Введение в SDL

Николай Белый

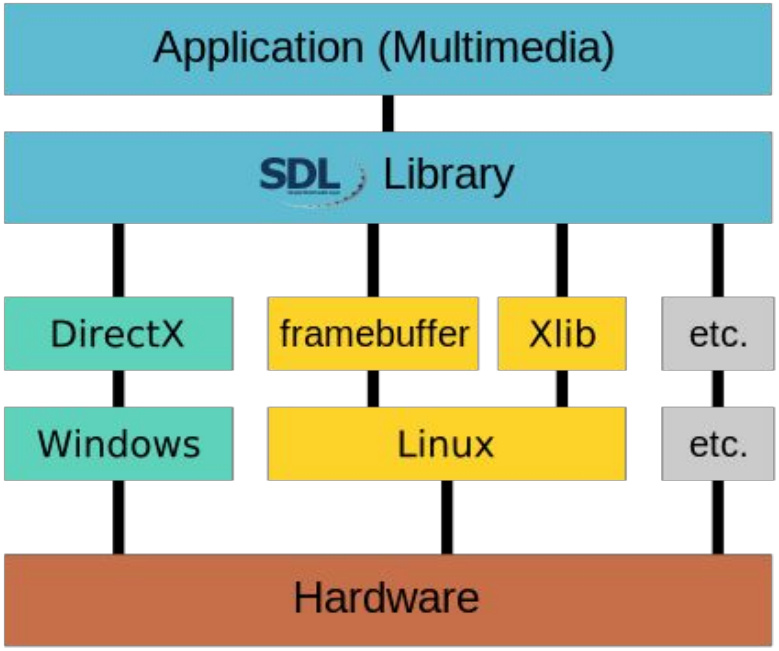
Никита Одиноких

О библиотеке

- SDL (Simple DirectMedia Layer) - это кроссплатформенная библиотека, обеспечивающая низкоуровневый доступ к звуку, клавиатуре, мыши, геймпаду и графике (OpenGL и Direct3D). Она используется в плеерах, эмуляторах, и популярных играх, в том числе игры от Valve и Humble Bundle.
- SDL официально поддерживает Windows, Mac OS X, Linux и Android. Неофициально поддерживаются и

Архитектура SDL

- SDL сам по себе довольно прост. Его можно рассматривать как тонкую прослойку, обеспечивающую поддержку для 2D-операций над пикселами, звука, доступа к файлам, обработки событий и т. п.



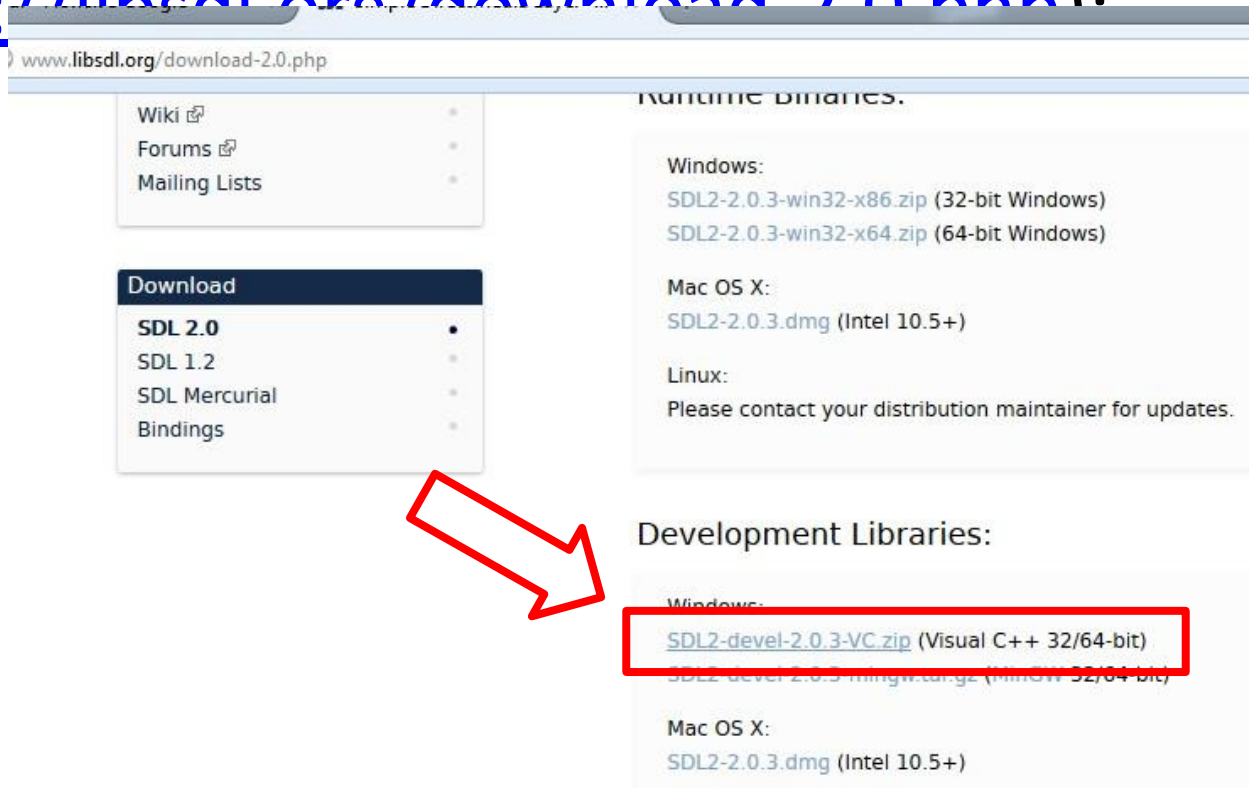
- Библиотека состоит из нескольких подсистем, таких как Video, Audio, CD-ROM, Joystick и Timer. В дополнение к этой базовой низкоуровневой функциональности, существует ряд стандартных библиотек, предоставляющих дополнительную

Подключение SDL

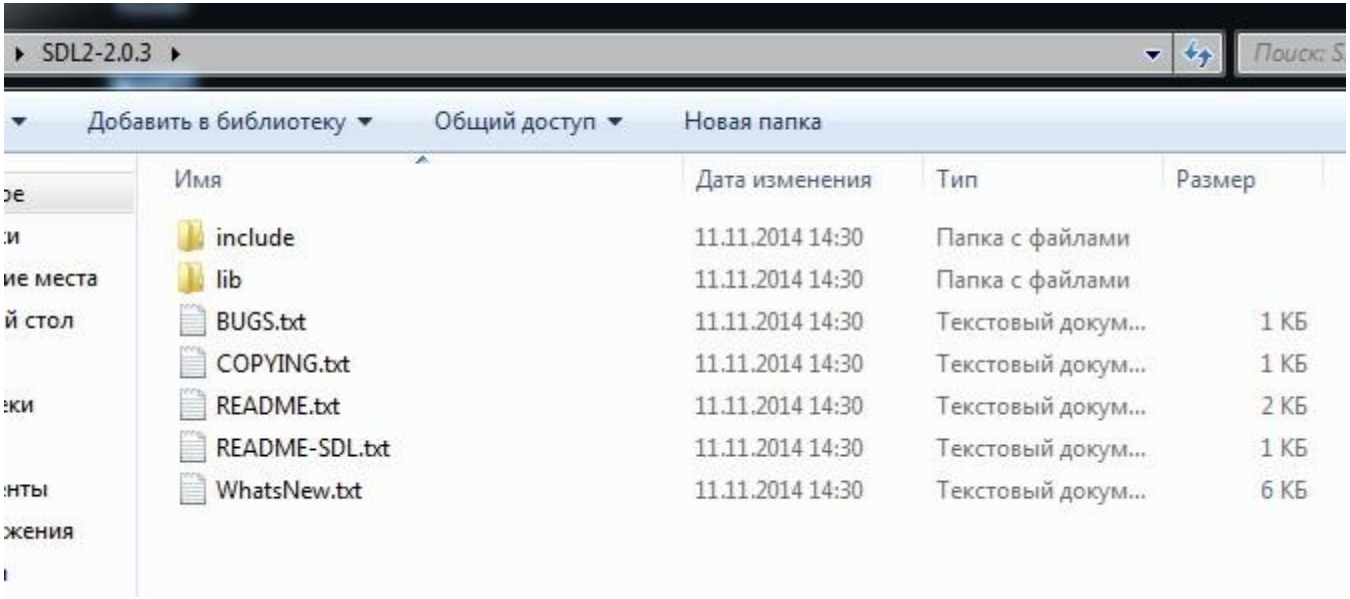
- SDL не входит в стандартную библиотеку C++. Для использования библиотеки в своих проектах необходимо скачать ее с официального сайта и подключить к своему проекту. Для каждого нового проекта библиотеку необходимо подключать заново.
- Хоть процесс подключения и выглядит громоздким, но на самом деле все очень просто.

Порядок подключения:

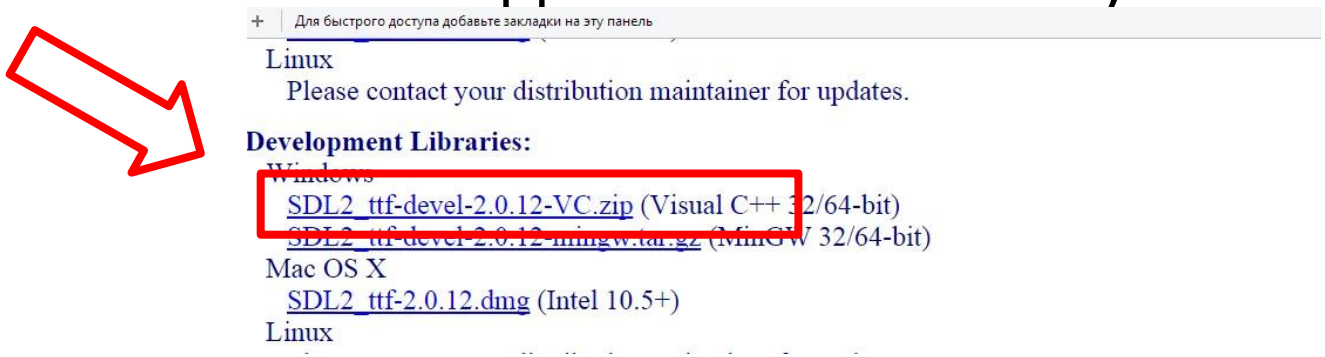
1. Скачиваем библиотеку с официального сайта (http://libsdl.org/download-2.0.php):



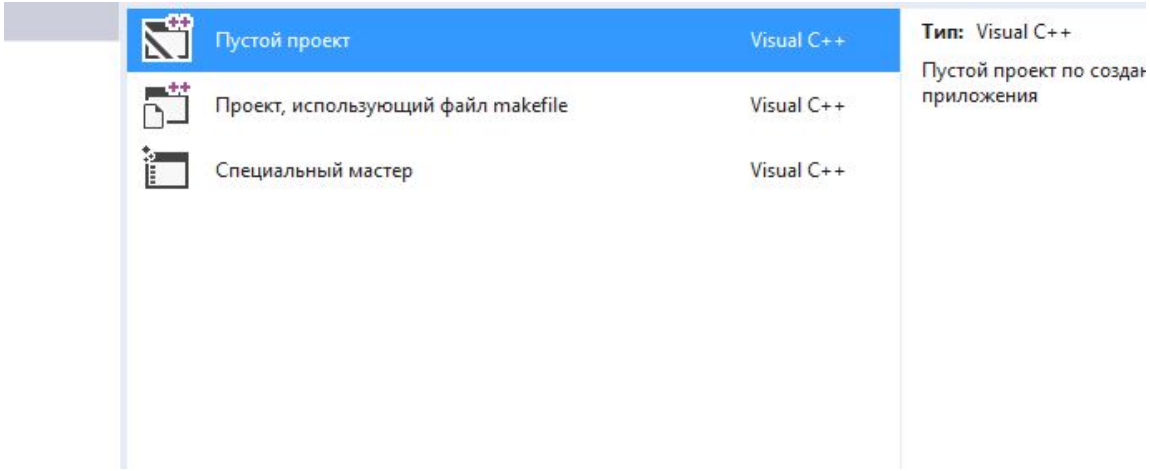
2. Распаковываем SDL. Важно помнить, что после подключения библиотеки к проекту папка с SDL не должна меняться.



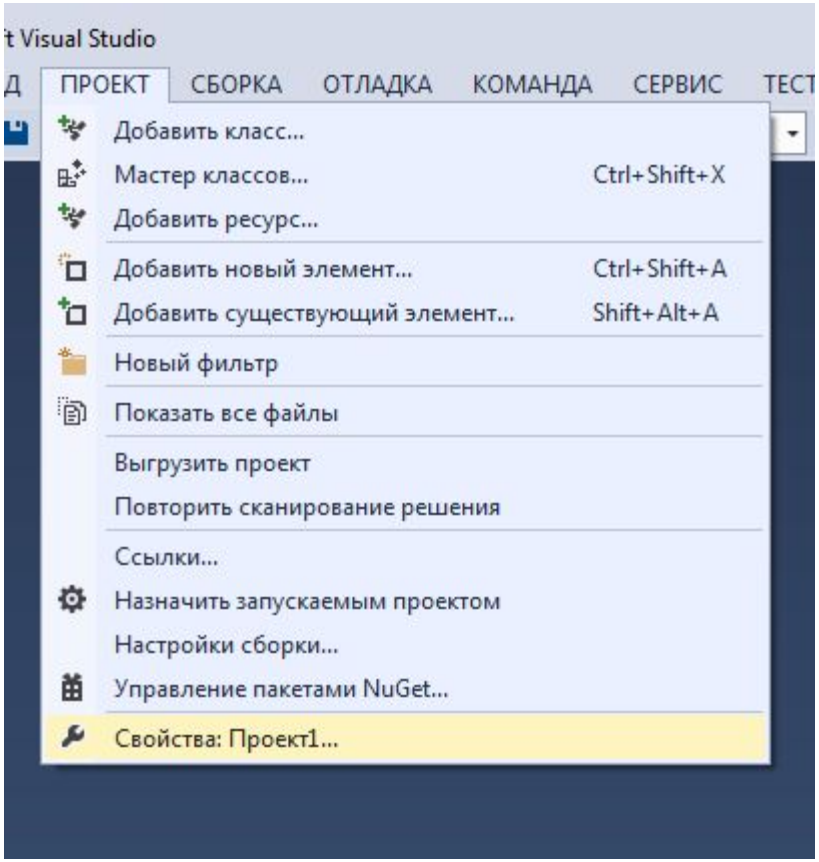
3. SDL не умеет рисовать текст.Но, существует плагин, позволяющий использовать TrueType шрифты в приложениях. Скачиваем его с сайта (https://www.libsdl.org/projects/SDL_ttf/) и распаковываем содержимое архива в папку с SDL (папки include и lib должны совпасть)



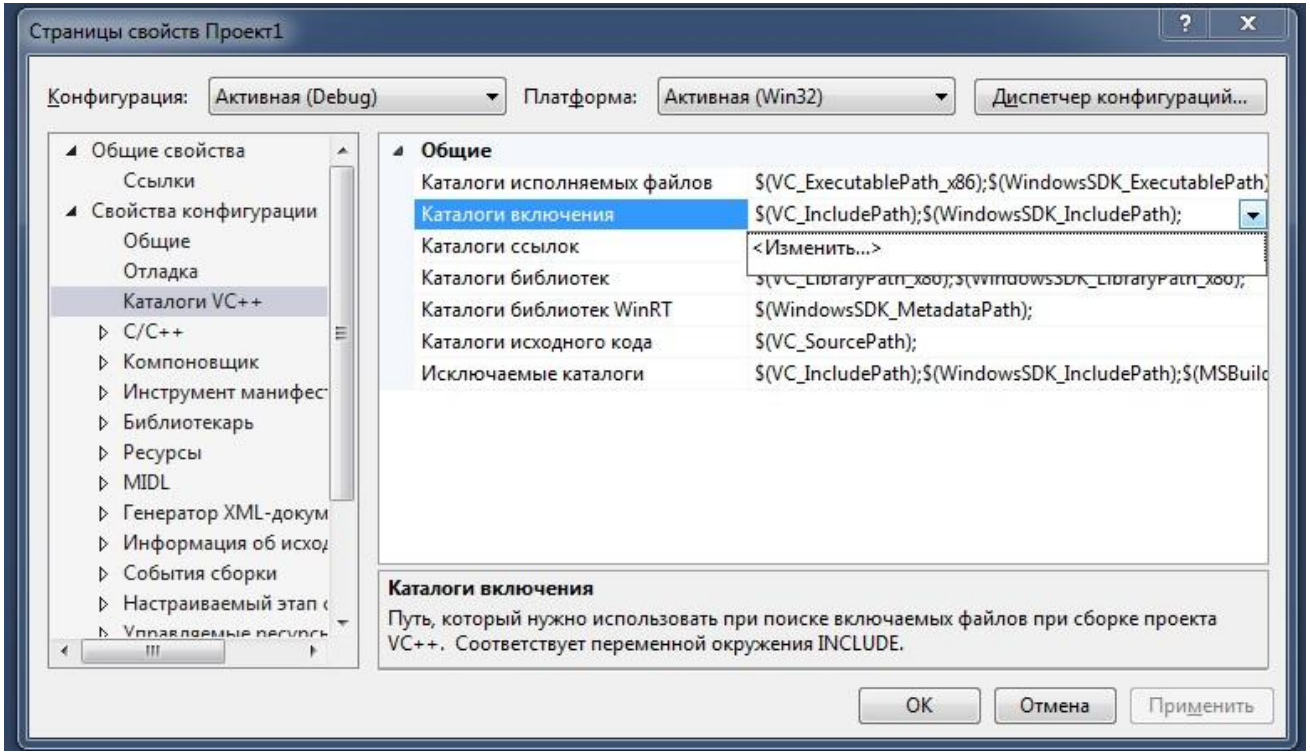
4. Теперь открываем Visual Studio и создаем пустой проект:



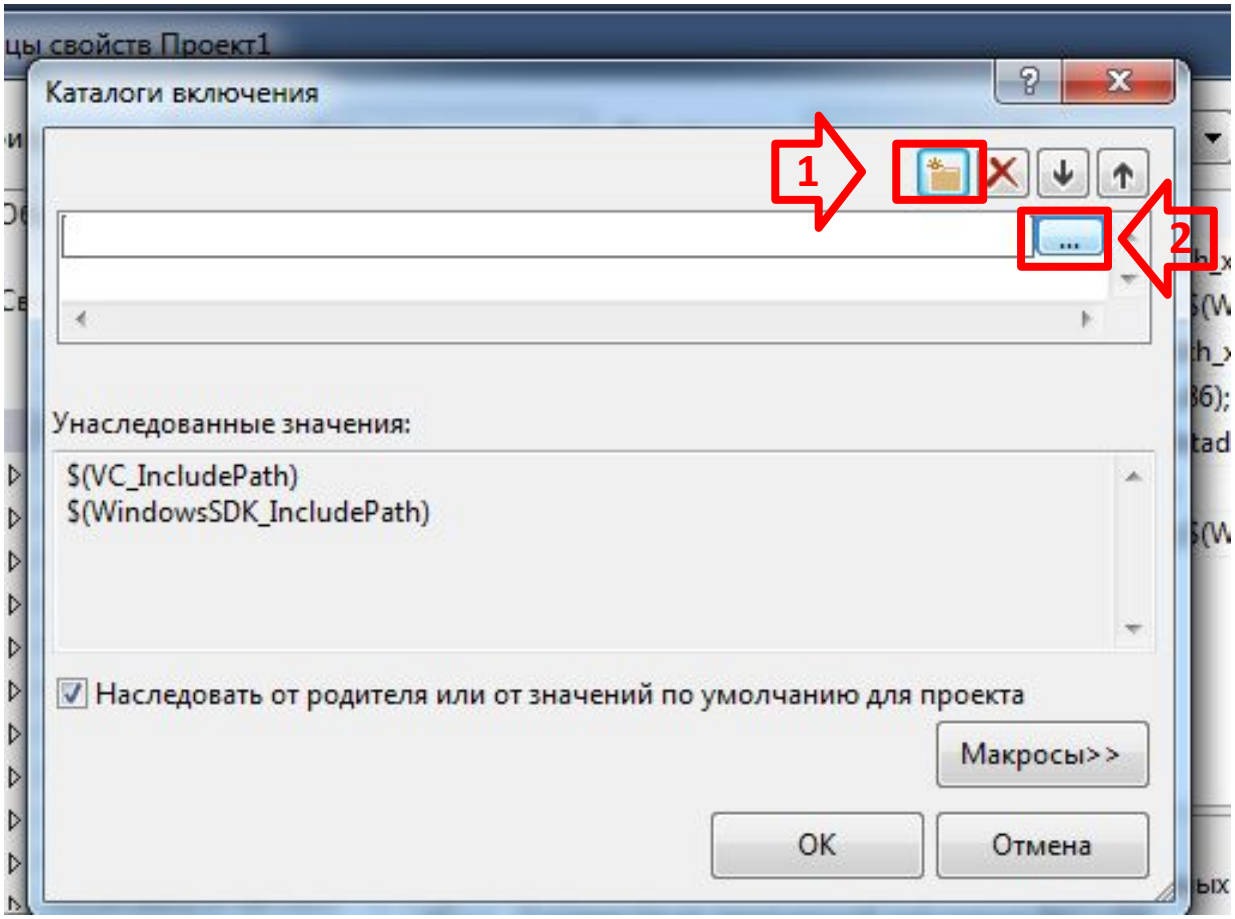
5. Заходим в свойства проекта:



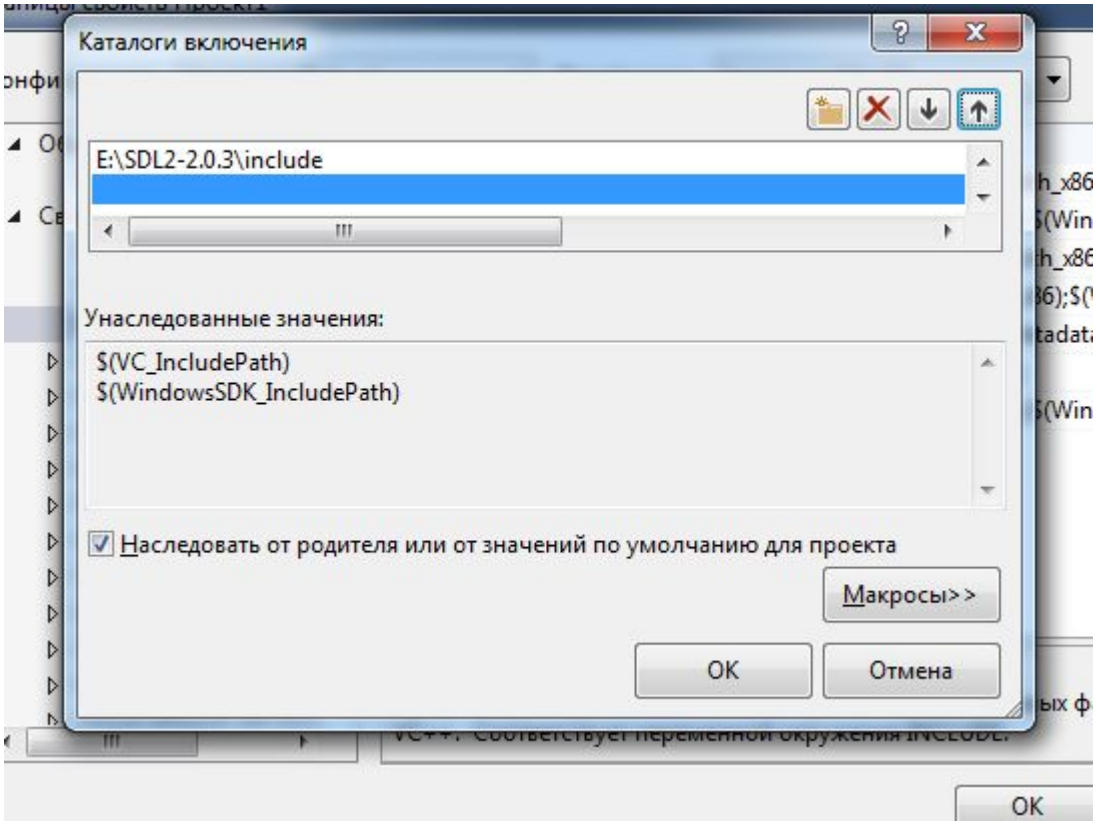
6. Свойства конфигурации -> Каталоги VC++. Теперь выбираем «Каталоги включения». Нажимаем «Изменить»



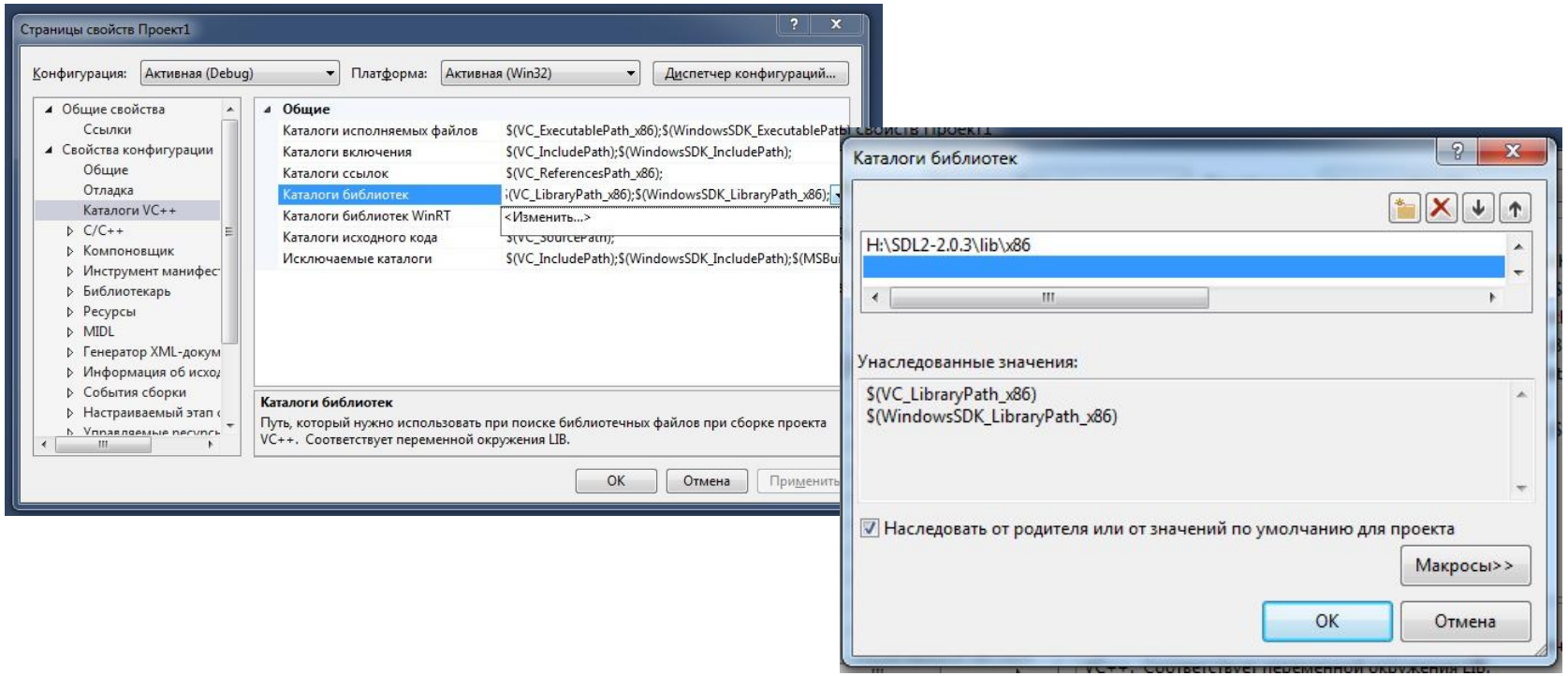
7.


цы свойств Проект1
Каталоги включения
1
2
...
Унаследованные значения:
$(VC_IncludePath)
$(WindowsSDK_IncludePath)
Наследовать от родителя или от значений по умолчанию для проекта
Макросы>>
OK
Отмена

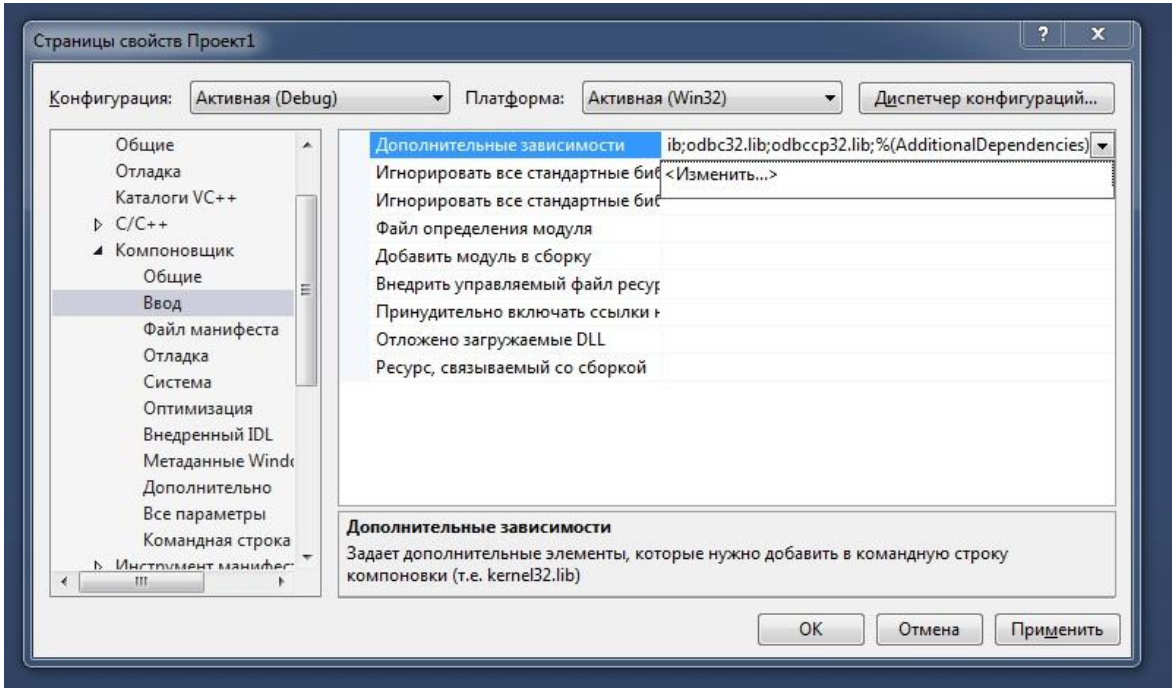
8. Теперь переходим в папку с SDL и выбираем папку «include



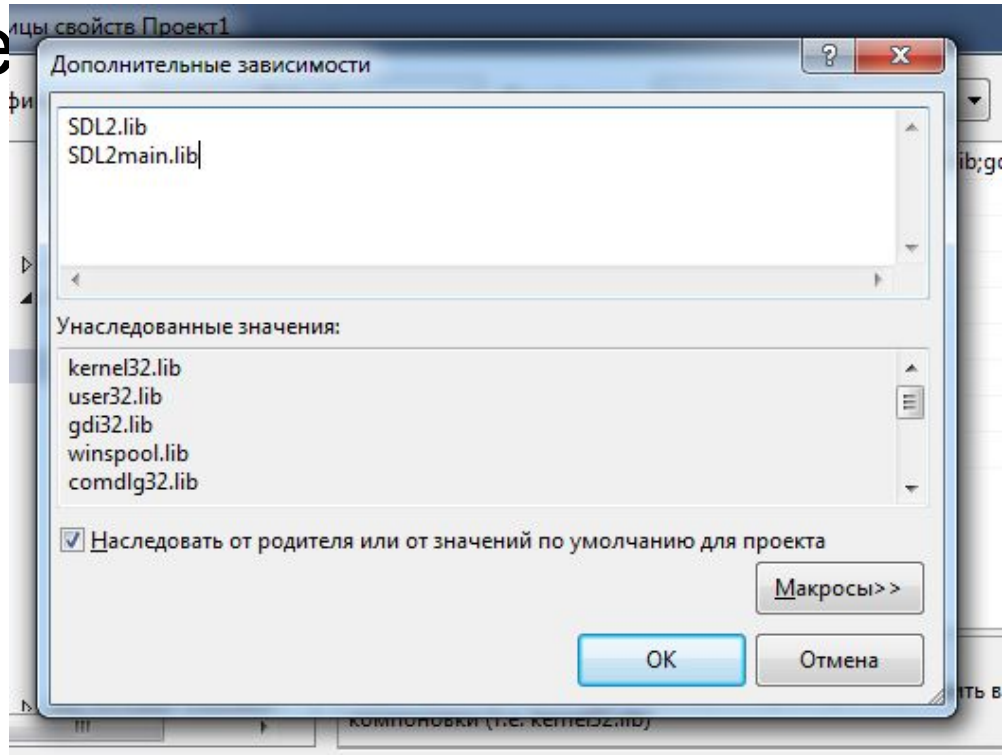
9. Нажимаем «ОК». Выбираем «Каталоги библиотек» и аналогично добавляем папку «li



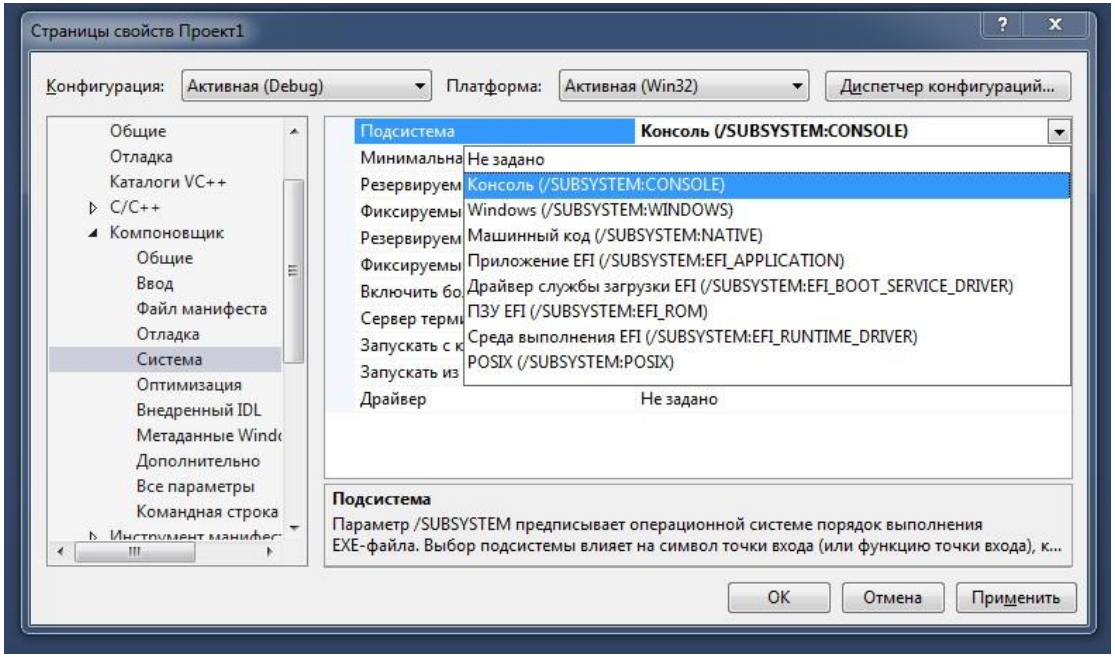
10. Теперь переходим Компоновщик -> Ввод и выбираем «Дополнительные зависимости»:



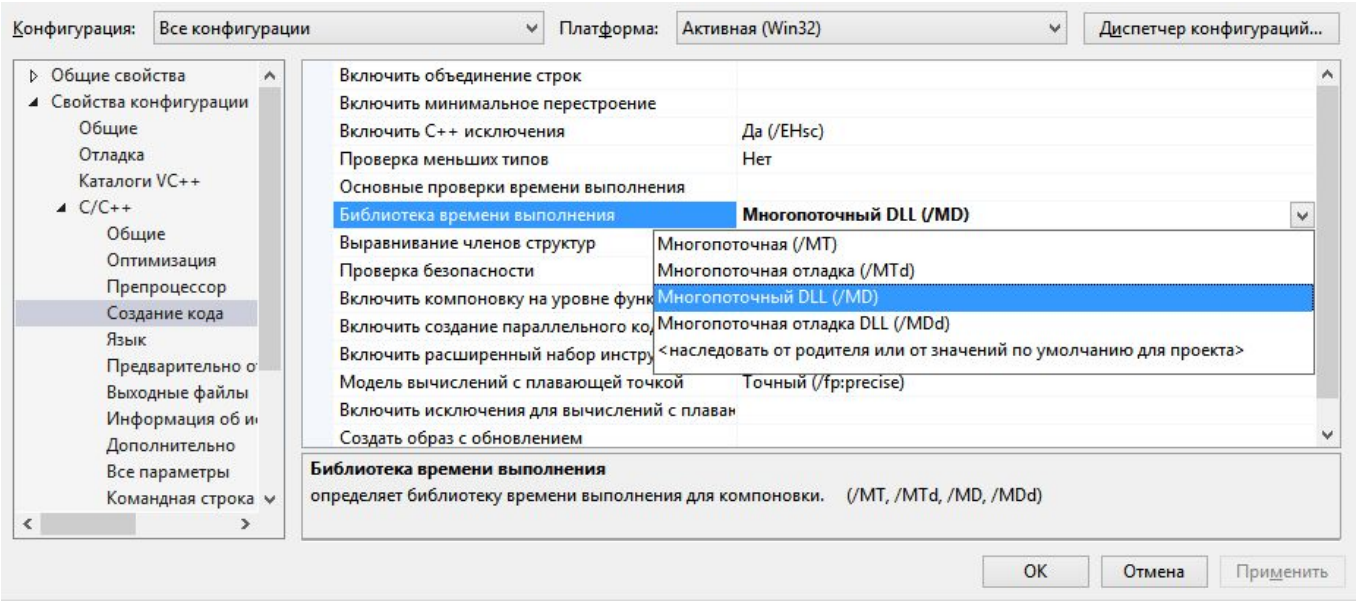
11. Вводим «SDL2.lib; SDL2main.lib;SDL2_ttf.lib;» и нажимае



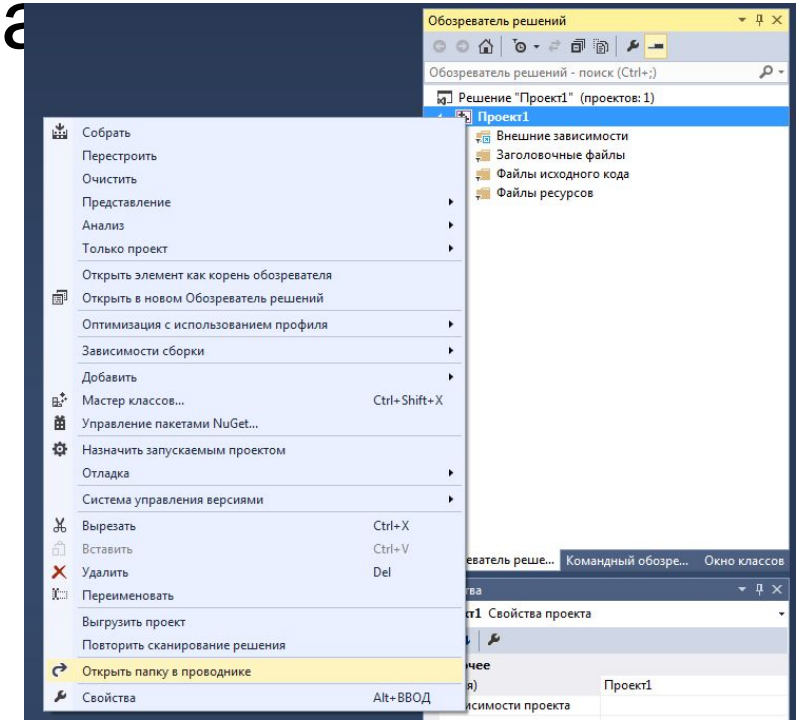
12. Переходим в «Компоновщик -> Система». Выбираем «Подсистема -> Консоль (/SUBSYSTEM:CONSOLE)».



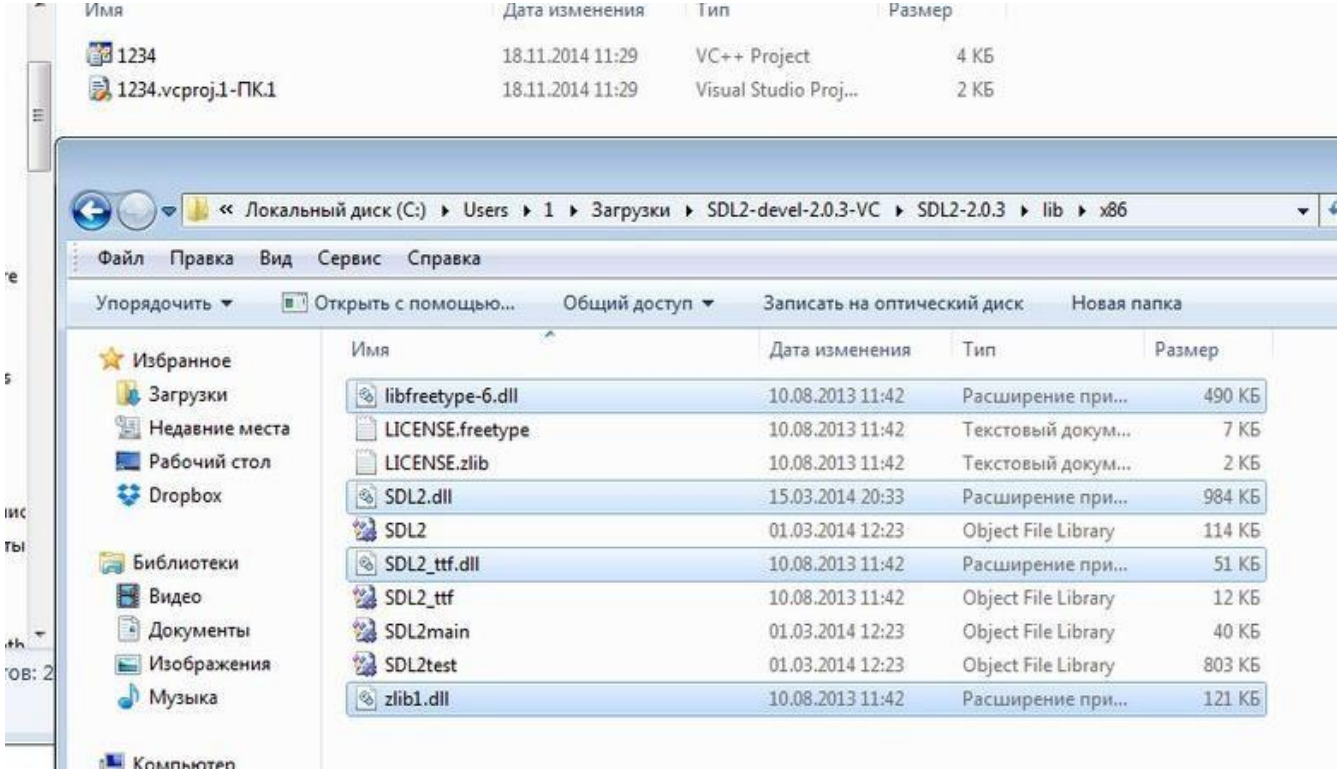
13. Переходим в «C/C++ -> Создание кода -> Библиотека времени выполнения». Выбираем «Многопоточный DLL (/MD)». Нажимаем «ОК»



14. Теперь в обозревателе решений щелкаем ПКМ по нашему проекту и выбираем «Открыть па

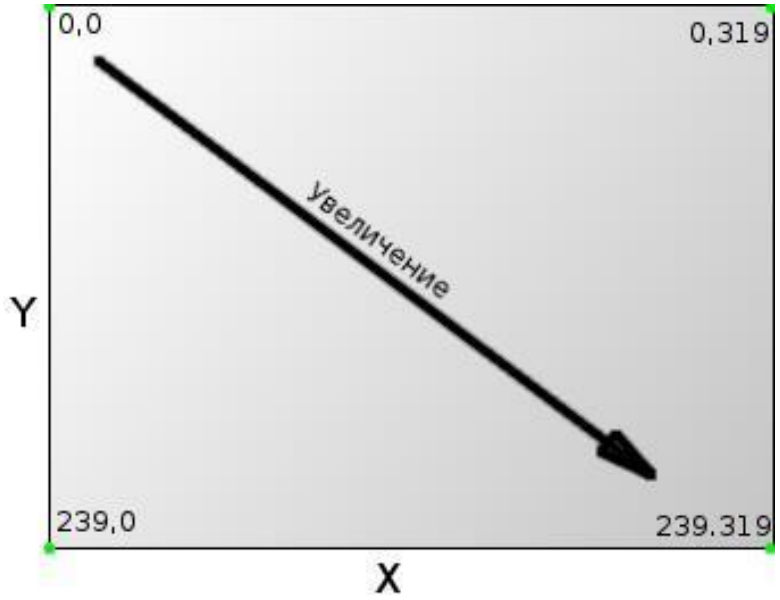


15. В открывшеюся папку копируем все *.dll файлы из папки /lib/x86.



Немного теории

Прежде всего, необходимо понять, как происходит отсчет координат в SDL. В отличие от привычной нам декартовой системы, здесь отсчет начинается с левого верхнего угла.



- Для создания окна необходимо объявить переменную типа SDL_Window. Напрямую в окне нельзя ничего отобразить. Однако, мы можем связать наше окно с некоторой структурой данных, содержимое которой и будет отображать наше окно.
- За возможность рисования геометрии отвечает переменная типа SDL_Renderer. В SDL нет графических примитивов, поэтому рисовать мы можем только точки, прямые и прямоугольники.
- Для отображения различных изображений, текстур, служат структуры SDL_Surface.
- Для отображения текста используется структура SDL_Texture
- Так как наша основная задача – нарисовать график. То в качестве основного контейнера SDL_Window будем использовать SDL_Renderer.

Структура программы

- **Функция инициализации** обрабатывает все загрузки данных.
- **Обработчик событий** обрабатывает все входящие сообщения от мыши, клавиатуры, джойстиков, или других устройств, либо программных таймеров.
- **Игровой цикл** обрабатывает все обновления процесса: сдвиг, поворот и т д
- **Отрисовка сцен** занимается отображением рассчитанных и подготовленных в предыдущей функции сцен на экран, и соответственно никаких манипуляций с данными производить не обязана.
- **Очистка памяти** просто удаляет все загруженные ресурсы из ОЗУ, и обеспечивает корректное завершение программы.

Важно понять, что игры, а SDL, прежде всего, применяется в играх, выполняются в одном цикле. В рамках этого цикла мы обрабатываем события, обновления данных и визуализацию изображений. Таким образом, основная структура игры может рассматриваться так:

```
init();


while(true)


{
    events();
    loop();
    render();
}
 close();
```

Практическая реализация

```
#include <iostream>

#include <SDL.h>
#include <SDL_ttf.h>

using namespace std;

const int SCR_WIDTH = 640; //Ширина окна
const int SCR_HEIGHT = 480; //Высота окна
const int X_ZERO = SCR_WIDTH / 2;
const int Y_ZERO = SCR_HEIGHT / 2;

SDL_Window* gWindow = nullptr; //Наше окно
SDL_Renderer* gRenderer = nullptr; //То, что отрисовывает наш график
TTF_Font* font = nullptr; //Шрифт для отображения подписей (будет рассмотрен
позже)  SDL_Color textColor = { 0, 0, 0 }; //Цвет текста
```

init()

```
bool init()
{
//SDL_Init - отвечает за инициализацию SDL. В случае ошибки возвращает значение < 0.
//Флаг SDL_INIT_VIDEO используется для инициализации графической подсистемы.
//Существует и множество других флагов, например SDL_INIT_EVERYTHING - все подсистемы,
//SDL_INIT_AUDIO - аудио. Информацию обо всех флагах можно найти здесь:
//https://wiki.libsdl.org/SDL_Init
if (SDL_Init(SDL_INIT_VIDEO) < 0) {
        cout << "SDL not initialize! Error: " << SDL_GetError() << endl;
        return false;
}
//Если с SDL все нормально, то переходим к созданию окна:
//Первые два параметра - координаты левого верхнего угла,
//Третий – флаги окна, в данном случае окно будет показано и можно будет изменять его
размер. Подробнее см. https://wiki.libsdl.org/SDL_WindowFlags,
//Четвертый и пятый – указатели на указатели окна и рендера (области для рисования):
if (SDL_CreateWindowAndRenderer(SCR_WIDTH, SCR_HEIGHT,
                                SDL_WINDOW_SHOWN | SDL_WINDOW_RESIZABLE,
                                &gWindow, &gRenderer)) {
        cout << "Window not created! Error: " << SDL_GetError() << endl;
        return false;
} . . .
```

```
//Аналогично SDL инициализируем плагин SDL_ttf:
if (TTF_Init() < 0) {
        cout << "TTF not initialize! Error: " << TTF_GetError() << endl;
        return false;
}

//Загружаем шрифт: путь к файлу и кегль. Здесь указан относительный путь, папка по
//умолчанию – каталог проекта, то место, куда мы копировали DLL:
font = TTF_OpenFont("calibri.ttf", 16);
if (!font) {
        cout << "TTF_OpenFont error: " << TTF_GetError() << endl;
        return false;
}
return true; // Инициализация всех компонентов прошла успешно.
}
```

close()

```
void close()
{
//Необходимо удалять все созданные
объекты!
//Удаляем рендер и окно:
SDL_DestroyRenderer(gRenderer);
SDL_DestroyWindow(gWindow);
TTF_CloseFont(font);

//Обнуляем указатели:
gWindow = NULL;
gRenderer = NULL;
font = NULL;

//Отключаем SDL и
плагины:  TTF_Quit();
SDL_Quit();
}
```

loop()

```
void loop()
{
//Устанавливаем цвет в рендере. Напомню, что цвет задается в RGB
//и еще один параметр для альфа-канала
SDL_SetRenderDrawColor(gRenderer, 255, 255, 255, 255);
//Очищаем рендер. Фоновым цветом станет последний выбранный,
//в данном случае – белый:
SDL_RenderClear(gRenderer);

//Устанавливаем в gRenderer серый цвет:
SDL_SetRenderDrawColor(gRenderer, 200, 200, 200, 255);
int stepX = 20;
int stepY = 20;
int arr = 5;
//В gRender рисуем сетку, т. е. линии проходящие через весь экран:
for (int i = 0; i<=SCR_WIDTH /stepX; i++)
//из точек (stepX * i, 0) в точки (stepX * i, SCR_HEIGHT)
SDL_RenderDrawLine(gRenderer, stepX * i, 0, stepX * i, SCR_HEIGHT);

for (int i = 0; i<=SCR_HEIGHT /stepY; i++)
//из точек (0, stepY * i, ) в точки (SCR_WIDTH, stepY * i)
SDL_RenderDrawLine(gRenderer, 0, stepY * i, SCR_WIDTH, stepY * i);
//Устанавливаем в gRender черный цвет
SDL_SetRenderDrawColor(gRenderer, 0, 0, 0, 255);
. . .
```

```
//Рисуем оси:

//Абсциссы:
SDL_RenderDrawLine(gRenderer, X_ZERO, 0, X_ZERO, SCR_HEIGHT);
SDL_RenderDrawLine(gRenderer, SCR_WIDTH, Y_ZERO, SCR_WIDTH - arr, Y_ZERO +arr);
SDL_RenderDrawLine(gRenderer, SCR_WIDTH, Y_ZERO, SCR_WIDTH - arr, Y_ZERO - arr);

//Ординаты:
SDL_RenderDrawLine(gRenderer, 0, Y_ZERO, SCR_WIDTH, Y_ZERO);
SDL_RenderDrawLine(gRenderer, X_ZERO, 0, X_ZERO - arr, arr);
SDL_RenderDrawLine(gRenderer, X_ZERO, 0, X_ZERO + arr, arr);

//Подписи к осям:
print_graph("X", SCR_WIDTH - arr-10, Y_ZERO + arr+2);
print_graph("Y", X_ZERO - arr-10, arr+2);
}
```

SDL_ttf

- Как уже было сказано ранее, SDL сам по себе не может печатать текст. Поэтому, для отрисовки текста, приходится использовать модуль SDL_ttf.
- Для того, чтобы напечатать текст в окне, необходимо сначала создать для него SDL_Texture, затем поместить ее на SDL_Render, и лишь затем печатать ее в окне.

Задание функции

Нашу задачу можно представить аналитически
F(x)=b*f(a*x+c)+d
Где:
a-задаёт растяжение
b-задаёт сжатие
c-сдвиг по оси X
d-сдвиг по оси Y

```
// Зададим f(x)
double fx(double x){
return sin(x);
}
```

print_graph()

```
void print_graph(int a, int b, int c, int d, char* s){
SDL_SetRenderDrawColor(gRenderer, 0, 0, 0, 255);
for (int i = 0; i < SCR_WIDTH; i++){
SDL_RenderDrawLine(gRenderer, i,
Y_ZERO-int(fx((i+d-X_ZERO)*a/2000.0)*20*b/100.0)+c,
i+1, Y_ZERO-int(fx((i+1+d-X_ZERO)*a/2000.0)*20*b/100.0)+c);
}
/*
Пояснение:
 a-задаёт растяжение изначально =100
 b-задаёт сжатие     изначально =100
 c-сдвиг по оси X    изначально =0
 d-сдвиг по оси Y    изначально =0

int(fx((i + d - X_ZERO)*a / 2000.0) * 20 * b / 100.0) – расчет
значения функции в пикселе i
в функции loop() задана сетка 1=20 пикселей
*/
```

```
//подписи
itoa(a, s, 10);
print_graph("a=", 10, 20);
print_graph(s, 30, 20);
print_graph("%", 60, 20);
itoa(b, s, 10);
print_graph("b=", 10, 40);
print_graph(s, 30, 40);
print_graph("%", 60, 40);
itoa(-c, s, 10);
print_graph("c=", 10, 60);
print_graph(s, 30, 60);
itoa(-d, s, 10);
print_graph("d=", 10, 80);
print_graph(s, 30, 80);
print_graph("Y=b*f(a*X+d)+c", 10, 100);
}
```

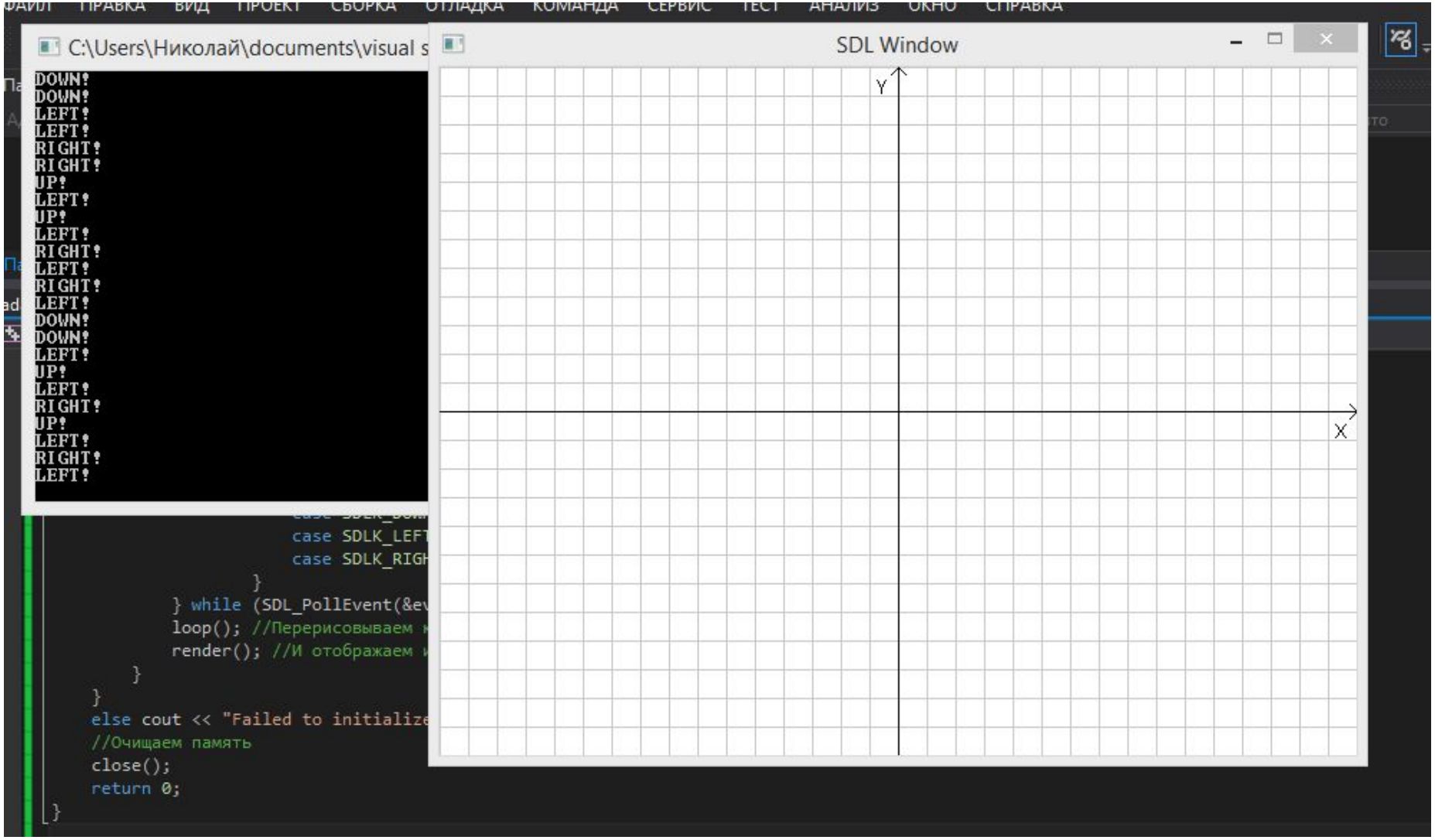
render()

```
void render()
{
//Теперь необходимо отобразить все то, что было нарисовано процедурой loop()
SDL_RenderPresent(gRenderer);
}
```

main

```
int main(int argc, char* args[]){//Без аргументов у main все сломается
if (init()){ //После удачной инициализации
    bool quit = false; //Флаг завершения игрового цикла
    SDL_Event event; //Событие (с клавиатуры, мыши, геймпада и пр.)
     while (!quit) {
       //Ждем и обрабатываем события
       SDL_WaitEvent(&event); //Как только произошло какое-нибудь событие
       do {
           if (event.type == SDL_QUIT) //Нажатие на "крестик"
           quit = true;
           //Реакция на зажатую кнопку
             if (event.type = SDL_KEYDOWN)
           switch (event.key.keysym.sym) {
           case SDLK_UP: cout << "UP!\n"; y += 5; break;// Вверх
           case SDLK_DOWN: cout << "DOWN!\n"; if (y > 0){ y -= 5; } break;// Вниз
           case SDLK_LEFT: cout << "LEFT!\n"; if (x > 0){ x -= 10; } break;// Влево
           case SDLK_RIGHT: cout << "RIGHT!\n"; x += 10; break;// Вправо
           case SDLK_KP_8: cout << "u!\n"; c -= 1; break;
           case SDLK_KP_2: cout << "d!\n"; c += 1; break;
           case SDLK_KP_6: cout << "r!\n"; d -= 1; break;
           case SDLK_KP_4: cout << "l!\n"; d += 1; break;}
       } while (SDL_PollEvent(&event)!=0);//Начинаем просматривать все события в очереди
       loop(); print_graph(x,y,c,d,s); //Перерисовываем кадр
       render(); //И отображаем изменения
    }
} else cout << "Failed to initialize!\n";
close();//Очищаем память
return 0; //Обязательно должны возвратить значение

}
```


ФАЙЛ ПРАВКА ВИД ПРОЕКТ СБОРКА ОТЛАДКА КОМАНДА СЕРВИС ТЕСТ АНАЛИЗ ОКНО СПРАВКА
C:\Users\Николай\documents\visual s
DOWN!
DOWN!
LEFT!
LEFT!
RIGHT!
RIGHT!
UP!
LEFT!
UP!
LEFT!
RIGHT!
LEFT!
RIGHT!
LEFT!
DOWN!
DOWN!
LEFT!
UP!
LEFT!
RIGHT!
UP!
LEFT!
RIGHT!
LEFT!
SDL Window
Y
X
case SDLK_LEFT
case SDLK_RIGH
} while (SDL_PollEvent(&ev
loop(); //Перерисовываем
render(); //И отображаем
}
}
else cout << "Failed to initialize
//Очищаем память
close();
return 0;
}

Список литературы:

- https://ru.wikipedia.org/wiki/Simple_DirectMedia_Layer
- http://libsdl.org/index.php
- http://lazyfoo.net/tutorials/SDL/
- http://habrahabr.ru/post/166875/
- https://wiki.libsdl.org
- http://habrahabr.ru/post/201392/