Балтийский государственный технический университет «ВОЕНМЕХ»
им. Д. Ф. Устинова

Кафедра И5
«Информационные системы и программная инженерия»

Практическое задание № 1
по дисциплине «Программирование на ЯВУ»
на тему «Подключение SDL 2.0»

Выполнил:
Студент Дубровский В.И.
Группа И582

Преподаватель:
Спирин Д.О.

Санкт-Петербург
2019

Цель работы – изучить ключевые функции и понятия SDL2.

Задание.

Подключить SDL 2.0 и вывести на экран любимую геометрическую фигуру, а также текст с ФИО автора и версией SDL. Версию SDL получать при помощи SDL_GetVersion(). Работа выполняется на языке C++. Программа обязательно должна содержать пользовательские класс(ы). Функия main обязательно должна содержать только одну строчку пользовательского кода. Все сообщения на русском языке.

Описание классов:

Класс Application ответственен за инициализацию окна, рендера и основного цикла приложения.

Класс SDLWidget базовый класс виджетов.

Класс Label ответственен за работу с текстом.

Класс RectRender за прорисовку квадрата.

Текст программы:

Application.h

```
#pragma once
#include<SDL.h>
#include<memory>
#include<vector>

class Application
{
private:
	bool _isRunning = false;
	SDL_Window* _window;
	SDL_Renderer* _renderer;
	SDL_Event _events;

protected:
	void init();
	void render();
	void eventUpdate();
	void destroy();
public:

	Application(const char* title, int w, int h);
	~Application();
	int run();
	SDL_Renderer* GetRender();
};
```

Application.cpp

```
#include "Application.h"
#include <iostream>
#include <SDL_ttf.h>
#include "Widgets.h"
#include <exception>

RectRender* rect;
Label* versionLabel;
Label* nameLabel;
```

```
Application::Application(const char* title, int w, int h)
{
	if(SDL_Init(SDL_INIT_EVERYTHING) == -1) throw std::exception(SDL_GetError());
	if (TTF_Init() == -1) throw std::exception(TTF_GetError());

	_window = SDL_CreateWindow(title, SDL_WINDOWPOS_CENTERED, SDL_WINDOWPOS_CENTERED,
w, h, SDL_WINDOW_SHOWN);
	if (_window) std::cerr << "Ok Window" << std::endl;
	else throw std::exception(SDL_GetError());

	_renderer = SDL_CreateRenderer(_window, -1, NULL);
	if (_renderer) std::cerr << "Ok Renderer" << std::endl;
	else throw std::exception(SDL_GetError());


}

Application::~Application()
{
	SDL_DestroyRenderer(_renderer);
	SDL_DestroyWindow(_window);
	SDL_Quit();
	TTF_Quit();
}


SDL_Renderer* Application::GetRender()
{
	return _renderer;
}

void Application::init()
{
	_isRunning = true;
	rect = new RectRender(this);
	versionLabel = new Label(this);
	nameLabel = new Label(this);


	versionLabel->SetFont("myfont.ttf", 30);
	SDL_version linked;
	SDL_GetVersion(&linked);
	std::string linkedText = "SDL: " + std::to_string(linked.major) + '.' +
std::to_string(linked.minor)+ '.' + std::to_string(linked.patch);

	versionLabel->SetText(linkedText, { 255,255,0 });
	versionLabel->SetPosition(0, 160);

	nameLabel->SetFont("myfont.ttf", 20);
	nameLabel->SetText("Дубровский Владислав.", {0, 255, 0} );
	nameLabel->SetPosition(0, 300);


	rect->SetRect(200, 50, 120, 120);
}


void Application::render()
{
	SDL_SetRenderDrawColor(_renderer, 0, 0, 0, SDL_ALPHA_OPAQUE);
	SDL_RenderClear(_renderer);

	rect->render();
```

```
	nameLabel->render();
	versionLabel->render();

	SDL_RenderPresent(_renderer);
}

void Application::eventUpdate()
{
	SDL_WaitEvent(&_events);
	switch (_events.type)
	{
		case SDL_QUIT:
			_isRunning = false;
			break;
		default:
			break;
	};
}

void Application::destroy()
{
	_isRunning = false;
	delete versionLabel;
	delete nameLabel;
	delete rect;
}
int Application::run()
{
	try {
		init();
		render();
		while (_isRunning)
		{
			eventUpdate();
		}
		destroy();

		return 0;
	}
	catch (std::exception& e)
	{
		destroy();
		std::cerr << e.what() << std::endl;
		return 1;
	}

}
```

SDLWidget.h:

```
#include <vector>
#include <memory>
#include "Application.h"
class SDLWidget
{

protected:
	Application* _parent;

public:
	SDLWidget(Application* parent) : _parent(parent) {};
	virtual ~SDLWidget() {};
	virtual void update() = 0;
	virtual void render() = 0;
```

```
};
```

Widgets.h:

```
#pragma once
#include "SDLWidget.h"
#include "CubeRender.h"
#include "Label.h"
```

Label.h :

```
#pragma once
#include "SDLWidget.h"
#include <string>
#include <SDL_ttf.h>
#include <SDL.h>
#include <iostream>


class Label : SDLWidget
{
private:
	TTF_Font* _font;
	SDL_Texture* _texture;
	SDL_Rect _destinationRect;
	std::string _text;

public:
	Label(Application* app) : SDLWidget(app)
	{
	}
	~Label()
	{
		SDL_DestroyTexture(_texture);
		TTF_CloseFont(_font);
	}

	void render() override
	{
		if(SDL_RenderCopy(_parent->GetRender(), _texture, nullptr,
&_destinationRect) == -1) std::cerr << SDL_GetError() << std::endl;
	}
	void update() override
	{

	}
	void SetText(std::string text, SDL_Color color)
	{

		SDL_DestroyTexture(_texture);
		_text = text;
		SDL_Surface* temp = TTF_RenderUTF8_Solid(_font, _text.c_str(), color);
		_texture = SDL_CreateTextureFromSurface(_parent->GetRender(), temp);
		SDL_FreeSurface(temp);
		SDL_QueryTexture(_texture, nullptr, nullptr, &_destinationRect.w,
&_destinationRect.h);


	}
	void SetFont(std::string path, int size)
```

```
	{
		_font = TTF_OpenFont(path.c_str(), size);
		if (_font == nullptr)
		{
			std::cerr << TTF_GetError() << std::endl;
			_parent->SetState(ErrorLog::BAD_FONT);
		}
	}
	void SetPosition(int x, int y)
	{
		_destinationRect.x = x;
		_destinationRect.y = y;

	}
};
```

RectRender.h :

```
#pragma once
#include "SDLWidget.h"
class RectRender : SDLWidget
{
private:
	SDL_Rect _rect;
public:
	RectRender(Application* app) : SDLWidget(app)
	{
		SetRect(0, 0, 0, 0);
	}
	void SetRect(int x, int y, int w, int h)
	{
		_rect.x = x;
		_rect.y = y;
		_rect.w = w;
		_rect.h = h;
	}
	void SetColor(int r, int g, int b)
	{
		SDL_SetRenderDrawColor(_parent->GetRender(), r, g, b,SDL_ALPHA_OPAQUE);
	}
	void update() override
	{

	}
	void render() override
	{
		SetColor(255, 0, 0);
		SDL_RenderFillRect(_parent->GetRender(), &_rect);
	}
};
```

main.cpp :

```
#include "Application.h"

constexpr auto TITLE = "Laboratornay rabota 1";
constexpr auto WIDTH = 500;
constexpr auto HEIGHT = 500;

Application apl(TITLE, WIDTH, HEIGHT);

int main(int argc, char* argv[])
{
	return apl.run();
}
```

Результат работы программы:

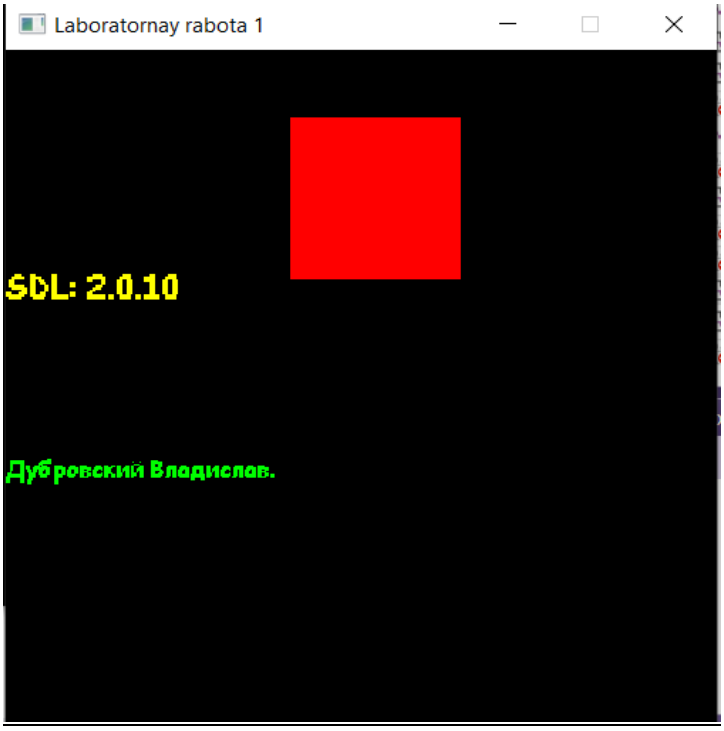

Рисунок 1 – Результат работы программы