Compsci.373 Tutorial 5

COMPUTER GRAPHICS I

Welcome back!

Welcome back from the break, and welcome to the next section of the course!

My name is Nick, and I'll be your tutor for the Computer Graphics section of Compsci 373

- **Tutorial sections**
  - Tutorial 5: Monday 20th April to Friday 24th April
  - Tutorial 6: Tuesday 28th April to Monday 4th May
  - Tutorial 7: Tuesday 5th May to Monday 8th May

  Note that the tutorial sections after this one will start from Tuesday next week onwards (this is due to Anzac day)

Tutor contact details

- **Tutors:**
  - Nick: nsto032@aucklanduni.ac.nz
    - Office hours: Monday to Wednesday 3pm-4pm
    - City campus, Building 303S, Area outside 396
  - Trevor: tgee862@aucklanduni.ac.nz
    - Office hours: Monday 11am-12pm, Tuesday 2pm-3pm, Wednesday 1pm-2pm
    - City Campus, Building 303S, Area outside 396

Come see us (during office hours) or send us an email if you have any issues! ☺

Today's outline

- A look at the next few assignments involving textured 3D graphics using raycasting techniques
- Setting up Visual Studio
- Setting up a window and drawing to the screen

What is 3D computer graphics?

Computer graphics is the practice of using a computer to display visual, non textual data.

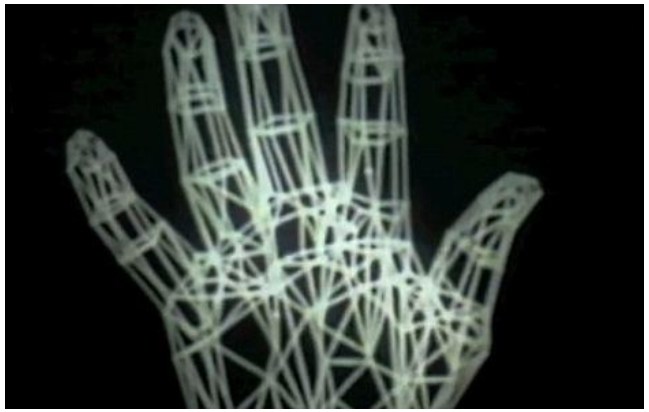
You're looking at computer graphics right now!

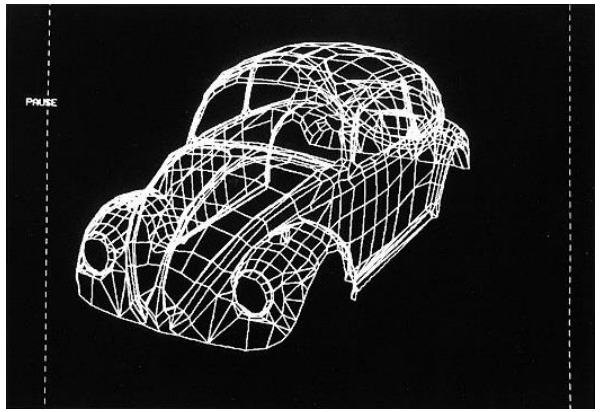
3D Computer graphics is the practice of generating seemingly 3-Dimensional images with a computer, and it will be the focus for this part of the course.

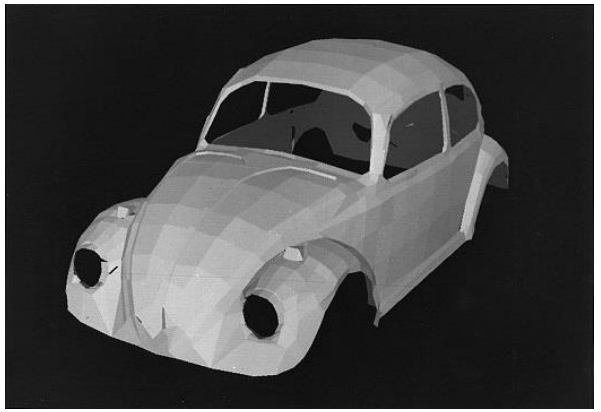
What is 3D computer graphics?

3D computer graphics have been around since the early 1970s, but were mostly non real-time up until the early 80s.

Single frames for simple wire frame scenes, such as those below, could take many seconds, or even minutes or hours to generate!



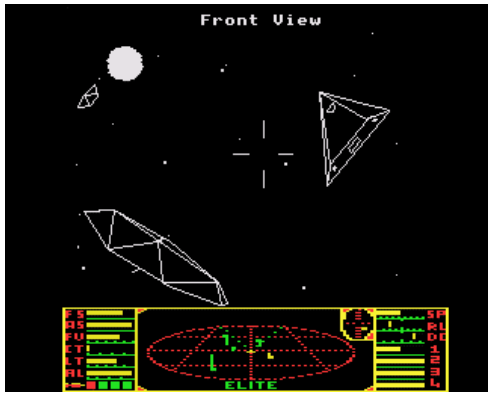




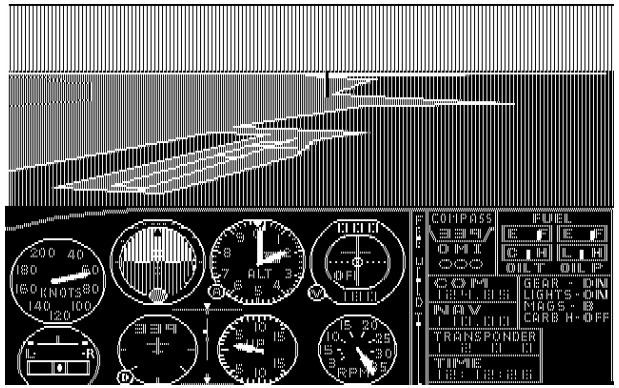
A selection of 3D renderings (1972-1977)

What is 3D computer graphics?

During the 80s, computers started to get powerful enough to be able to render frames in a fraction of a second. This was quickly picked up by game developers.

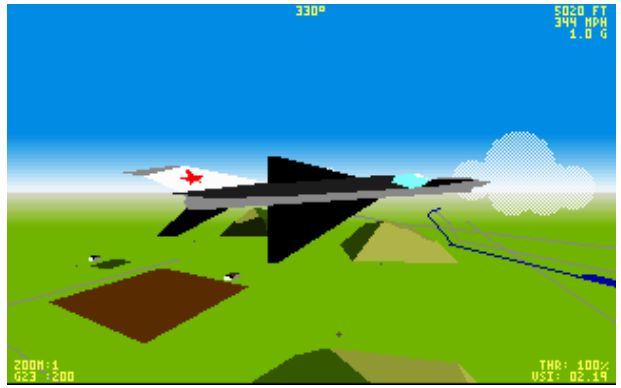


Elite (1982)



Microsoft Flight Simulator (1983)

Vette! (1989)



Chuck Yeager's Air Combat (1991)

Our focus for the next few weeks

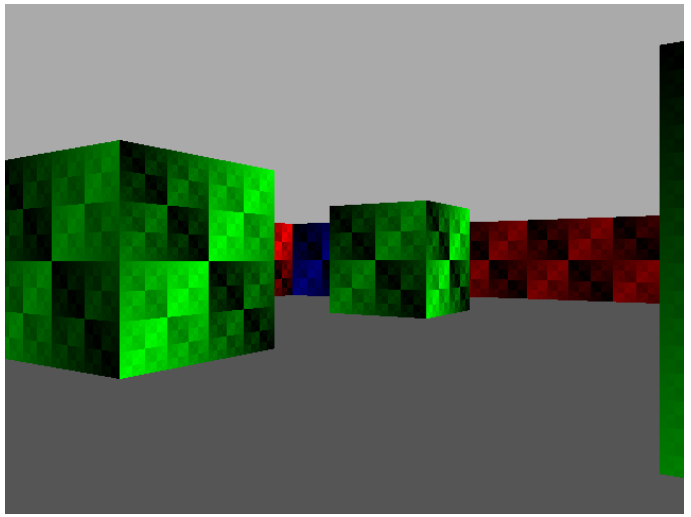
For the next few weeks, we will be focusing one particular 3D rendering technique popularised in 1992 by id Software, and used in numerous games of the time (Blake Stone, Corridor 7, Rise of the Triad).

Most notably, Wolfenstein 3d®:

Lets take a look at it!

What you will be doing

You will be working on a very similar 3D renderer. By the end of the next few weeks, you should be able to make a nearly identical renderer, including texture mapping:



A texture-mapped raycasted scene

Lets take a look at it!

Getting Visual Studio set up

For the computer graphics assignments, you will be writing your code in C. Because most of you will probably be using Windows, we will take a look at how to set up the programming environment for the assignments in Visual Studio (Available free to you as students).

Instructions on downloading Visual Studio will be made available on the resources page of the course website.

Loading a sample project

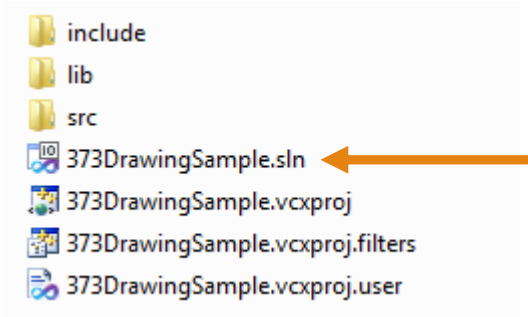
Assuming you have Visual Studio set up (any version after and including VS 2010 is okay), you will be ready to start writing graphics programs.
If you can't install it on your own computer, that's okay, you should be able to use the lab computers.

To make getting started easier, we are going to provide a simple library which provides functions to draw simple 2D shapes (lines and rectangles), as well as give access to individual pixels on the screen.

Loading a sample project

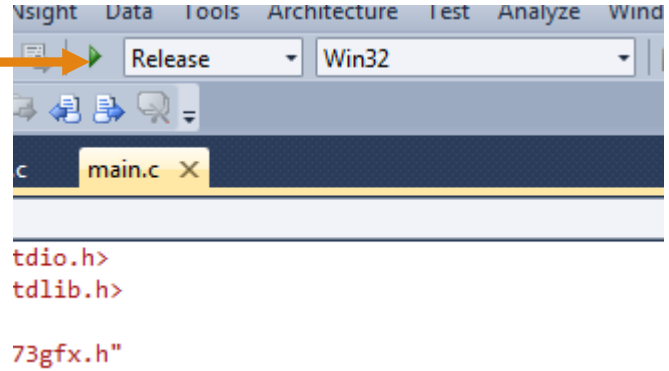
Lets load a simple project which will display a blank window for us.

Loading and running a template project



Open this first

Then click this to run the program



Drawing to the screen

The graphics library provided for the course includes a few basic functions for drawing shapes to the screen. They operate very similarly to the functions in Java's Graphics library:

```
void setDrawColor(int r, int g, int b, int a);
void drawLine(int x1, int y1, int x2, int y2);
void fillRect(int x, int y, int w, int h);
void drawRect(int x, int y, int w, int h);
```

Drawing to the screen

```
void setDrawColor(int r, int g, int b, int a);
```

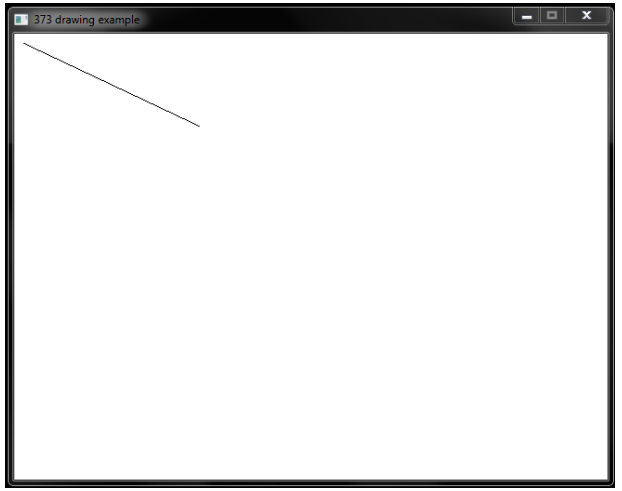
This function allows you to set the current drawing color (as an RGB vector). The first three arguments specify the red, green, and blue components (0 – 255). The last specifies the alpha (transparency) component, which should generally always be set to 255.

Drawing to the screen

```
void drawLine(int x1, int y1, int x2, int y2);
```

This function allows you draw a line to the screen. The first two parameters specify the x and y location of the start of the line, the last two parameters specify the x and y location of the end of the line.

Example for drawLine(10,10,200,100):



Drawing to the screen

```
void fillRect(int x, int y, int w, int h);
```

This function allows you draw a solid rectangle to the screen. The first two parameters specify the top left x,y coordinate of the rectangle, the last two parameters specify the width and height of the rectangle.

Example for fillRect(10,10,100,200):

Drawing to the screen

```
void drawRect(int x, int y, int w, int h);
```

This function is identical to fillRect(), except that it draws the outline of a rectangle.

Example for drawRect(10,10,20,20):

Drawing to the screen

The graphics library also includes two extra functions. One for clearing the screen, and one for refreshing the screen (required after having drawn to the screen):

```
void clearRenderer();
void presentRenderer();
```

Important note: clearRenderer() will fill the window with the last color that was set using setDrawColor().

An example

Once you have a window, drawing a simple shape such as a square is very simple.

In order to set the drawing color, setDrawColor() is provided. Once a color has been set, every shape drawn after that will have that color, until setDrawColor() is called again.

An example: drawing some squares

For example, this is the code for drawing a red square:

```
setDrawColor(255, 0, 0, 255);
fillRect(20, 20, 50, 50);
```

And this is the code to draw a green square:

```
setDrawColor(0, 255, 0, 255);
fillRect(20, 70, 50, 50);
```

An example: drawing some squares

Remember, before drawing to the screen, you should clear it so that any shapes from the previous draw are removed. This is especially important for animation. After drawing the shapes, you also will need to refresh the screen.

Here is full code for drawing a red square above a green square:

```
clearRenderer();
setDrawColor(255, 0, 0, 255);
fillRect(20, 20, 50, 50);
setDrawColor(0, 255, 0, 255);
fillRect(20, 70, 50, 50);
presentRenderer();
```

Clear the window

Set the drawing color to r=255, g=0, b=0, a=255

Draw a filled 50x50 pixel square at x=20, y=20

Set the drawing color to r=0, g=255, b=0, a=255

Draw a filled 50x50 pixel square at x=20, y=70

Refresh the window

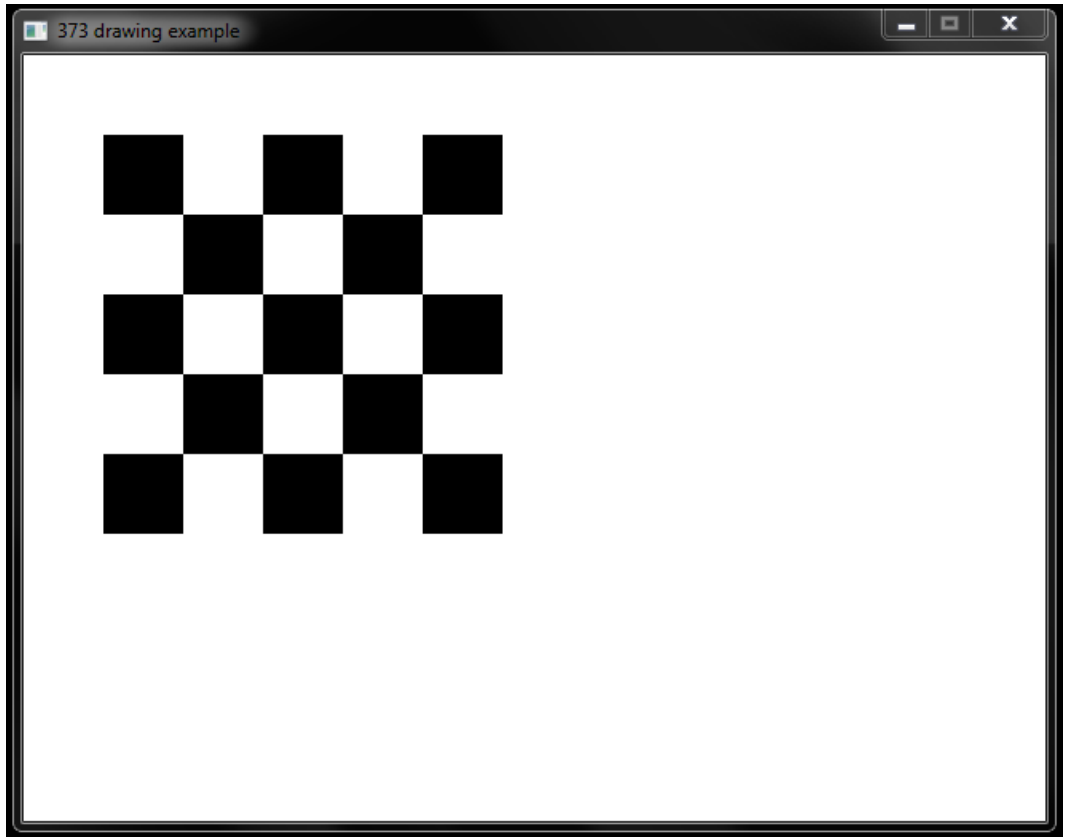
Here's what that looks like

An example: drawing a checkerboard pattern

```
int row, col;
int i = 0;
clearRenderer();
for (row = 0; row < 5; row++) {
    for (col = 0; col < 5; col++) {
        if (i % 2 == 0)
            setDrawColor(0, 0, 0, 255);
        else
            setDrawColor(255, 255, 255, 255);

        fillRect(20 + col*20, 20 + row*20, 20, 20);
        i++;
    }
}
presentRenderer();
```

Here’s what that looks like



That’s all for today!