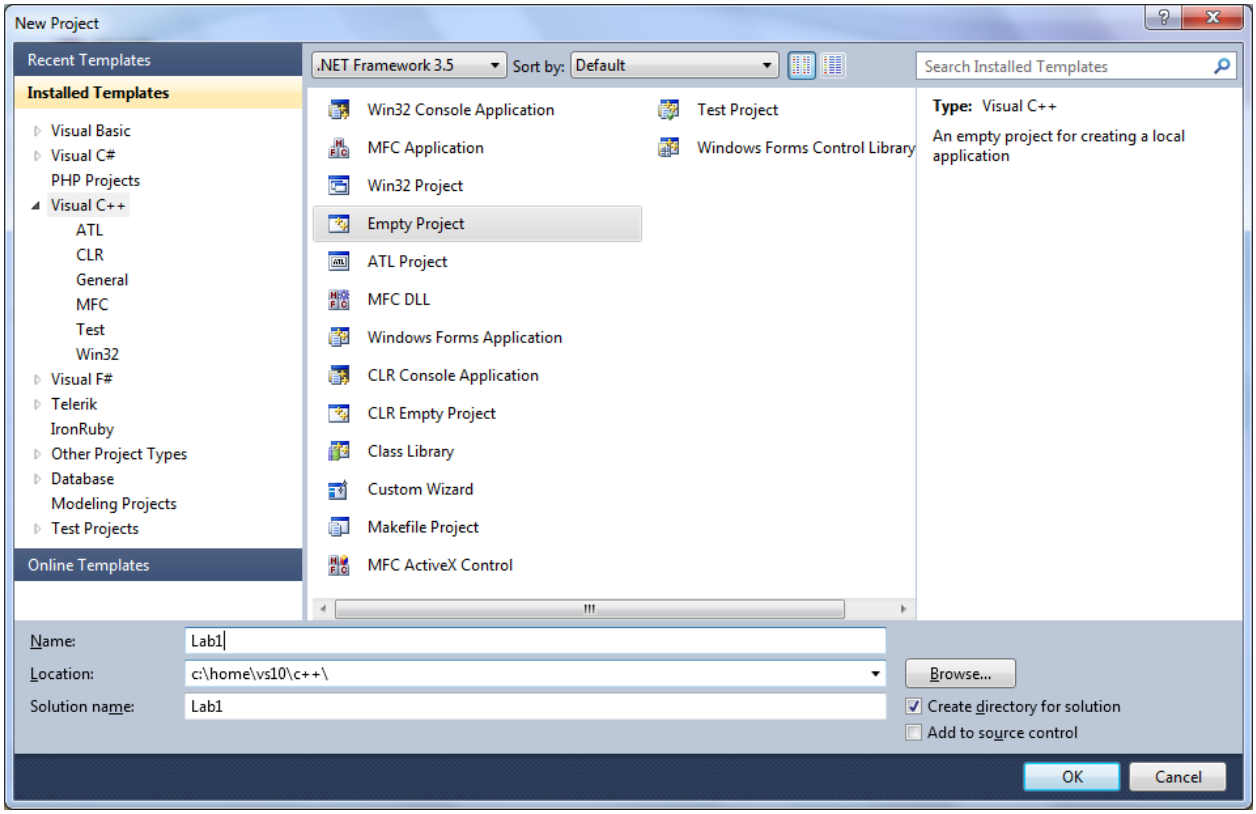
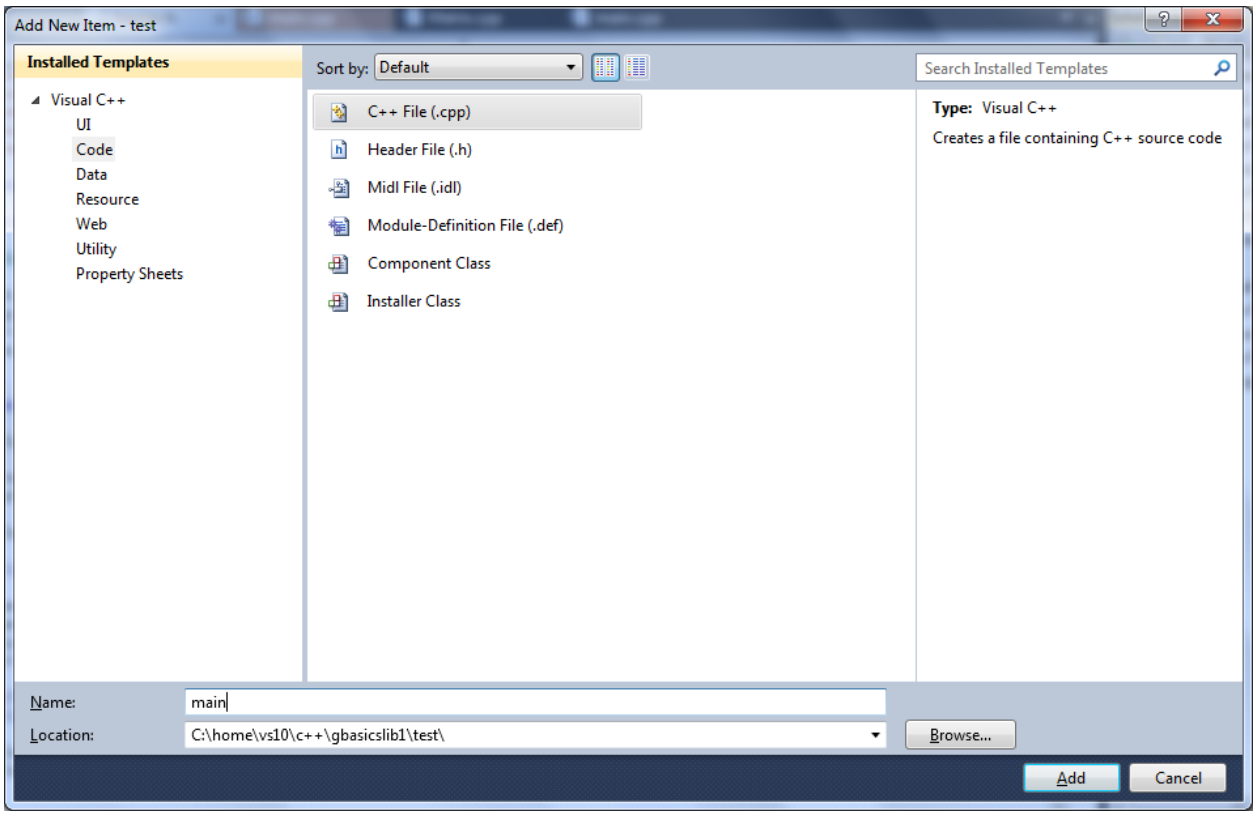
Лабораторные работы

Для выполнения лабораторных работ применима среда Microsoft Visual C++ Express. Выполнение работы начинается с создания проекта, который представляет собой совокупность исходных кодов, ресурсов и прочей информации, необходимой для сборки исполняемого файла.

Выберем пункт New project либо на приветственном экране, либо в меню File. В категории Visual C++ нас интересует шаблон проекта “Empty project”. Он не содержит никаких специфичных для C++ включений, что при изучении Си подходит лучше всего. Введём название для проекта, например Lab1, выберем, где он будет располагаться, и нажмём OK.



После создания проекта, необходимо добавить в него файл для исходного кода. Найдем окно Solution Explorer, кликнем в нём по иконке проекта правой кнопкой мыши, в появившемся меню выберем Add --> New item. В категории Code выберем C++ File, дадим ему имя main и нажмём Add.



В лабораторных работах мы будем использовать специальную библиотеку готовых функций для вывода графики, которую необходимо подключить. Для этого снова кликнем по иконке проекта в окошке Solution Explorer и выберем пункт Open Folder in Windows Explorer. Откроется папка, в которой расположены файлы проекта. Скопируем в неё gbasicslib1.lib и gbasicslib1.h.

Можно приступать к написанию программы!

Минимальный шаблон, с которого можно начать выполнение лабораторной выглядит так:

```
// подключаем заголовочный файл учебной библиотеки
#include "gbasicslib1.h"

// подключаем саму библиотеку
#pragma comment (lib, "gbasicslib1.lib")

// здесь будут описания структур данных и функций

void main()
{
 // инициализация окна, аргументы - размеры и текст заголовка
 init(800, 600, "Lab 1");

 // главный цикл. doEvent() возвращает true пока окно не закрыто
 while (doEvent())
 {
       clear(); // очистить буфер изображения

       // здесь будет код, необходимый для отрисовки

       present(); // показать содержимое буфера на экране

       if (keyPressed()) // возвращает true, если была нажата клавиша
       {
             int keyCode = getLastVKey(); // получает код нажатой клавиши
```

```
                // в лабораторных нам достаточно клавиш,
                // коды которых описаны константами VK_*
                // всплывающую подсказку можно увидеть,
                // набрав VK_ и нажав Ctrl+Space
                if (keyCode == VK_ESCAPE)
                {
                      break;
                }
          }
    }
}
```

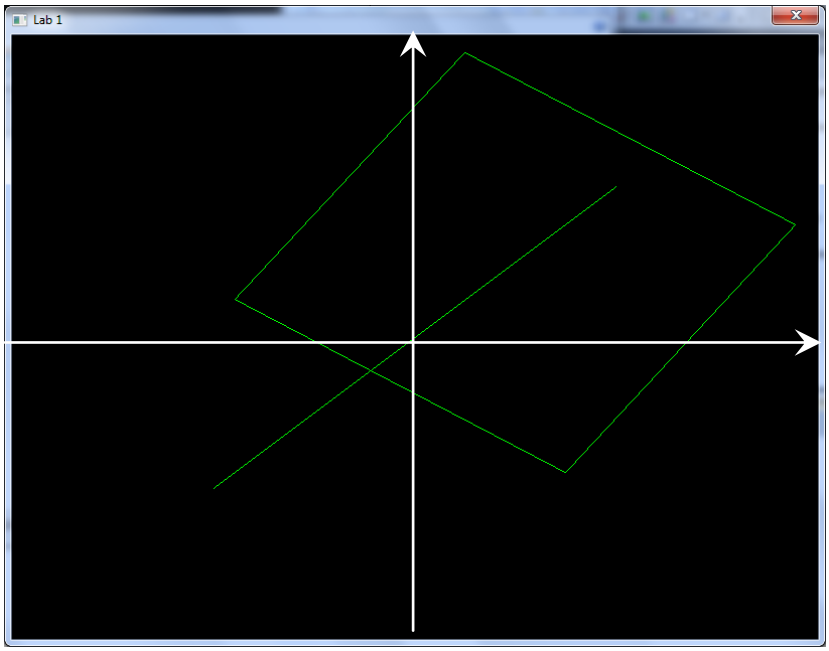
Кроме показанных в шаблоне, библиотека включает несколько функций для рисования:

```
void setPixel(int x, int y, Color4b color);

void line(int x, int y, int x2, int y2, Color4b color);
void line(float x, float y, float x2, float y2, Color4b color);

void circle(int ox, int oy, int r, Color4b c);
void circle(float ox, float oy, float r, Color4b color);
```

Принимающие аргументами целые числа, действуют в системе оконных координат соответственно размерам окна. Принимающие числа с плавающей точкой – в показанной на иллюстрации:



При этом края окна соответствуют 1 и -1 по соответствующим осям.

Лабораторная #1 – “Трансформации векторов на плоскости”

Цель работы

Изучить базовые принципы построения динамичных графических приложений : главный цикл, вектор – элементарная единица геометрии, трансформации вектора на плоскости.

Задание

Написать программу, при старте которой в консоли запрашиваются координаты точки. Затем в появившемся окне должна показываться заданная вариантом фигура, вращающаяся вокруг указанной точки. При нажатии клавиш Вверх\Вниз на клавиатуре скорость вращения должна меняться.

Заголовок окна с графическим выводом должен содержать ФИО студента, номер группы и номер лабораторной работы.

Алгоритм выполнения работы

- Составить структуру для описания вектора на плоскости с координатами X и Y
- Написать функцию поворота вектора вокруг центра координат на плоскости
- С помощью этой функции и функций учебной библиотеки изобразить вращающийся треугольник
- Составить структуру для описания матрицы трансформации вращения вектора на плоскости; Написать функции инициализации матрицы вращения и трансформации вектора при помощи этой матрицы
- Усовершенствовать основную программу с использованием новых функций
- Усовершенствовать структуру матрицы трансформации так, чтоб с её помощью можно было выполнять трансформацию переноса (смещения) центра координат; Написать функцию инициализации матрицы переноса и исправить существующие функции для работы с новой матрицей
- Написать функцию для перемножения двух матриц трансформации
- Довести основную программу до соответствия заданию лабораторной работы

Пример программы, выполняющей похожую задачу

Данная программа демонстрирует смещённый в сторону первой координатной четверти квадрат, который циклично масштабируется, оказываясь сплющенным то вдоль оси X, то вдоль оси Y.

```
void main()
{
 // инициалзация окна, в которое ведется графический вывод
 init(800, 600, "Lab 1 sample");

 // переменные для масштабирования с начальными значениями
 float scale = 1.0f, scaleStep = 0.01f;

 // цвет, которым рисуется фигура
 Color4b color = {255, 255, 255, 0};

 // вектора, описывающие вершины фигуры
 Vector2f v1 = {0.5f, 0.5f},
             v2 = {-0.5f, 0.5f},
             v3 = {-0.5f, -0.5f},
             v4 = {0.5f, -0.5f};

 // матрицу смещения описываем заранее т.к. она не меняется
 Matrix3f offsetTransform = TranslationMatrix(0.25f, 0.25f);

 // главный цикл
 while (doEvent())
 {
       // очищаем буфер изображения
       clear();
```

```
            // переменные для хранения трансформированных векторов вершин
            Vector2f tv1, tv2, tv3, tv4;

            // получаем полное преобразование умножением
            // матрицы масштабирования на матрицу смещения
            Matrix3f m = Multiply(
                  ScalingMatrix(scale, 2.0f - scale),
                  offsetTransform
            );

            // трансформируем все вершины полным преоразованием,
            // получая масштабированные и смещённые вектора
            TransformVector(v1, m, tv1);
            TransformVector(v2, m, tv2);
            TransformVector(v3, m, tv3);
            TransformVector(v4, m, tv4);

            // рисуем в буфер изображения линии, соединющие вершины фигуры
            line(tv1.X, tv1.Y, tv2.X, tv2.Y, color);
            line(tv2.X, tv2.Y, tv3.X, tv3.Y, color);
            line(tv3.X, tv3.Y, tv4.X, tv4.Y, color);
            line(tv4.X, tv4.Y, tv1.X, tv1.Y, color);

            // изменяем масштаб на шаг масштабирования
            scale += scaleStep;

            // по достижении масштабом краевого значения
            if (scale < 0.5f || scale > 1.5f)
            {
                  // инвертируем шаг масштабирования
                  scaleStep *= -1.0f;
            }

            // показываем в окне содержимое буфера изображения
            present();

            // проверяем, не была ли нажата какая-нибудь клавиша
            if (keyPressed())
            {
                  // если да – получаем код нажатой клавиши
                  int keyCode = getLastVKey();

                  if (keyCode == VK_ESCAPE)
                  {
                        // при нажатии [Escape] выходим из главного
                        // цикла, и программа завершается
                        break;
                  }
            }
      }
}
```

Варианты

В качестве фигуры предлагается выбрать букву русского алфавита с номером, соответствующим порядковому номеру студента в группе.

Лабораторная #2 – “Простая игра в двухмерном пространстве”

Цель работы

Применить полученные при выполнении первой лабораторной работы навыки для создания игры.

Задание

Выбрать и согласовать с преподавателем вариант реализуемой игры.

Реализовать игру.

Заголовок окна должен содержать ФИО студента, номер группы, номер лабораторной работы и номер варианта (название игры).

Алгоритм выполнения работы

- Составить основные алгоритмы игровой логики в соответствии с вариантом
- Составить необходимые для реализации игровой логики структуры данных
- Запрограммировать игровую логику и продемонстрировать получившуюся программу преподавателю
- Дополнить программу в соответствии с рекомендациями преподавателя

Варианты

1. Сверху вниз падают объекты (представленные в виде простых геометрических фигур), необходимо их ловить.
2. Космический корабль, летящий сквозь облако астероидов, от которых необходимо уклоняться.
3. Змейка.
4. Снежки 2D
5. Арканоид.
6. Тетрис.
7. Волейбол.
8. Packman.
9. Танки.

Лабораторная #3 – “Трансформации векторов в пространстве”

Цель работы

Изучить трансформации вектора в пространстве.

Задание

Написать программу, показывающую в окне заданную вариантом вращающуюся в горизонтальной плоскости фигуру. При нажатии на клавиши Вверх\Вниз должно осуществляться вращение в вертикальной плоскости.

Заголовок окна с графическим выводом должен содержать ФИО студента, номер группы и номер лабораторной работы.

Алгоритм выполнения работы

- Составить структуру для описания вектора в пространстве с координатами X, Y и Z
- Составить структуру для описания матрицы трансформации вектора в пространстве
- Написать функции инициализации матриц вращения в пространстве относительно разных осей, матриц переноса, масштабирования и перспективной проекции, функцию для перемножения двух матриц и функцию трансформации вектора
- Довести основную программу до соответствия заданию лабораторной работы

Пример программы, выполняющей похожую задачу

[...]

Варианты

[...]

Некоторые функции из стандартной библиотеки Си

Могут понадобиться при выполнении лабораторных работ. За более полными описаниями самих функций и их аргументов рекомендуется обращаться к полной документации (например, MSDN).

Ввод-вывод

int printf(const char* format, …) и аналоги

Осуществляет вывод в stdout строки символов, сформированной в соответствии с форматной строкой format.

Форматная строка может включать ссылки на значения других аргументов функции, форматируемые в согласно спецификации и подставляемые в соответствующие места выходной строки. Минимальная форма такой ссылки: "%[символ]", – где символ определяет, как функция интерпретирует соответствующий ссылке аргумент.

Символ	Интерпретация аргумента	Вывод
c	Код символа	Символ
i d	Знаковое целое	Десятичное число
u	Беззнаковое целое	Десятичное число
x		Шестнадцатеричное число с использованием символов "abcdef"
X		То же что 'x', но с символами "ABCDEF"
e	Число с плавающей точкой	Число в экспоненциальной форме вида [-]d.dddd e [sign] dd[d]
f		Десятичная дробь вида [-]dddd.dddd
s	Указатель на нуль-терминированную строку символов	Строка

stdout (поток стандартного вывода) по-умолчанию ассоциирован с консолью.

Из аналогичных функций могут пригодиться:

- int fprintf (FILE *stream, const char *format [, argument, ...]);
  Вывод происходит в предварительно открытый функцией fopen(..) файл, указываемый первым аргументом.
- int sprintf (char *buffer, const char *format [, argument, ...]);
  Вывод происходит в заданный первым аргументом буфер, представляющий из себя последовательность байт длиной не менее выходной строки.

Возвращаемое значение – длина выходной строки, либо EOF в случае ошибки.

Пример использования:

```
int i = 100;
char c = 'C';
float f = 1.234;
char* s = "some string";

printf("Here are some values : %i, %c, %f and '%s'\n", i, c, f, s);
```

int scanf(const char* format, …) и аналоги

Осуществляет ввод из stdin в соответствии со списком шаблонов, заданных аргументом format, последовательности значений и присвоение этих значений соответствующим переменным, перечисленным в последних аргументах.

Литература

<1> “Информатика. 6-7 класс” ред. Н. В. Макаровой 1998 ISBN 5-314-00145-4
<2> “Инцеклопедия языка Си” Я. Белецкий 1992 ISBN 5-03-002113-2
<3> “Компьютерная графика” В. Н. Порев 2004 ISBN 5-94157-139-9