inner->p[i][0]*a[j] + b[j];
      float y_1 = p[(j+2)%3][0]*a[j] + b[j];
      one_side[i*3 + j] = sgn(y_0 - inner->p[i][1]) == sgn(y_1 -
   p[(j+2)%3][1]);
    }
  }
  bool into = true;
  for(int i = 0; i < 9; ++i) into = into && one_side[i];
  if(!into)
  {
    printf("\nNot␣nested!");
    inner = NULL;
  }
}
```

```
void Triangle::rotate(double a)
{
  static const glm::vec2 c((p[0][0]+p[1][0]+p[2][0])/3,
   (p[0][1]+p[1][1]+p[2][1])/3);
  if(a > 0.0)
  {
    if(blocked[LEFT])
      if(c[1] < blocked_vtx[1]) a = 0.0;
      else blocked[LEFT] = false;
    if(blocked[UP])
      if(c[0] > blocked_vtx[0]) a = 0.0;
      else blocked[UP] = false;
    if(blocked[RIGHT])
      if(c[1] > blocked_vtx[1]) a = 0.0;
      else blocked[RIGHT] = false;
    if(blocked[DOWN])
      if(c[0] < blocked_vtx[0]) a = 0.0;
      else blocked[DOWN] = false;
  }
  if(a < 0.0)
  {
    if(blocked[LEFT])
      if(c[1] > blocked_vtx[1]) a = 0.0;
      else blocked[LEFT] = false;
    if(blocked[UP])
      if(c[0] < blocked_vtx[0]) a = 0.0;
      else blocked[UP] = false;
    if(blocked[RIGHT])
      if(c[1] < blocked_vtx[1]) a = 0.0;
      else blocked[RIGHT] = false;
    if(blocked[DOWN])
      if(c[0] > blocked_vtx[0]) a = 0.0;
      else blocked[DOWN] = false;
  }

  angle += a;
  if(angle >= 360) angle -= 360;
  else if(angle <= -360) angle += 360;
}

void Triangle::translate(double x, double y)
{
  if(x > 0.0)
  {
    if(blocked[RIGHT]) x = 0.0;
    blocked[LEFT] = false;
  }
  if(x < 0.0)
  {
```

```
    if(blocked[LEFT]) x = 0.0;
    blocked[RIGHT] = false;
  }

  if(y > 0.0)
  {
    if(blocked[DOWN]) y = 0.0;
    blocked[UP] = false;
  }
  if(y < 0.0)
  {
    if(blocked[UP]) y = 0.0;
    blocked[DOWN] = false;
  }

  trnsl[0] += x;
  trnsl[1] += y;
}

void Triangle::setMode(tri_mode m)
{
  mode = m;
}

std::istream &operator>>(std::istream &stream, Triangle &obj)
   throw(TriangleInputException)
{
  char cache;
  for(int i = 0; i < 3; ++i)
  {
    stream >> cache;
    if(cache != '(') throw(TriangleInputException());
    stream >> obj.p[i][0] >> cache;
    if(cache != ',') throw(TriangleInputException());
    stream >> obj.p[i][1] >> cache;
    if(cache != ')') throw(TriangleInputException());
  }
  return stream;
}
```

Листинг 8: Exceptions.h

```
// Exceptions.h

#ifndef _LAB2_EXCEPTION
#define _LAB2_EXCEPTION

#include "main.h"

/*
```

```
** Generated when SDL can't be initialized.
*/
class SDLInitException : public std::exception
{
  public:
    SDLInitException();
    virtual ~SDLInitException() throw();
    virtual const char* what() throw();
};

/*
** Generated when window can't be created.
*/
class SDLWindowCreateException : public std::exception
{
  public:
    SDLWindowCreateException();
    virtual ~SDLWindowCreateException() throw();
    virtual const char* what() throw();
};

/*
** Generates when GL context can't be created.
*/
class GLContextCreateException : public std::exception
{
  public:
    GLContextCreateException();
    virtual ~GLContextCreateException() throw();
    virtual const char* what() throw();
};

class TriangleInputException : public std::exception
{
  public:
    TriangleInputException();
    virtual ~TriangleInputException() throw();
    virtual const char* what() throw();
};

#endif
```

Листинг 9: Exceptions.cc

```
#include "Exceptions.h"

SDLInitException::SDLInitException(): exception() {}

SDLInitException::~SDLInitException() throw() {}
```

```
const char* SDLInitException::what() throw()
{
  std::ostringstream out;
  out << "SDL␣initialization␣failed:␣" << SDL_GetError();
  return out.str().c_str();
}

SDLWindowCreateException::SDLWindowCreateException():
   exception() {}

SDLWindowCreateException::~SDLWindowCreateException() throw() {}

const char* SDLWindowCreateException::what() throw()
{
  std::ostringstream out;
  out << "SDL␣window␣creation␣failed:␣" << SDL_GetError();
  return out.str().c_str();
}

GLContextCreateException::GLContextCreateException():
   exception() {}

GLContextCreateException::~GLContextCreateException() throw() {}

const char* GLContextCreateException::what() throw()
{
  std::ostringstream out;
  out << "GL␣context␣creation␣failed:␣" << SDL_GetError();
  return out.str().c_str();
}

TriangleInputException::TriangleInputException(): exception() {}

TriangleInputException::~TriangleInputException() throw() {}

const char* TriangleInputException::what() throw()
{
  std::ostringstream out;
  out << "Trouble␣with␣triangle␣formatted␣input.";
  return out.str().c_str();
}
```

Для компиляции программы используется `Makefile`:

```
CC=g++
CXXFLAGS=-std=c++11 -lSDL2 -lSDL2_image -lGL -lGLU
SOURCES=./src/*.cc

all:
  $(CC) $(CXXFLAGS) -g $(SOURCES)
```

Набор тестов

№	Входные данные	Назначение
1	`(10,10)(120,30)(30,60) 33 FF 33` `(50,20)(70,40)(30,40)`	Корректные входные данные
2	`(10,10)(120,30)(30,60) 33 FF 33` `(50,10)(70,40)(30,40)`	Треугольники не вложены друг в друга
3	`(10,10)(120,800)(600,60) 33 FF 33` `(50,20)(80,50)(30,40)`	Треугольник выходит за границы окна

Результаты работы программы