Московский Авиационный Институт

(Национальный Исследовательский Университет)

Кафедра 806 «Вычислительная информатика и программирование»

Факультет: «Информационные технологии и прикладная математика»

Лабораторная работа

Дисциплина: «Объектно-ориентированное программирование»

III семестр

Задание 7: «Проектирование структуры классов»

|  |  |
| --- | --- |
| Группа: | М8О-208Б-18, №3 |
| Студент: | Овечкин Виталий Андреевич |
| Преподаватель: | Журавлёв Андрей Андреевич |
| Оценка: |  |
| Дата: | 30.12.2019 |

Москва, 2019

1. **Задание** (*вариант № 19* ):

Реализовать графический редактор, позволяющий отрисовывать четырёхугольник, многоугольник, круг, ломаную, задавать их цвета, реализовать функцию undo

1. **Адрес репозитория на GitHub** https://github.com/vitalouivi/oop\_exercise\_7
2. **Код программы на С++**

main.cpp

#include <array>

#include <memory>

#include <vector>

#include <stack>

#include "sdl.h"

#include "imgui.h"

#include "quadrangle.h"

#include "curve\_line.h"

#include "circle.h"

#include "polygon.h"

#include "color.h"

#include "Document.h"

int main() {

sdl::renderer renderer("Editor");

bool quit = false;

std::unique\_ptr<builder> active\_builder = nullptr;

bool active\_deleter = false;

const int32\_t file\_name\_length = 128;

char file\_name[file\_name\_length] = "";

int32\_t remove\_id = 0;

std::vector<int> color(3);

Document currentDocument;

while (!quit) {

renderer.set\_color(0, 0, 0);

renderer.clear();

sdl::event event;

while (sdl::event::poll(event)) {

sdl::quit\_event quit\_event;

sdl::mouse\_button\_event mouse\_button\_event;

if (event.extract(quit\_event)) {

quit = true;

break;

} else if (event.extract(mouse\_button\_event)) {

if (active\_builder && mouse\_button\_event.button() == sdl::mouse\_button\_event::left && mouse\_button\_event.type() == sdl::mouse\_button\_event::down) {

std::unique\_ptr<figure> figure = active\_builder->add\_vertex(vertex{mouse\_button\_event.x(), mouse\_button\_event.y()});

if (figure) {

figure -> setColor(color);

currentDocument.addFigure(std::move(figure));

active\_builder = nullptr;

}

}

if (active\_builder && mouse\_button\_event.button() == sdl::mouse\_button\_event::right && mouse\_button\_event.type() == sdl::mouse\_button\_event::down) {

std::unique\_ptr<figure> figure = active\_builder->add\_vertex(vertex{-1, -1});

if (figure) {

figure -> setColor(color);

currentDocument.addFigure(std::move(figure));

active\_builder = nullptr;

}

}

if (active\_deleter && mouse\_button\_event.button() == sdl::mouse\_button\_event::left && mouse\_button\_event.type() == sdl::mouse\_button\_event::down) {

currentDocument.removeByClick(vertex{mouse\_button\_event.x(), mouse\_button\_event.y()});

active\_deleter = false;

}

}

}

currentDocument.render(renderer);

ImGui::Begin("Menu");

if (ImGui::Button("New canvas")) {

currentDocument.clear();

}

ImGui::InputText("File name", file\_name, file\_name\_length - 1);

if (ImGui::Button("Save")) {

std::ofstream os(file\_name);

if (os) {

currentDocument.Save(os);

}

}

ImGui::SameLine();

if (ImGui::Button("Load")) {

std::ifstream is(file\_name);

if (is) {

currentDocument.Load(is);

}

}

ImGui::InputInt("R", &color[0]);

ImGui::InputInt("G", &color[1]);

ImGui::InputInt("B", &color[2]);

if (ImGui::Button("Red")) {

color[0] = 255;

color[1] = 0;

color[2] = 0;

}

ImGui::SameLine();

if (ImGui::Button("Green")) {

color[0] = 0;

color[1] = 255;

color[2] = 0;

}

ImGui::SameLine();

if (ImGui::Button("Blue")) {

color[0] = 0;

color[1] = 0;

color[2] = 255;

}

if (ImGui::Button("Quadrangle")) {

active\_builder = std::make\_unique<quadrangle\_builder>();

}

if (ImGui::Button("Broken Line")) {

active\_builder = std::make\_unique<curve\_line\_builder>();

}

if (ImGui::Button("Circle")) {

active\_builder = std::make\_unique<circle\_builder>();

}

if (ImGui::Button("Polygon")) {

active\_builder = std::make\_unique<polygon\_builder>();

}

ImGui::InputInt("Remove id", &remove\_id);

if (ImGui::Button("Remove")) {

if (remove\_id >= 0 && remove\_id < (currentDocument.figures).size()) {

currentDocument.removeFigure(remove\_id);

}

}

if (ImGui::Button("Remove by click")) {

active\_deleter = true;

}

if (ImGui::Button("UNDO")) {

currentDocument.undo();

}

ImGui::End();

renderer.present();

}

}

Figure.h

#ifndef D\_FIGURE\_H

#define D\_FIGURE\_H 1

#include "vertex.h"

#include "sdl.h"

#include <array>

#include <vector>

#include <memory>

//#include <iostream>

//#include <fstream>

#include <string>

struct figure {

virtual void render(const sdl::renderer& renderer) const = 0;

virtual void save(std::ostream& os) const = 0;

virtual bool isPointInside(vertex v) const = 0;

virtual void setColor(std::vector<int> color) = 0;

virtual ~figure() = default;

};

#endif // !D\_FIGURE\_H

vertex.h

#ifndef D\_VERTEX\_H

#define D\_VERTEX\_H 1

#include <memory>

#include <fstream>

#include <iostream>

struct vertex {

int32\_t x, y;

};

inline std::istream& operator>> (std::istream& is, vertex& p) {

is >> p.x >> p.y;

return is;

}

#endif // !D\_VERTEX\_H

circle.h

#ifndef D\_CIRCLE\_H

#define D\_CIRCLE\_H 1

#include "figure.h"

#include "builder.h"

#include <math.h>

struct circle : figure {

circle(const std::vector<vertex>& vertices);

void setColor(std::vector<int> color) override;

void render(const sdl::renderer& renderer) const override;

void save(std::ostream& os) const override;

bool isPointInside(vertex v) const override;

private:

std::vector<int> color\_;

std::vector<vertex> vertices\_;

int radius;

};

struct circle\_builder : builder {

std::unique\_ptr<figure> add\_vertex(const vertex& v) override;

std::string getType() override;

private:

int32\_t n\_ = 0;

std::vector<vertex> vertices\_;

};

#endif

document.h

#pragma once

#ifndef D\_DOCUMENT\_H

#define D\_DOCUMENT\_H 1

#include <array>

#include <fstream>

#include <iostream>

#include <memory>

#include <vector>

#include <stack>

#include "sdl.h"

#include "imgui.h"

#include"quadrangle.h"

#include "curve\_line.h"

#include "circle.h"

#include "polygon.h"

struct Command;

struct CommandAdd;

struct CommandRemove;

struct Document;

struct Document {

public:

Document() = default;

void addFigure(std::unique\_ptr<figure> fig);

void removeFigure(int id);

void removeByClick(vertex v);

void undo();

void Save(std::ofstream& os);

void Load(std::ifstream& is);

void render(const sdl::renderer& renderer);

void clear();

std::vector<std::shared\_ptr<figure>> figures;

std::stack<std::unique\_ptr<Command>> commandStack;

};

struct Command {

virtual ~Command() = default;

virtual void undo() = 0;

};

struct CommandAdd : Command {

int index\_\_;

Document \* doc\_\_ = new Document();

CommandAdd(int index, Document \* doc) : index\_\_(index), doc\_\_(doc) {}

void undo() {

(doc\_\_ -> figures).erase((doc\_\_ -> figures).begin() + index\_\_);

}

};

struct CommandRemove : Command {

Document \* doc\_\_;

int index\_\_;

std::shared\_ptr<figure> figure\_\_ = nullptr;

CommandRemove(int index, std::shared\_ptr<figure> figure\_, Document \* doc) : index\_\_(index), figure\_\_(figure\_), doc\_\_(doc) {}

void undo() {

if (index\_\_ > (doc\_\_ -> figures).size() - 1)

(doc\_\_ -> figures).push\_back(std::move(figure\_\_));

else

(doc\_\_ -> figures).insert((doc\_\_ -> figures).begin() + index\_\_, std::move(figure\_\_));

}

};

#endif

CMakeLists.txt

cmake\_minimum\_required(VERSION 3.0)

project(lab7)

set(CMAKE\_CXX\_STANDARD\_REQUIRED YES)

set(CMAKE\_CXX\_STANDARD 14)

add\_executable(lab7

main.cpp

sdl.cpp

circle.cpp

curve\_line.cpp

polygon.cpp

quadrangle.cpp

Document.cpp

)

add\_subdirectory(lib/SDL2/)

target\_link\_libraries(lab7 SDL2-static)

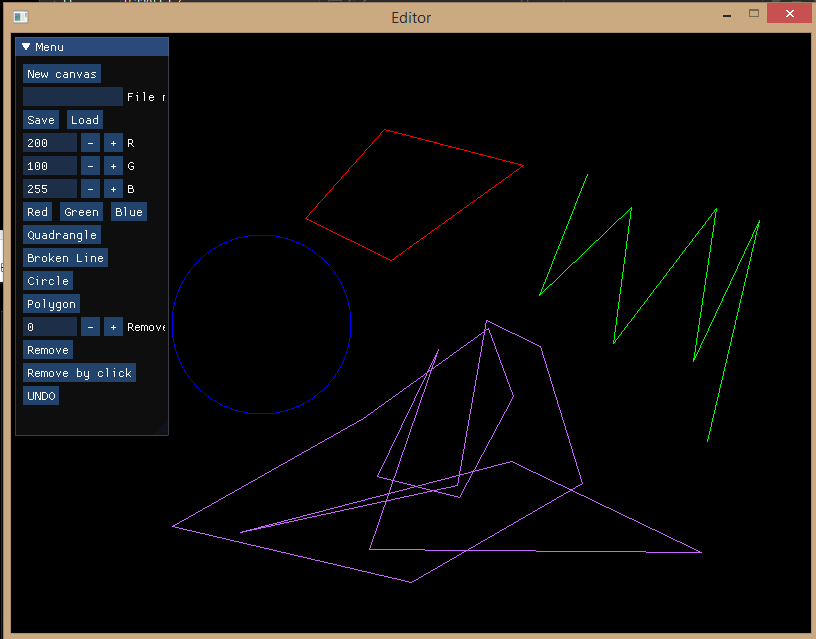
target\_include\_directories(lab7 PRIVATE **${SDL2\_INCLUDE\_DIR}**)

add\_subdirectory(lib/imgui/)

target\_include\_directories(imgui PRIVATE lib/SDL2/include/)

target\_link\_libraries(lab7 imgui)

Пример Использования



1. **Объяснение результатов работы программы - вывод**

В figure.h задаётся базовый класс Figure задающий структуру для классов — наследников — Quadrangle, Polygon и Circle.

Наследование позволяет избежать дублирования кода при написании классов, т. к. класс может использовать переменные и методы другого класса как свои собственные.

Программа представляет собой визуальное приложение, способное строить прямоугольник, ромб, трапецию, ломаную линию, многоугольник и круг. Возможно удаление по индексу, клику внутрь фигуры , операция undo, загрузка и сохранение в файл. Я познакомился с визуальными библиотеками в C++, углубил свои знания в области полиморфизма, познакомился с написанием и сборкой многофайльвых проектов, а так же подключением сторонних библиотек. Также в очередной раз я практиковался в разбиении кода на модули.