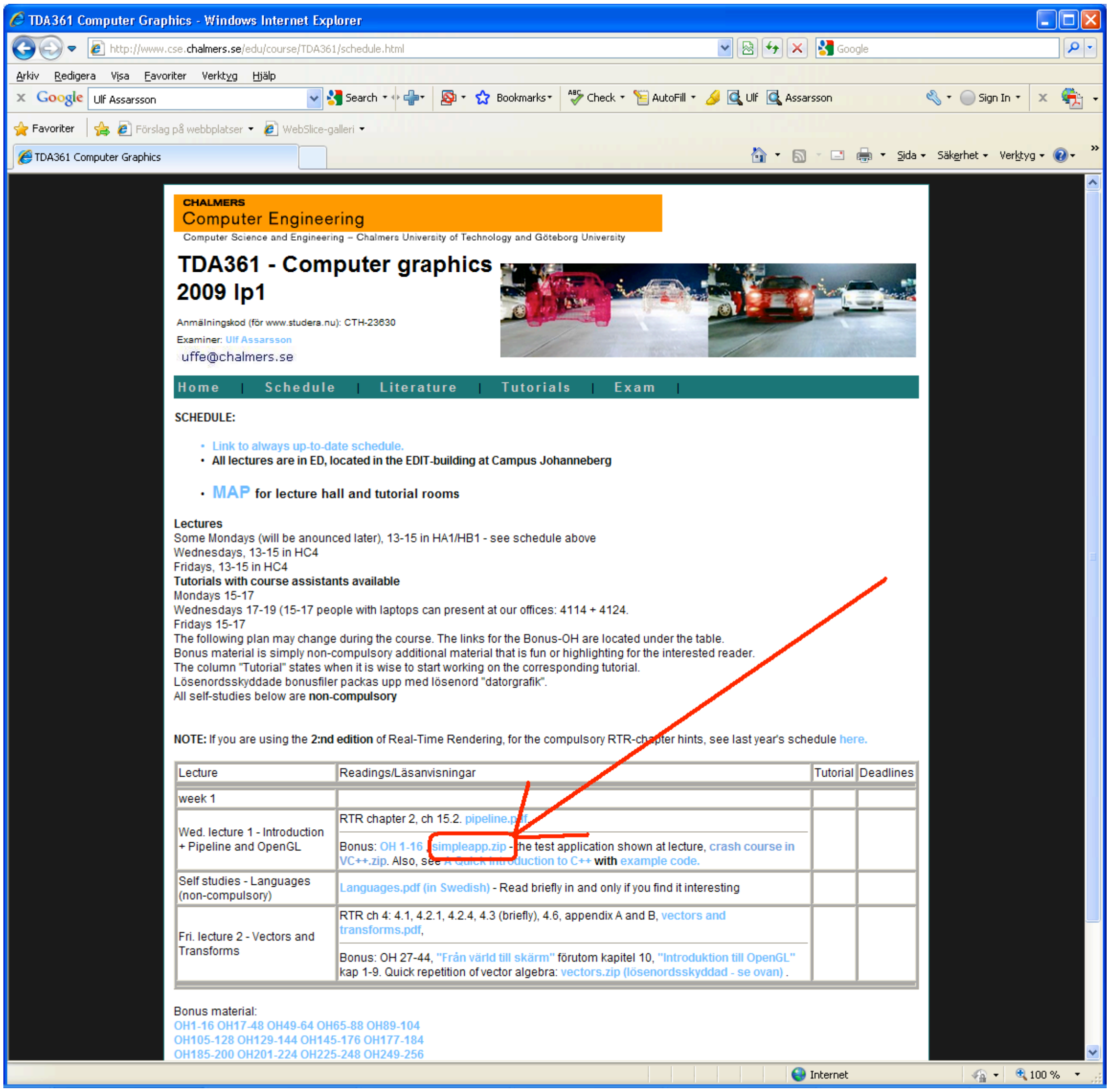
A Crash Course in Microsoft Visual C++ 2008

This document presents the most essential things to know in order to program with VC++. This document is intended to use with the provided “OpenGLTutorials.zip-file” and is part of the TDA 361 - Computer Graphics course at Chalmers.

Copying and modification of this material is free.

Download the OpenGLTutorials.zip file from the course home page:


TDA361 Computer Graphics - Windows Internet Explorer
http://www.cse.chalmers.se/edu/course/TDA361/schedule.html
Google
Arkiv Redigera Visa Favoriter Verktyg Hjälp
Google Ulf Assarsson Search Bookmarks Check AutoFill Ulf Assarsson Sign In
Favoriter Förslag på webbplatser WebSlice-galleri
TDA361 Computer Graphics
Sida Säkerhet Verktyg
Internet 100 %

CHALMERS
Computer Engineering
Computer Science and Engineering – Chalmers University of Technology and Göteborg University

TDA361 - Computer graphics 2009 lp1

Anmälningskod (för www.studera.nu): CTH-23630
Examiner: Ulf Assarsson
uffe@chalmers.se

Home | Schedule | Literature | Tutorials | Exam |

SCHEDULE:

- Link to always up-to-date schedule.
- All lectures are in ED, located in the EDIT-building at Campus Johanneberg
- MAP for lecture hall and tutorial rooms

Lectures
Some Mondays (will be anounced later), 13-15 in HA1/HB1 - see schedule above
Wednesdays, 13-15 in HC4
Fridays, 13-15 in HC4
Tutorials with course assistants available
Mondays 15-17
Wednesdays 17-19 (15-17 people with laptops can present at our offices: 4114 + 4124.
Fridays 15-17
The following plan may change during the course. The links for the Bonus-OH are located under the table.
Bonus material is simply non-compulsory additional material that is fun or highlighting for the interested reader.
The column "Tutorial" states when it is wise to start working on the corresponding tutorial.
Lösenordsskyddade bonusfiler packas upp med lösenord "datorgrafik".
All self-studies below are **non-compulsory**

NOTE: If you are using the **2:nd edition** of Real-Time Rendering, for the compulsory RTR-chapter hints, see last year's schedule here.

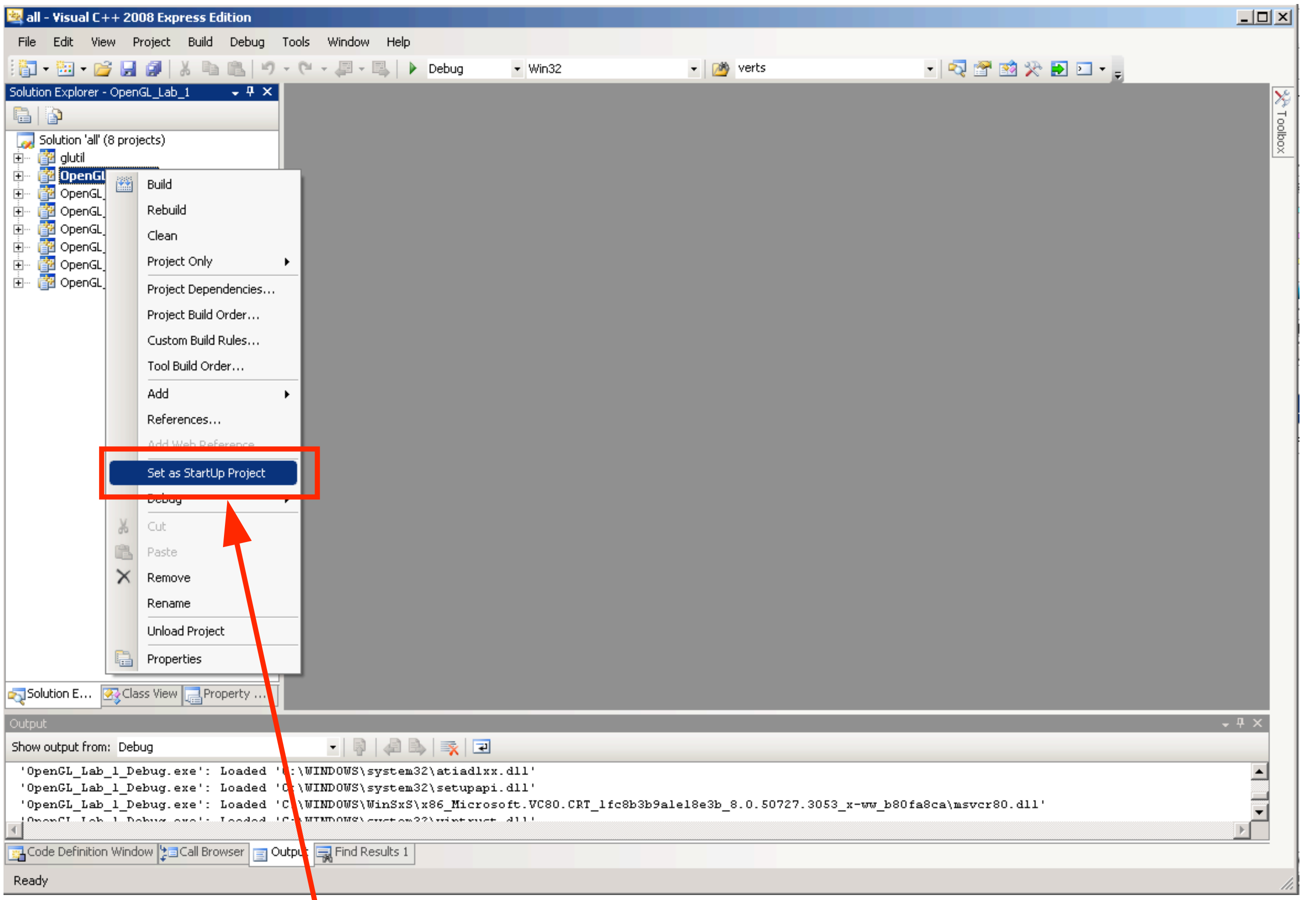
Lecture	Readings/Läsanvisningar	Tutorial	Deadlines
week 1			
Wed. lecture 1 - Introduction + Pipeline and OpenGL	RTR chapter 2, ch 15.2. pipeline.pdf. Bonus: OH 1-16 , simpleapp.zip - the test application shown at lecture, crash course in VC++.zip. Also, see A Quick Introduction to C++ **with** example code.		
Self studies - Languages (non-compulsory)	Languages.pdf (in Swedish) - Read briefly in and only if you find it interesting		
Fri. lecture 2 - Vectors and Transforms	RTR ch 4: 4.1, 4.2.1, 4.2.4, 4.3 (briefly), 4.6, appendix A and B, vectors and transforms.pdf, Bonus: OH 27-44, "Från värld till skärm" förutom kapitel 10, "Introduktion till OpenGL" kap 1-9. Quick repetition of vector algebra: vectors.zip (lösenordsskyddad - se ovan) .		

Bonus material:
OH1-16 OH17-48 OH49-64 OH65-88 OH89-104
OH105-128 OH129-144 OH145-176 OH177-184
OH185-200 OH201-224 OH225-248 OH249-256

Unzip the OpenGLTutorials.zip -file and find the all.sln-file which is located in the “OpenGLTutorials”- directory where you chose to unzip. Simply double click this .sln file to start VC++. Should VC++ want to convert the project from 2005 to 2008 then let it do so.

NOTE: If this is the first time you start Visual Studio, be sure to select C++ developer profile (or whatever it is called).

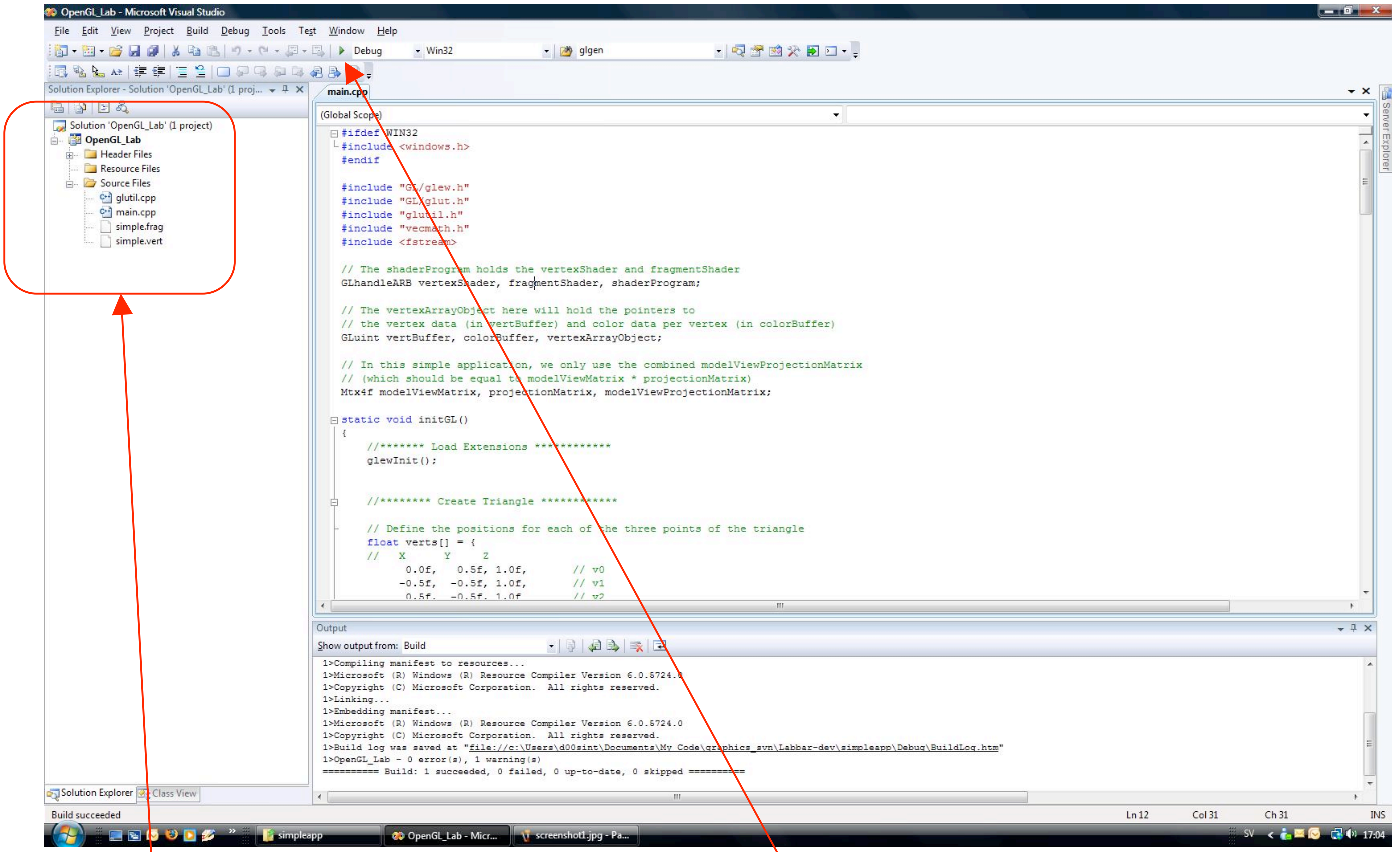
After a few seconds a window similar to this should appear:



Show output from: Debug

```
'OpenGL_Lab_1_Debug.exe': Loaded 'C:\WINDOWS\system32\atiadlxx.dll'
'OpenGL_Lab_1_Debug.exe': Loaded 'C:\WINDOWS\system32\setupapi.dll'
'OpenGL_Lab_1_Debug.exe': Loaded 'C:\WINDOWS\WinSxS\x86_Microsoft.VC80.CRT_1fc8b3b9a1e18e3b_8.0.50727.3053_x-ww_b80fa8ca\msvcr80.dll'
'OpenGL_Lab_1_Debug.exe': Loaded 'C:\WINDOWS\system32\wintrust.dll'
```

This solution contains all the OpenGL tutorials for the course. To select which of the project is to be active, right click on the project (in this case select OpenGL_Lab_1) and select “Set as Startup Project”.



```
#ifdef WIN32
#include <windows.h>
#endif

#include "GL/glew.h"
#include "GL/glut.h"
#include "glutil.h"
#include "vecmath.h"
#include <fstream>

// The shaderProgram holds the vertexShader and fragmentShader
GLhandleARB vertexShader, fragmentShader, shaderProgram;

// The vertexArrayObject here will hold the pointers to
// the vertex data (in vertBuffer) and color data per vertex (in colorBuffer)
GLuint vertBuffer, colorBuffer, vertexArrayObject;

// In this simple application, we only use the combined modelViewProjectionMatrix
// (which should be equal to modelViewMatrix * projectionMatrix)
Mtx4f modelViewMatrix, projectionMatrix, modelViewProjectionMatrix;

static void initGL()
{
    //******* Load Extensions ************
    glewInit();


    //******** Create Triangle ************

    // Define the positions for each of the three points of the triangle
    float verts[] = {
    //   X      Y     Z
          0.0f,   0.5f, 1.0f,        // v0
         -0.5f,  -0.5f, 1.0f,        // v1
          0.5f,  -0.5f, 1.0f         // v2
```

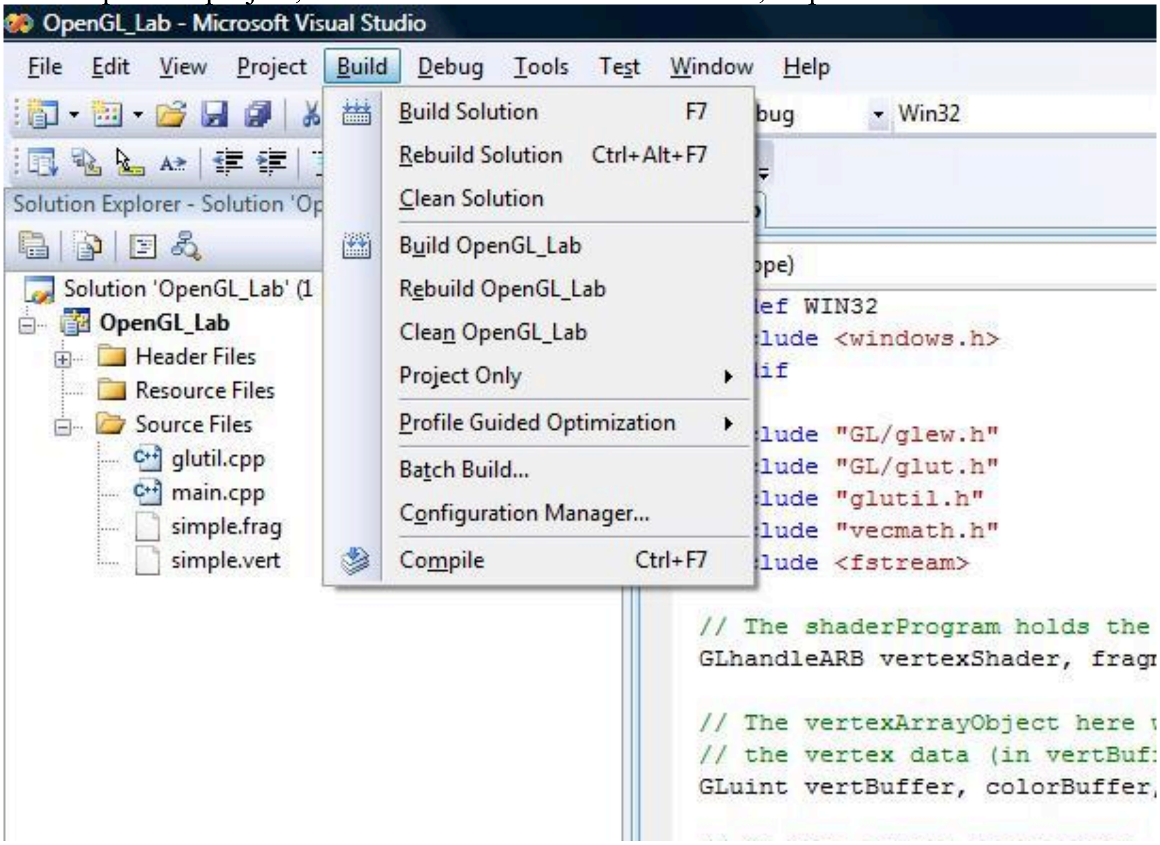
```
1>Compiling manifest to resources...
1>Microsoft (R) Windows (R) Resource Compiler Version 6.0.5724.0
1>Copyright (C) Microsoft Corporation.  All rights reserved.
1>Linking...
1>Embedding manifest...
1>Microsoft (R) Windows (R) Resource Compiler Version 6.0.5724.0
1>Copyright (C) Microsoft Corporation.  All rights reserved.
1>Build log was saved at "file://c:\Users\d00sint\Documents\My Code\graphics_svn\Labbar-dev\simpleapp\Debug\BuildLog.htm"
1>OpenGL_Lab - 0 error(s), 1 warning(s)
========== Build: 1 succeeded, 0 failed, 0 up-to-date, 0 skipped ==========
```

To the left, you see the files in the projects.

Compile and run the project by pressing F5, or clicking this “play” button with a green arrow.

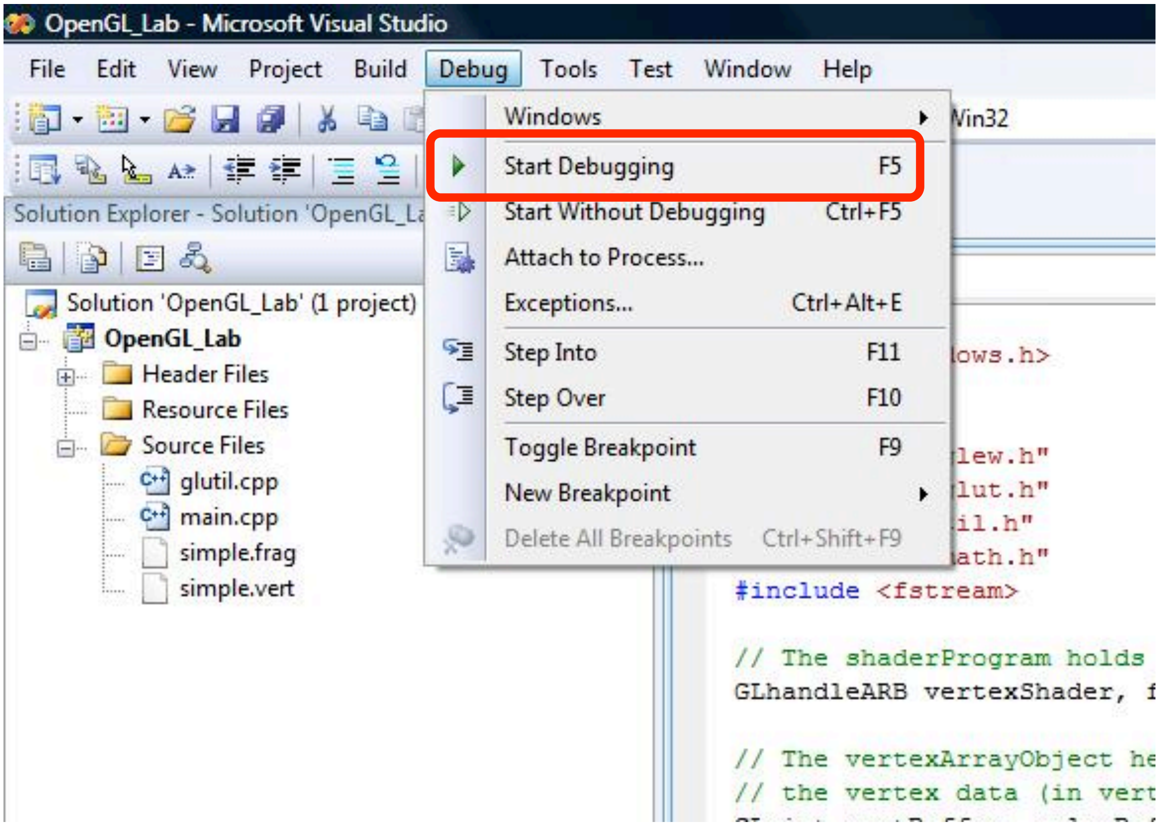
When the program starts, you should see a white triangle on a blue background.

To compile the project, select menu Build->Build Solution, or press F7:



```
ope)
lef WIN32
lude <windows.h>
lif

lude "GL/glew.h"
lude "GL/glut.h"
lude "glutil.h"
lude "vecmath.h"
lude <fstream>

// The shaderProgram holds the
GLhandleARB vertexShader, fragm

// The vertexArrayObject here w
// the vertex data (in vertBuf
GLuint vertBuffer, colorBuffer,
```

To run the application, press F5 or select menu Debug->Start.



```
ows.h>

lew.h"
lut.h"
il.h"
ath.h"
#include <fstream>

// The shaderProgram holds
GLhandleARB vertexShader, f

// The vertexArrayObject he
// the vertex data (in vert
```

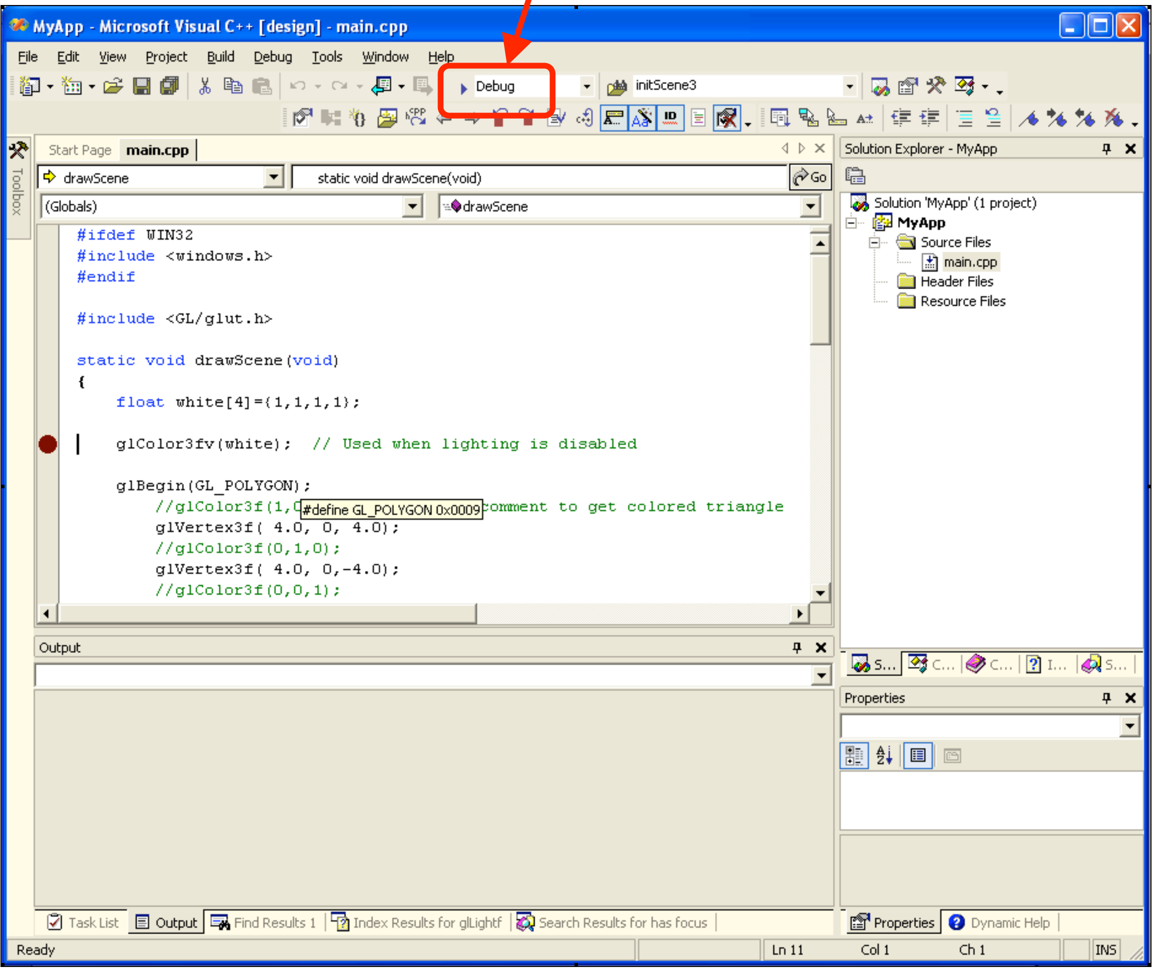
The images below are from using VC++ 2003, but the principles are still the same.

Using the debugger

Using the debugger is very easy and is highly recommended.

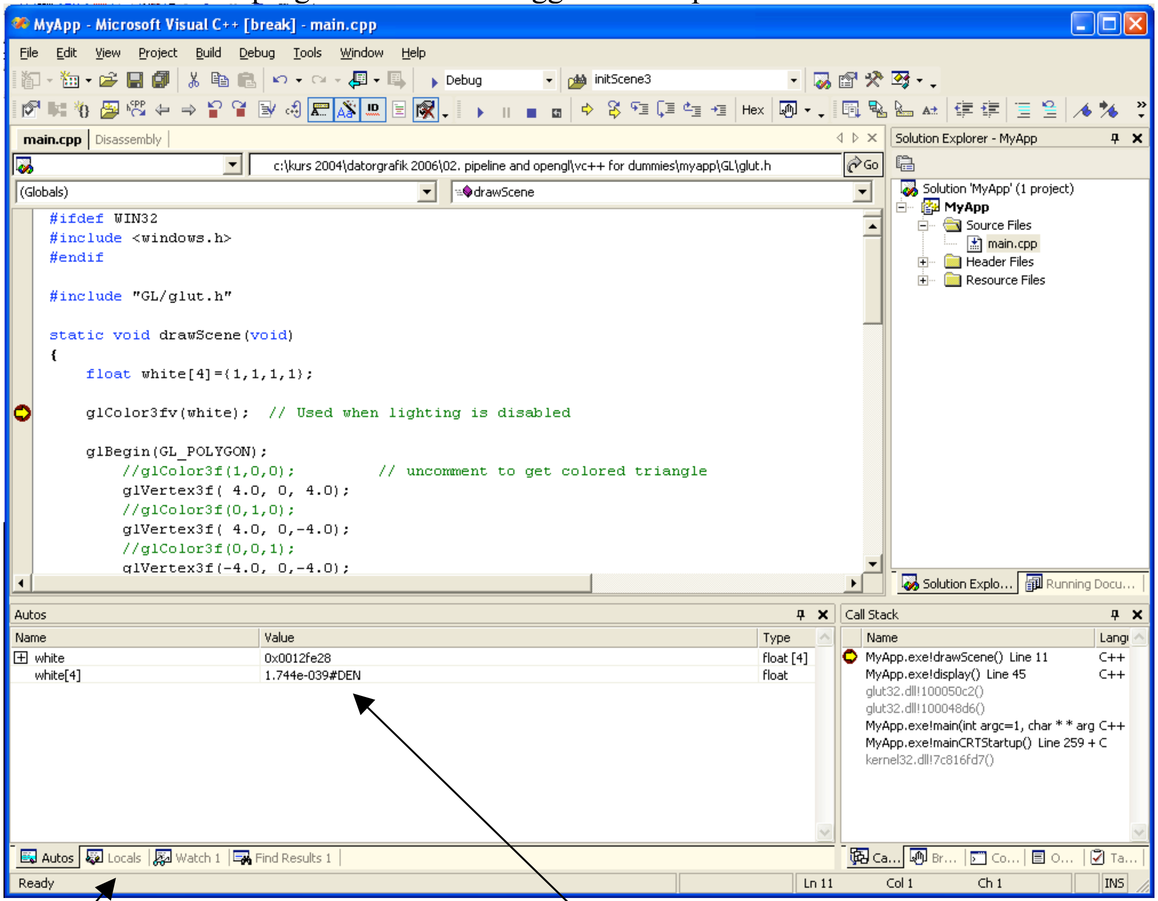
F9 sets/removes a break point.
Set a break point by pressing F9 at the desired location in your source code. Remove the breakpoint by pressing F9 at the same location.

Note: Ensure the project is set to build in debug mode; otherwise you will not be able to set a breakpoint or inspect variables.



```
#ifdef WIN32
#include <windows.h>
#endif

#include <GL/glut.h>

static void drawScene(void)
{
    float white[4]={1,1,1,1};

    glColor3fv(white);  // Used when lighting is disabled

    glBegin(GL_POLYGON);
        //glColor3f(1,C#define GL_POLYGON 0x0009comment to get colored triangle
        glVertex3f( 4.0, 0, 4.0);
        //glColor3f(0,1,0);
        glVertex3f( 4.0, 0,-4.0);
        //glColor3f(0,0,1);
```

Press F5 to run the program and the debugger will stop at the location. Do this!


MyApp - Microsoft Visual C++ [break] - main.cpp
File Edit View Project Build Debug Tools Window Help
Debug
initScene3
Hex
main.cpp
Disassembly
c:\kurs 2004\datorgrafik 2006\02. pipeline and opengl\vc++ for dummies\myapp\GL\glut.h
Go
(Globals)
drawScene
Solution Explorer - MyApp
Solution 'MyApp' (1 project)
MyApp
Source Files
main.cpp
Header Files
Resource Files
Solution Explo...
Running Docu...
Autos
Locals
Watch 1
Find Results 1
Ca...
Br...
Co...
O...
Ta...
Ready
Ln 11
Col 1
Ch 1
INS

```
#ifdef WIN32
#include <windows.h>
#endif

#include "GL/glut.h"

static void drawScene(void)
{
    float white[4]={1,1,1,1};

    glColor3fv(white);  // Used when lighting is disabled

    glBegin(GL_POLYGON);
        //glColor3f(1,0,0);         // uncomment to get colored triangle
        glVertex3f( 4.0, 0, 4.0);
        //glColor3f(0,1,0);
        glVertex3f( 4.0, 0,-4.0);
        //glColor3f(0,0,1);
        glVertex3f(-4.0, 0,-4.0);
```

Autos

Name	Value	Type
⊞ white	0x0012fe28	float [4]
white[4]	1.744e-039#DEN	float

Call Stack

Name	Lang
MyApp.exe!drawScene() Line 11	C++
MyApp.exe!display() Line 45	C++
glut32.dll!100050c2()	
glut32.dll!100048d6()	
MyApp.exe!main(int argc=1, char * * arg	C++
MyApp.exe!mainCRTStartup() Line 259 +	C
kernel32.dll!7c816fd7()	

The values of the most relevant variables are shown at tab “Autos” here. Note that the values can be modified directly by typing new values.

The values of all variables are shown at tab “Autos” here:

Summary of most common functions:

F9	– sets/removes a break point at the location of the cursor
F10	– steps one line in the source code
F5	– continues the running of the program
Shift + F5	– stops debugging
F11	– steps into a function call if a function call is made at that line and source code for the function is available
Shift + F11	– steps out from a function call

Miscellaneous about C++

Typically you store the class definition in a class_name.h file and the implementation of the methods (functions) in a class_name.cpp file.

The class definition in the .h-file looks like this:

```cpp
// The #ifndef avoids the problems with multiple includes of the
// same file. Otherwise, the compiler may complain that the class
// is already defined.
#ifndef YOUR_CLASS_NAME_AND_SOMETHING_H
#define YOUR_CLASS_NAME_AND_SOMETHING_H

class Example
{
public:
      // Regarding speed it is good to inline the
      // constructors if they are short. Here they are empty.
      Example() {};

      // And the same applies for destructors
      ~Example() {};

      bool  Method1();

private:
      int m_a;
};
#endif //YOUR_CLASS_NAME_AND_SOMETHING_H
```

The class_name.cpp file looks like this:

```cpp
#include "class_name.h"
#include <stdio.h>

bool Example::Method1()
{
      printf("Hi");
      return true;
}
```