**Курсовая работа по дисциплине «»**

**«Разработка игры «Тараканьи бега» на C++»**

2019 г

Оглавление

[Задание 3](#_Toc12466628)

[Введение 4](#_Toc12466629)

[1 Иерархия классов 5](#_Toc12466630)

[2 Описание программы 7](#_Toc12466631)

[Заключение 10](#_Toc12466632)

[Список используемых источников 11](#_Toc12466633)

[Приложение А. Листинг программы 12](#_Toc12466634)

# Задание

Необходимо разработать игру «Тараканьи бега» (вариант 6) для Windows на языке C++, используя графическую библиотеку SDL и среду разработки DevC++.

Правила игры.

В игре могут принимать участие как один, так и несколько игроков, которые делают ставки на тех тараканов, которые им симпатичны. В каждом забеге участвуют от двух до пяти тараканов, о каждом из которых имеется следующая информация: кличка, количество забегов, количество побед. Каждый таракан бежит по своей дорожке в произвольном направлении с произвольной скоростью. Победителем является таракан, первым пересекший финишную черту. Игроки, которые сделали ставки на таракана-победителя, получают выигрыш.

Замечание. По желанию программиста можно составить базу данных игроков, содержащую информацию об их капитале и соответственно принимать решения в возможной сумме ставок.

# Введение

Данная работа посвящена изучению современных подходов к разработке программных систем на языке высокого уровня С++ с использованием в качестве среды разработки бесплатно распространяемой системы DevС++.

Целью работы является закрепление и углубление знаний, полученных при изучении языка C++, развитие практических навыков при работе с функциями графической библиотеки SDL, а также при выборе представления исходных данных, тестировании, отладке программы и оформлении сопроводительной документации.

# Иерархия классов

Все классы в программе произведены от двух базовых классов – Window, предка всех окон, и ScreenObject, предка всех отображаемых в окне объектов.

Класс Window инкапсулирует такие примитивы SDL как SDL\_Window, и SDL\_Surface, сохраняя соответствующие указатели в качестве защищённых членов класса. Конструктор принимает два обязательных параметра – размеры экрана и один необязательный, заголовок окна. Кроме того, этот класс содержит общие для всех окон признак продолжения цикла обработки сообщений SDL и список объектов, а также чисто виртуальную функцию show(), которую каждый класс перегружает для отображения конкретного содержимого.

В программе присутствует 3 окна – стартовый экран с заставкой, окно ввода имени пользователя и главный игровой экран, им соответствуют классы StartScreen, InputWindow и Game, каждый из которых унаследован от класса Window.

Каждый визуальный объект на экране унаследован от базового класса ScreenObject. Этот класс инкапсулирует примитив поверхности SDL, SDL\_Surface, и базовые операции с ней – сохранение, управление расположением на экране, отображение на родительской поверхности и очистка занимаемых ресурсов.

Все имеющиеся в игре графические объекты классифицируются на картинки (класс Image), Надписи (Text), Пункты меню (MenuItem) и игровой объект «Таракан» (Cockroach). Все соответствующие классы наследуют от базового класса ScreenObject.

На главном игровом экране присутствуют объекты трёх видов – Image (фоновый рисунок и крестик), Text (надписи) и Cockroach (тараканы). Экземпляры классов Image, Text и Cockroach помещаются в список объектов окна, что позволяет отображать их в рамках общей сцены в общем цикле, вызывая их метод show().

Отдельный класс Menu управляет списком тестовых пунктов (MenuItem) и позволяет пользователю выбрать один из них. Это класс отображает пользовательское меню на поверхности родительского окна.

В отдельном модуле, Utils, собраны функции инициализации/освобождения графических библиотек и функция вывода текста ошибки на экран.

Исходный код программы приведён в «Приложении А».

# Описание программы

После запуска программы на экране отображается заставка с названием игры, именем и учебной группой автора, её вид показан на рисунке 1.



Рисунок 1 – Вид стартового окна с заставкой

Это окно закрывается клавишей Esc или щелчком левой клавыиши мыши (ЛКМ) в любом месте окна, после чего программа запросит имя пользователя, как показано на рисунке 2.

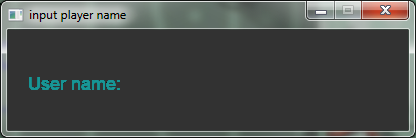


Рисунок 2 – Запрос имени игрока

На этом этапе можно отменить запуск игры, нажав клавишу Esc. Если ввести имя пользователя и нажать Enter, программа запустит основное окно игры, и игра сразу начнётся.

По краям игрового поля появятся сразу 5 бегущих с разной скоростью тараканов, а в центре отобразится прицел, как показано на рисунке 3.

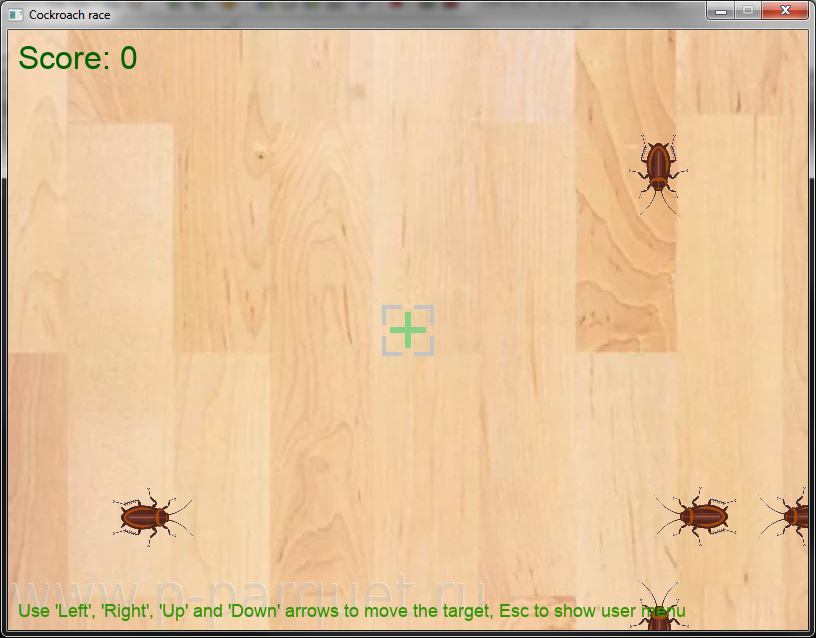


Рисунок 3 – Вид игрового поля

В левом верхнем углу экрана отображается текущее количество очков – число убитых а данный момент тараканов. Внизу игрового поля приведена краткая справка по клавишам управления.

Крестиком прицела можно управлять с помощью стрелок – клавиш «Влево», «Вправо», «Вверх» и «Вниз».Клавиша «Esc» выведет меню, запрашивающее подтверждение на выход.

Чтобы «прихлопнуть» бегущего таракана, необходимо просто навести на него прицел. В момент совпадения одна из клавиш со стрелками должна быть нажата, т.е. под неподвижным прицелом тараканы могут спокойно пробегать.

Игра длится одну минуту, по её завершении имя пользователя и количество убитых тараканов сохраняются в двоичном файле. Сохраняются только 10 последних результатов.

# Заключение

В результате работы создана игра, демонстрирующая принципы разработки приложений Windows на языке C++, а также особенности работы с функциями графической библиотеки SDL.

Приобретены практические навыки разработки приложений, использования языковых конструкций С++, отладки программ и составления сопроводительной документации. На базе практического применения, закреплены базовые знания, полученные ранее, при изучении теории.

# Список используемых источников

1. Герберт Шилдт Искусство программирования на C++ / Герберт Шилдт. - М.: БХВ-Петербург, 2005. - 496 c.

2. Герб Саттер Решение сложных задач на С++ / Герб Саттер. - М.: Вильямс, 2015. - 400 c.

3. Александр Чиртик Программирование на C++. Трюки и эффекты / Александр Чиртик. - М.: "Издательство "Питер", 2010. - 352 c.

4. Lazy Foo’ Production <http://lazyfoo.net/SDL_tutorials/index.php>

5. ГОСТ 7.32-2001 Отчёт о научно-исследовательской работе: структура и правила оформления.

6. ГОСТ 2.105-95 Общие требования к текстовым документам.

# Приложение А. Листинг программы

main.cpp

#include <iostream>

#include "Game.h"

#include "StartScreen.h"

#include "InputWindow.h"

#include "Utils.h"

static const char\* gameName = "Cockroach race";

int main(int argc, char\* argv[])

{

if (!initializeGraphics()) {

std::cerr << "Failed to initialize graphics" << std::endl;

return 1;

}

StartScreen screen(600, 400, gameName);

screen.show();

std::string userName = "";

bool requestUserName = true;

while (requestUserName) {

InputWindow dialog(400, 100, "input player name");

dialog.show();

if (!dialog.confirmed()) {

std::cout << "cancelled by user" << std::endl;

return 0;

}

userName = dialog.input();

if (userName.size() > 0) {

requestUserName = false;

}

else {

std::cout << "user name is empty" << std::endl;

}

}

std::cout << "user: " << userName << std::endl;

Game game(800, 600, gameName, userName);

game.show();

finalizeGraphics();

return 0;

}

ScreenObject.h

#pragma once

#include <SDL\_rect.h>

struct SDL\_Surface;

class ScreenObject

{

public:

ScreenObject(SDL\_Surface\* screen);

virtual ~ScreenObject();

// нарисовать себя на экране

virtual void draw(SDL\_Rect\* dest = nullptr);

// очистить поверхность

void clear();

// методы доступа

SDL\_Surface\* screen() const {

return \_screen;

}

SDL\_Surface\* surface() const {

return \_surface;

}

void surface(SDL\_Surface\* value);

int x() const {

return \_pos.x;

}

void x(int value) {

\_pos.x = value;

}

int y() const {

return \_pos.y;

}

void y(int value) {

\_pos.y = value;

}

int w() const {

return \_pos.w;

}

int h() const {

return \_pos.h;

}

SDL\_Rect pos() const {

return \_pos;

}

// переместить по координатам

void locate(int x, int y) {

\_pos.x = x;

\_pos.y = y;

}

// переместить по оси X

void moveX(int delta) {

\_pos.x += delta;

}

// переместить по оси Y

void moveY(int delta) {

\_pos.y += delta;

}

// попадают координаты на поверхность?

bool inside(int x, int y) const {

return (x >= \_pos.x && x <= \_pos.x + \_pos.w && y >= \_pos.y && y <= \_pos.y + \_pos.h);

}

protected:

SDL\_Surface\* \_screen = nullptr;

SDL\_Surface\* \_surface = nullptr;

SDL\_Rect \_pos = { 0, 0, 0, 0 };

};

ScreenObject.cpp

#include "ScreenObject.h"

#include <SDL.h>

ScreenObject::ScreenObject(SDL\_Surface\* screen): \_screen(screen), \_surface(nullptr)

{

}

ScreenObject::~ScreenObject()

{

clear();

}

void ScreenObject::surface(SDL\_Surface\* value) {

clear();

\_surface = value;

if (\_surface) {

\_pos.w = \_surface->clip\_rect.w;

\_pos.h = \_surface->clip\_rect.h;

}

}

void ScreenObject::draw(SDL\_Rect\* dest)

{

if (\_surface) {

if (dest) {

SDL\_BlitSurface(\_surface, NULL, \_screen, dest);

}

else {

SDL\_BlitSurface(\_surface, NULL, \_screen, &\_pos);

}

}

}

void ScreenObject::clear() {

if (\_surface) {

SDL\_FreeSurface(\_surface);

\_surface = nullptr;

}

}

Image.h

#pragma once

#include "ScreenObject.h"

#include <string>

class Image : public ScreenObject

{

public:

Image(SDL\_Surface\* screen, const char\* fileName);

~Image() = default;

std::string fileName() const {

return \_fileName;

}

private:

// имя файла с картинкой

std::string \_fileName;

};

Image.cpp

#include "Image.h"

#include "Utils.h"

#include <SDL.h>

#include <SDL\_image.h>

#include <iostream>

Image::Image(SDL\_Surface\* screen, const char\* fileName)

: ScreenObject(screen), \_fileName(fileName)

{

surface(IMG\_Load(fileName));

if (surface()) {

logSDLError(std::cerr, "renderText");

}

}

Text.h

#pragma once

#include "ScreenObject.h"

#include <string>

struct \_TTF\_Font;

class Text : public ScreenObject

{

public:

Text(SDL\_Surface\* screen, const char\* fileName, int size, const SDL\_Color& color);

~Text();

std::string fileName() const {

return \_fileName;

}

SDL\_Color color() const {

return \_color;

}

void color(const SDL\_Color& value) {

\_color = value;

}

std::string text() const {

return \_text;

}

void text(const std::string& value);

private:

// объект шрифта

\_TTF\_Font\* \_font = nullptr;

// имя файла с шрифтом

std::string \_fileName;

// цвет шрифта

SDL\_Color \_color;

std::string \_text;

};

Text.cpp

#include "Text.h"

#include "Utils.h"

#include <SDL.h>

#include <SDL\_ttf.h>

#include <iostream>

Text::Text(SDL\_Surface\* screen, const char\* fileName, int size, const SDL\_Color& color)

: ScreenObject(screen), \_fileName(fileName), \_color(color)

{

\_font = TTF\_OpenFont(fileName, size);

if(!\_font) {

logSDLError(std::cerr, "TTF\_OpenFont");

}

}

Text::~Text()

{

if (\_font) {

TTF\_CloseFont(\_font);

}

}

void Text::text(const std::string& value) {

\_text = value;

surface(TTF\_RenderText\_Blended(\_font, \_text.c\_str(), \_color));

}

Cockroach.h

#pragma once

#include "ScreenObject.h"

class Cockroach : public ScreenObject

{

public:

enum class Orient : int {

up, down, left, right

};

Cockroach(int fieldW, int fieldH, SDL\_Surface\* screen, const char\* imageName, Orient orient, int d = 1);

~Cockroach() = default;

bool initGraphics(const char\* imageName);

// сделать следующий шаг (смена координат)

void move();

// вышел за пределы экрана?

bool away() const;

// увернулся от тапка?

bool evade(int x, int y) const;

Orient orient() const {

return \_orient;

}

void orient(Orient value) {

\_orient = value;

}

int d() const {

return \_d;

}

void d(int value) {

\_d = value;

}

private:

Orient \_orient = Orient::right;

int \_d = 0;

int \_fieldW = 0;

int \_fieldH = 0;

};

Cockroach.cpp

#include "Cockroach.h"

#include <SDL.h>

#include <SDL\_image.h>

#include <iostream>

// для генерации пседвослучайных чисел (ПСЧ)

#include <random>

#include <ctime>

Cockroach::Cockroach(int fieldW, int fieldH, SDL\_Surface\* screen, const char\* imageName, Orient orient, int d)

: ScreenObject(screen), \_orient(orient), \_d(d), \_fieldW(fieldW), \_fieldH(fieldH)

{

initGraphics(imageName);

}

bool Cockroach::initGraphics(const char\* imageName) {

if (!\_screen || !imageName) {

return false;

}

\_surface = IMG\_Load(imageName);

if (\_surface) {

\_pos.w = \_surface->clip\_rect.w;

\_pos.h = \_surface->clip\_rect.h;

std::mt19937 gen;

gen.seed(static\_cast<uint32\_t>(time(0)));

switch (\_orient) {

case Orient::up:

{

std::uniform\_int\_distribution<> uidLoc(4, \_fieldW - 5);

\_pos.x = uidLoc(gen);

\_pos.y = \_fieldH;

\_d = -\_d;

\_pos.x -= \_pos.w / 2;

}

break;

case Orient::down:

{

std::uniform\_int\_distribution<> uidLoc(4, \_fieldW - 5);

\_pos.x = uidLoc(gen);

\_pos.y = -\_pos.h;

\_pos.x -= \_pos.w / 2;

}

break;

case Orient::left:

{

std::uniform\_int\_distribution<> uidLoc(4, \_fieldH - 5);

\_pos.x = \_fieldW;

\_pos.y = uidLoc(gen);

\_d = -\_d;

\_pos.y -= \_pos.h / 2;

}

break;

case Orient::right:

{

std::uniform\_int\_distribution<> uidLoc(4, \_fieldH - 5);

\_pos.x = -\_pos.w;

\_pos.y = uidLoc(gen);

\_pos.y -= \_pos.h / 2;

break;

}

}

return true;

}

return false;

}

void Cockroach::move()

{

int& pos = (Orient::up == \_orient || Orient::down == \_orient) ? \_pos.y : \_pos.x;

pos += \_d;

}

bool Cockroach::away() const

{

switch (\_orient) {

case Orient::up:

return (\_pos.y < 0);

case Orient::down:

return (\_pos.y > \_fieldH);

case Orient::left:

return (\_pos.x < 0);

case Orient::right:

return (\_pos.x > \_fieldW);

}

return false;

}

bool Cockroach::evade(int x, int y) const

{

int x0 = \_pos.x + 10;

int y0 = \_pos.y + 10;

int x1 = \_pos.x + \_pos.w - 10;

int y1 = \_pos.y + \_pos.h - 10;

return !(x > x0 && y > y0 && x < x1 && y < y1);

}

MenuItem.h

#pragma once

#include <vector>

#include <string>

#include "ScreenObject.h"

struct \_TTF\_Font;

class MenuItem : public ScreenObject

{

public:

MenuItem(SDL\_Surface\* screen, const std::string& label);

~MenuItem();

void verify(int x, int y, \_TTF\_Font\* font, const SDL\_Color& clrOn, const SDL\_Color& clrOff);

std::string label() const {

return \_label;

}

bool selected() const {

return \_selected;

}

void select(bool value) {

\_selected = value;

}

private:

std::string \_label = "";

bool \_selected = false;

};

MenuItem.cpp

#include "MenuItem.h"

#include <SDL.h>

#include <SDL\_ttf.h>

MenuItem::MenuItem(SDL\_Surface\* screen, const std::string& label)

: ScreenObject(screen), \_label(label), \_selected(false)

{

}

MenuItem::~MenuItem()

{

clear();

}

void MenuItem::verify(int x, int y, \_TTF\_Font\* font, const SDL\_Color& clrOn, const SDL\_Color& clrOff) {

if (inside(x, y)) {

if (!selected()) {

select(true);

surface(TTF\_RenderText\_Blended(font, \_label.c\_str(), clrOn));

}

}

else {

if (selected()) {

select(false);

surface(TTF\_RenderText\_Blended(font, \_label.c\_str(), clrOff));

}

}

}

Menu.h

#pragma once

#include <vector>

#include <string>

struct SDL\_Window;

struct SDL\_Surface;

struct SDL\_Color;

struct \_TTF\_Font;

class MenuItem;

class Menu

{

public:

using Labels = std::vector<std::string>;

using Items = std::vector<MenuItem\*>;

Menu(SDL\_Window\* window);

~Menu();

int show(SDL\_Surface\* screen, \_TTF\_Font\* font, const Labels& labels);

void clear(Items& items);

private:

SDL\_Window\* \_window = nullptr;

};

Menu.cpp

#include "Menu.h"

#include "MenuItem.h"

#include <iostream>

#include <sstream>

#include <fstream>

#include <SDL.h>

#include <SDL\_ttf.h>

Menu::Menu(SDL\_Window\* window)

: \_window(window)

{

}

Menu::~Menu()

{

}

int Menu::show(SDL\_Surface\* screen, \_TTF\_Font\* font, const Labels& labels)

{

Items items;

if (labels.empty()) {

return -1;

}

items.reserve(labels.size());

for (const auto& label : labels) {

items.push\_back(new MenuItem(screen, label));

}

items[0]->select(true);

// normal, selected

SDL\_Color color[2] = { { 255, 255, 255 }, { 255, 0, 0 } };

uint32\_t time;

int height = 0;

for (int i = 0; i < items.size(); ++i) {

items[i]->surface(TTF\_RenderText\_Blended(font, items[i]->label().c\_str(), color[i == 0 ? 1 : 0]));

height += items[i]->surface()->clip\_rect.h;

}

int top = (screen->clip\_rect.h - height) / 2;

int h = items[0]->surface()->clip\_rect.h;

int left = (screen->clip\_rect.w - items[0]->surface()->clip\_rect.w) / 2;

for (int i = 0; i < items.size(); ++i) {

items[i]->locate(left, top);

top += h;

}

SDL\_FillRect(screen, &screen->clip\_rect, SDL\_MapRGB(screen->format, 0x00, 0x00, 0x00));

SDL\_Event event;

while (true) {

time = SDL\_GetTicks();

while (SDL\_PollEvent(&event)) {

switch (event.type) {

case SDL\_QUIT:

clear(items);

return -1;

case SDL\_MOUSEMOTION:

for (auto item : items) {

item->verify(event.motion.x, event.motion.y, font, color[1], color[0]);

}

break;

case SDL\_MOUSEBUTTONDOWN:

for (size\_t i = 0; i < items.size(); ++i) {

if (items[i]->inside(event.button.x, event.button.y)) {

clear(items);

return static\_cast<int>(i);

}

}

break;

case SDL\_KEYDOWN:

switch (event.key.keysym.sym) {

case SDLK\_ESCAPE:

clear(items);

return -1;

}

break;

}

}

for (int i = 0; i < items.size(); ++i) {

items[i]->draw();

}

SDL\_UpdateWindowSurface(\_window);

//SDL\_Flip(screen);

SDL\_Delay(20);

} // while

clear(items);

return 0;

}

void Menu::clear(Items& items) {

for (MenuItem\* item : items) {

item->clear();

delete item;

}

items.clear();

}

Window.h

#pragma once

#include <string>

#include <list>

struct SDL\_Window;

struct SDL\_Surface;

struct SDL\_Color;

struct \_TTF\_Font;

class ScreenObject;

class Window

{

public:

Window(int width, int height, const std::string& title = "");

virtual ~Window();

void stop();

protected:

virtual int show() = 0;

protected:

using Objects = std::list<ScreenObject\*>;

// графические объекты

SDL\_Window\* \_window = nullptr;

SDL\_Surface\* \_screen = nullptr;

// список объектов на экране

Objects objects;

// размеры окна

int \_scrWidth, \_scrHeight;

// флаг активности цикла обрабтки событий

bool \_active;

// заголовок окна

const std::string \_title;

};

Window.cpp

#include "Window.h"

#include "Utils.h"

#include "ScreenObject.h"

#include <SDL.h>

#include <SDL\_image.h>

#include <SDL\_ttf.h>

#include <iostream>

using namespace std;

Window::Window(int width, int height, const string& title)

: \_active(true), \_title(title)

{

\_scrWidth = width;

\_scrHeight = height;

\_window = SDL\_CreateWindow(title.c\_str(), SDL\_WINDOWPOS\_CENTERED,

SDL\_WINDOWPOS\_CENTERED, \_scrWidth, \_scrHeight, SDL\_WINDOW\_SHOWN);

if (\_window) {

\_screen = SDL\_GetWindowSurface(\_window);

}

}

Window::~Window()

{

SDL\_DestroyWindow(\_window);

\_window = nullptr;

\_screen = nullptr;

}

void Window::stop()

{

\_active = false;

}

InputWindow.h

#pragma once

#include "Window.h"

struct SDL\_Window;

struct SDL\_Surface;

struct SDL\_Color;

class InputWindow: public Window

{

public:

InputWindow(int width, int height, const std::string& title);

~InputWindow() = default;

int show() override;

bool confirmed() const {

return \_confirmed;

}

std::string input() const {

return \_input;

}

private:

// признак подтверждения

bool \_confirmed = false;

// введённая строка

std::string \_input = "";

};

InputWindow.cpp

#include "InputWindow.h"

#include "Utils.h"

#include "Text.h"

#include <SDL.h>

#include <SDL\_image.h>

#include <SDL\_ttf.h>

#include <iostream>

InputWindow::InputWindow(int width, int height, const std::string& title)

: Window(width, height, title)

{

}

int InputWindow::show()

{

SDL\_FillRect(\_screen, &\_screen->clip\_rect, SDL\_MapRGB(\_screen->format, 50, 50, 50));

Text \*txtMain = new Text(\_screen, "res/arial.ttf", 20, { 25, 150, 150, 255 });

if (!txtMain) {

return 3;

}

txtMain->locate(20, \_scrHeight / 3 + 10);

objects.push\_back(txtMain);

// событие SDL

SDL\_Event evt;

// время одного шага - 50 мс

constexpr uint32\_t stepDelay = 50; // ms

// цикл обработки событий

while (\_active) {

SDL\_Delay(stepDelay);

// Обработка событий пользователя

while (SDL\_PollEvent(&evt)) {

switch (evt.type) {

case SDL\_QUIT:

std::cout << "SDL\_QUIT" << std::endl;

stop();

break;

case SDL\_TEXTINPUT:

\_input += evt.text.text;

break;

case SDL\_KEYDOWN:

switch (evt.key.keysym.scancode) {

case SDL\_SCANCODE\_ESCAPE:

\_confirmed = false;

stop();

break;

case SDL\_SCANCODE\_RETURN:

\_confirmed = true;

stop();

break;

}

break;

case SDL\_MOUSEBUTTONDOWN:

stop();

break;

} // case

} // while

// Текст

txtMain->text("User name: " + \_input);

// Отрисовка сцены

for (auto it = begin(objects); it != end(objects); it++) {

(\*it)->draw();

}

// показ сцены в окне

SDL\_UpdateWindowSurface(\_window);

} // while(active)

// Очистка ресурсов

for (auto it = begin(objects); it != end(objects); it++) {

delete (\*it);

}

return 0;

}

StartScreen.h

#pragma once

#include "Window.h"

struct SDL\_Window;

struct SDL\_Surface;

struct SDL\_Color;

class StartScreen : public Window

{

public:

StartScreen(int width, int height, const std::string& title);

~StartScreen() = default;

int show() override;

};

StartScreen.cpp

#include "StartScreen.h"

#include "Utils.h"

#include "Image.h"

#include "Text.h"

#include <SDL.h>

#include <SDL\_image.h>

#include <SDL\_ttf.h>

#include <iostream>

StartScreen::StartScreen(int width, int height, const std::string& title)

: Window(width, height, title)

{

}

int StartScreen::show()

{

// текстовые объекты

Text \*txtMain = new Text(\_screen, "res/arial.ttf", 40, { 50, 50, 200, 255 });

Text \*txtAuthor = new Text(\_screen, "res/arial.ttf", 18, { 30, 30, 200, 255 });

if (!txtMain || !txtAuthor) {

return 3;

}

txtMain->locate(20, \_scrHeight / 3 + 10);

txtMain->text(\_title);

txtAuthor->locate(20, \_scrHeight / 2 + 10);

txtAuthor->text("Author: \_\_\_\_\_\_\_\_ Group: \_\_\_\_\_\_\_\_\_\_\_");

// загружаем фон

objects.push\_back(new Image(\_screen, "res/init.jpg"));

objects.push\_back(txtMain);

objects.push\_back(txtAuthor);

// событие SDL

SDL\_Event evt;

// время одного шага - 50 мс

constexpr uint32\_t stepDelay = 50; // ms

// цикл обработки событий

while (\_active) {

SDL\_Delay(stepDelay);

// Обработка событий пользователя

while (SDL\_PollEvent(&evt)) {

switch (evt.type) {

case SDL\_QUIT:

stop();

break;

case SDL\_KEYDOWN:

switch (evt.key.keysym.scancode) {

case SDL\_SCANCODE\_ESCAPE:

stop();

break;

}

break;

case SDL\_MOUSEBUTTONDOWN:

stop();

break;

} // case

} // while

// Отрисовка сцены

for (auto it = begin(objects); it != end(objects); it++) {

(\*it)->draw();

}

// показ сцены в окне

SDL\_UpdateWindowSurface(\_window);

} // while(active)

// Очистка ресурсов

for (auto it = begin(objects); it != end(objects); it++) {

delete (\*it);

}

return 0;

}