Xi'an Jiaotong-Liverpool University
西交利物浦大学

CPT205 Computer Graphics

General Introduction Hardware and Software

Lecture 01
2022-23

Yong Yue

What is Computer graphics?

‘Computer Graphics’ is concerned with all aspects of producing pictures or images using a computer. There are three closely related meanings, each representing a different perspective on the same thing.

- the images that you see on the computer screen
- the computer code that is used to create the images
- a mathematical model of the real-world (which is sometimes called the virtual world)

When working at the most advanced levels of Computer Graphics, the computer graphics specialist will

- create a virtual world
- implement the virtual world in computer code
- run the code to see life-like images on the computer screen

What are the application areas?

The ‘application areas’ are the different types of computer application that can be produced using computer graphics technology. They are also the technical areas within which a student may seek to work when he or she leaves university with a relevant degree.

Example application areas:

- Display of information
- Design
- Simulation/modelling and animation
- User interfaces
- Virtual reality

Why learn computer graphics?

There are numerous good reasons. These include:

- the application areas are exciting;
- the subject matter is intellectually stimulating and the knowledge that you will get on the course is relatively rare;
- computer graphics is a major facet of computer science (e.g. try to think of a computer program that does not involve some kind of graphics);
- the concepts associated with computer graphics are time independent (i.e. what you learn now will still be valid in the future, unlike some programming languages which disappear with passing time);
- computer programming is the type of job in computing that is in great demand, and the computer graphics course involves a good deal of “hands on” programming.

Basic concepts of computer graphics

- Graphics hardware and software
- Fundamental mathematics
- Objects / geometric primitives – point, curve, surface and solids
- Modelling and representation schemes
- Geometric transformations
- Viewing and projections
- Clipping
- Removal of hidden curves and surfaces
- Lighting and materials
- Texture mapping
- Animation
- Programming and applications

This module

Introduces a wide range of topics in computer graphics and its applications, providing you with both fundamental theory and hands-on experience through lab-based practice and assessment. It follows a standard textbook with additional materials used for contemporary developments and applications.

In terms of learning outcomes, you will be able to:

- demonstrate a good understanding of topics and applications in computer graphics covered in the module;
- demonstrate an in-depth knowledge of geometric creation and transformation, projection, clipping and hidden geometry removal, lighting and materials, and texture mapping;
- apply relevant techniques / algorithms covered in the module to specific scenarios;
- write programming code in conjunction with a popular graphics platform (e.g. OpenGL).

Delivery and assessment

Delivery: You will have a two-hour formal lecture followed by a two-hour lab weekly. It assumes knowledge of matrices and vectors and previous experience of computer programming in a high-level procedural language (e.g. Java or C). Sample programs will be provided during the lab sessions.

Assessment:

1) Assessment 1 (2D project – 15%): Out Week 4 and due Week 8
2) Assessment 2 (3D project – 15%): Weeks 9 to 13
3) Assessment 3 (Final exam – 70%): Early January 2023
4) Resit Assessment (Exam – 100%): Early August 2023

No resit is allowed for assessments 1 and 2. If a student does not obtain an overall grade of 40% or higher for Assessments 1-3, she/he will be required to take the resit exam which will have a weighting of 100% for the module (regardless of grades of assessments 1 and 2).

***Important**: Plagiarism is a serious academic offence and will not be tolerated. Copying from other sources without appropriate acknowledgement may result in plagiarism! If in doubt, consult relevant members of academic staff.*

Marking scheme of coursework

Category	Requirements (each category builds on the requirements for the preceding category)
First Class (≥70%)	Overall outstanding work. Very neat program implements effectively all the graphics techniques covered. Artefact produced with realistic / real-life content and visual effect. Well-structured and concise written report providing all the required information.
Second Upper (60 to 69%)	Comprehensive program that utilises effectively the full range of the graphics techniques covered to date. Good commenting and layout of the program. An impressive artefact produced with a good range of features achieved by calling appropriate OpenGL functions. A comprehensive and clear report containing all required information within the page limit.
Second Lower (50 to 59%)	Substantial working program implements a good range of graphics techniques. Nice layout and objects in the artefact. Written report contains all the information of the features and functions of the program including some screenshots.
Third (40 to 49%)	Working program that generates a recognisable artefact with some objects and a limited range of the graphics techniques utilised. Written report provides a good overview and describes all the basic information for the work completed.
Fail (0 to 39%)	Some code produced attempts to the use of some graphics techniques covered in the module. No or very limited artefact produced. Written report covers very limited number of the items required in the assignment brief, acknowledging properly sources used if any.
Non-submission	A mark of 0 will be awarded.

Schedule and Module Team

- **Delivery Schedule**

Lecture: *Tuesday 09:00-11:00*
Online (Week 2)
EE101 (Weeks 3-6 and 8-14)

Lab: *Wednesday 09:00-11:00*
Online (Week 2)
SD319 / SD546 / SD554 (Weeks 3-6 and 8-14)

- **Module Leader and Contact Details**

Name: *Yong Yue*
Email: *yong.yue@xjtlu.edu.cn*
Office telephone number: *(0512) 8816 1503*
Room number: *SD523*
Office hours: *16:00-17:00 Tuesday and 17:00-18:00 Wednesday*
Preferred means of contact: *email*

Schedule and Module Team

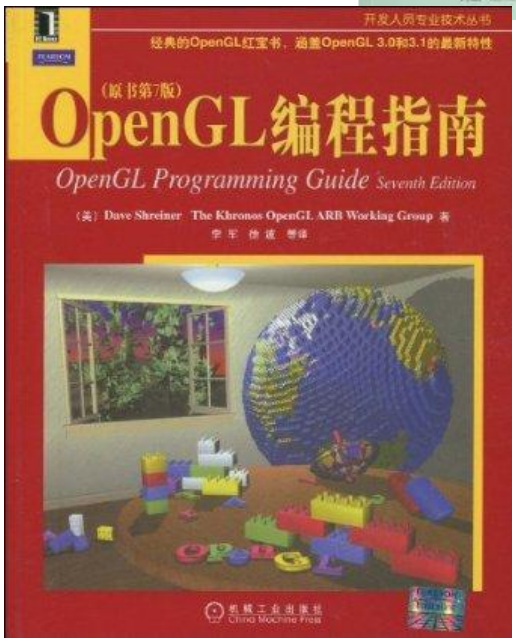
- **Teaching Assistants (TAs)**

  Lei Chen (lei.chen02@xjtlu.edu.cn) – Lead TA
  Shanliang Yao (shanliang.yao19@student.xjtlu.edu.cn)
  Yijie Chu (yijie.chu16@student.xjtlu.edu.cn)
  Ruben Ng (r.ng19@student.xjtlu.edu.cn)
  Fabiola Polidoro (f.polidoro18@student.xjtlu.edu.cn)
  Zhuoxiao Li (zhuoxiao.li21@student.xjtlu.edu.cn)
  Xiaoyu Huang (xiaoyu.huang20@student.xjtlu.edu.cn)
  Zitian Peng (zitian.peng20@student.xjtlu.edu.cn)
  Xingshuo Li (xingshuo.li17@student.xjtlu.edu.cn)

Depending on the demand for support later on, students may be divided into groups to receive support from dedicated TAs.

Textbooks

- Edward Angel and Dave Shreiner, *Interactive Computer Graphics: A Top-Down Approach with Shader-Based OpenGL*, Sixth Edition, ISBN: 978-7121177095, 1 July 2012, Electronic Industry Press, China.

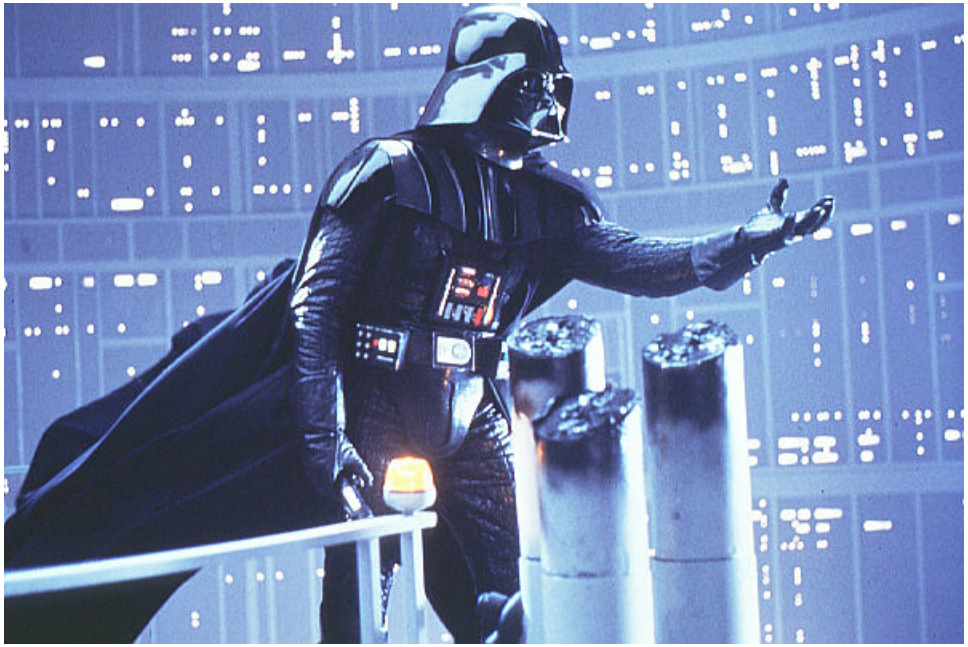


- Dave Shreiner, Graham Sellers, John M. Kessenich and Bill M. Licea-Kane, *OpenGL Programming Guide: The Official Guide to Learning OpenGL*, Versions 4.3, ISBN: 978-0321773036, 20 March 2013, Addison Wesley.

What's next?

To do well on the course:

- try to enjoy yourself
- attend all lecture and practical sessions
- work consistently



Hardware and Software

- Graphics Hardware
  - Input, Processing and Output Devices
  - Framebuffers
  - Pixels and Screen Resolution
- Graphics Software
  - Techniques (Algorithms, Procedures)
  - Programming Library / API (OpenGL, JOGL and so on)
  - Not our focus: High level Interactive Systems (Maya, Studio Max, Unity, AutoCAD and so on)

Thank about

- Basic
  - How will you define graphics hardware?
  - Does graphics hardware involve input, processing and output?
  - When you buy a computer how will you specify the graphics requirement?
- Intermediate
  - What is the graphics board on your home desktop computer?
  - How many lines/sec or triangles/sec can you draw on your computer?
  - Screen resolution: How many pixels do you see on your screen?
- Advanced
  - How do you represent a scene (a house) graphically in your computer?
  - How do you display the house on the computer screen (framebuffer)?
  - How does the graphics hardware transform your scene to the display?

Graphics devices

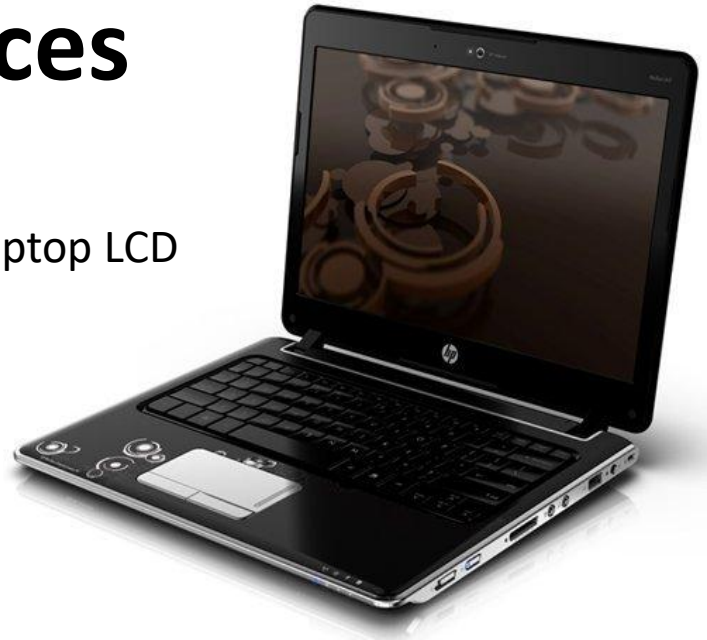
- Input Devices
- Processing Devices
- Output Devices

How do we link the three?

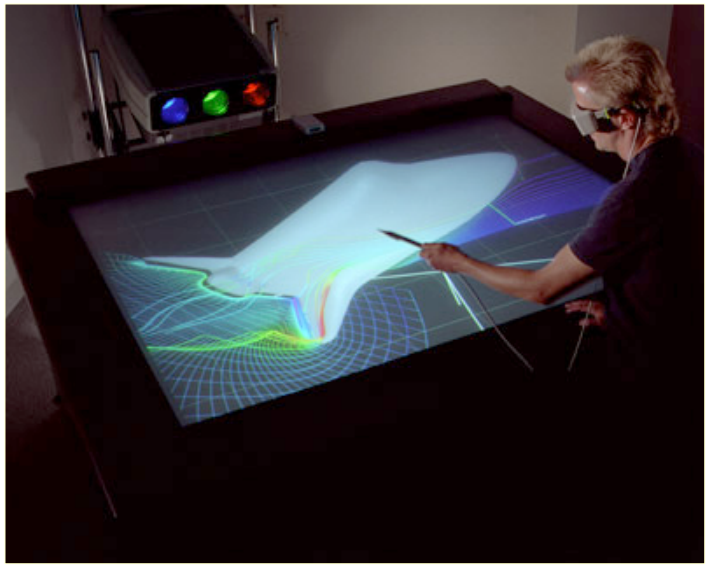
Output devices

CRT

Laptop LCD



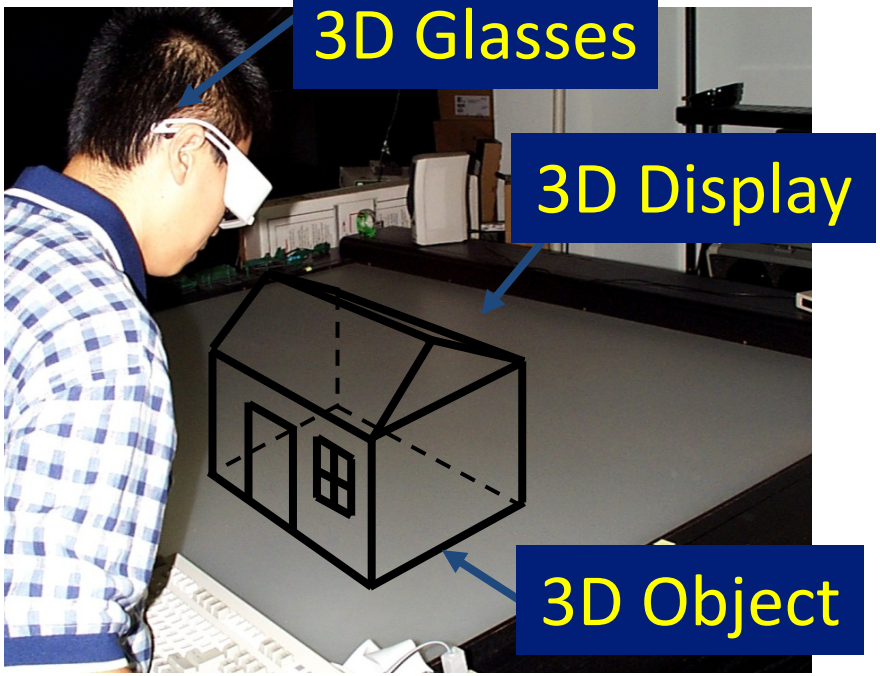
LCD

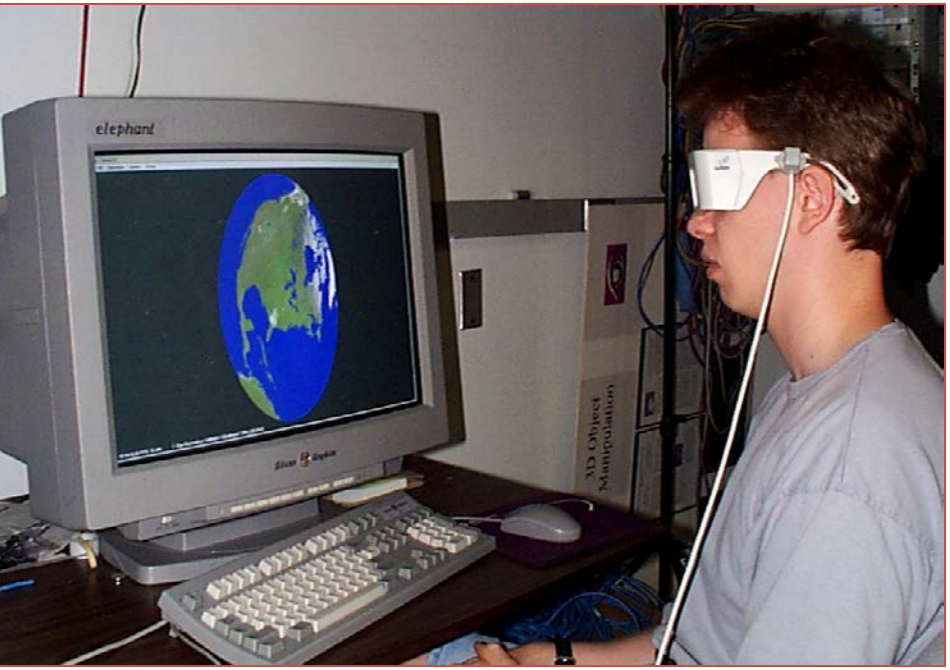


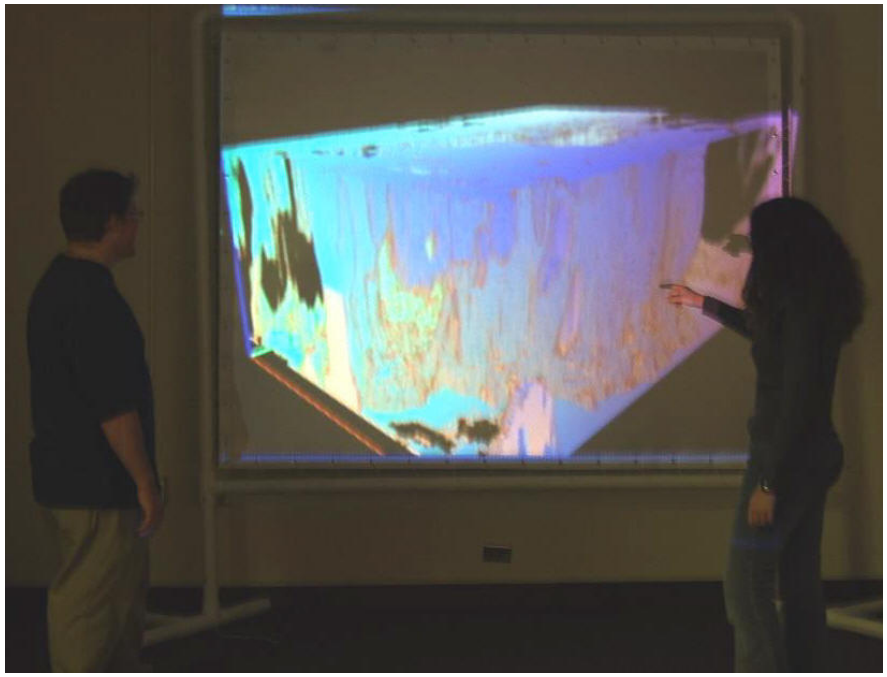
Projection Table

Graphic display in virtual reality

- Head-Mounted Displays (HMDs)
  - The display and a position tracker are attached to the user's head
- Head-Tracked Displays (HTDs)
  - Display is stationary, tracker tracks the user's head relative to the display.
  - Example: CAVE, Workbench, Stereo monitor


3D Glasses
3D Display
3D Object


elephant
3D Object Manipulation



Graphic processing unit (GPU)

- Graphics Boards / GPUs sit inside

Input devices

- Enables graphical interaction

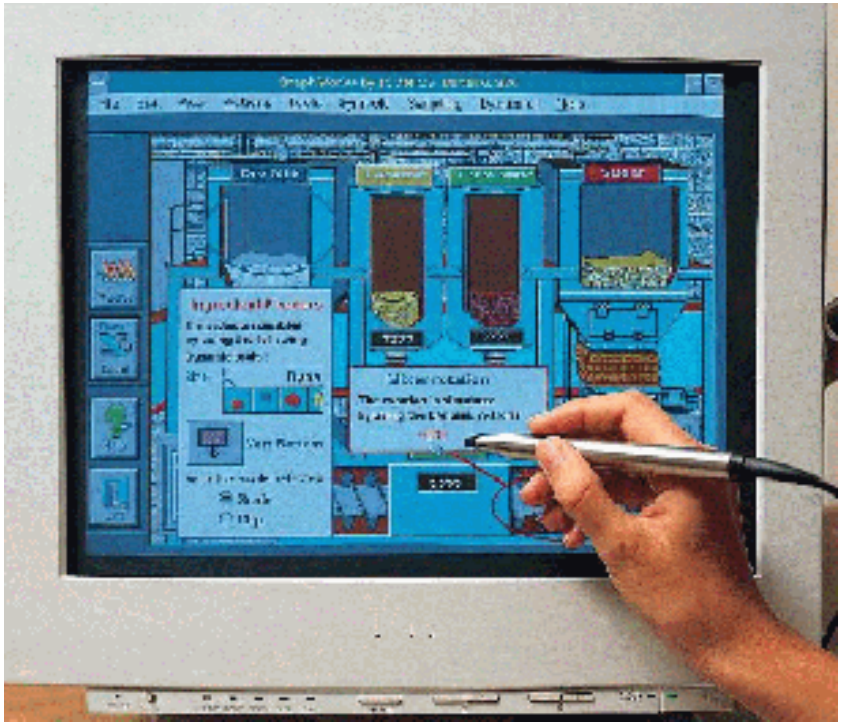
Input devices

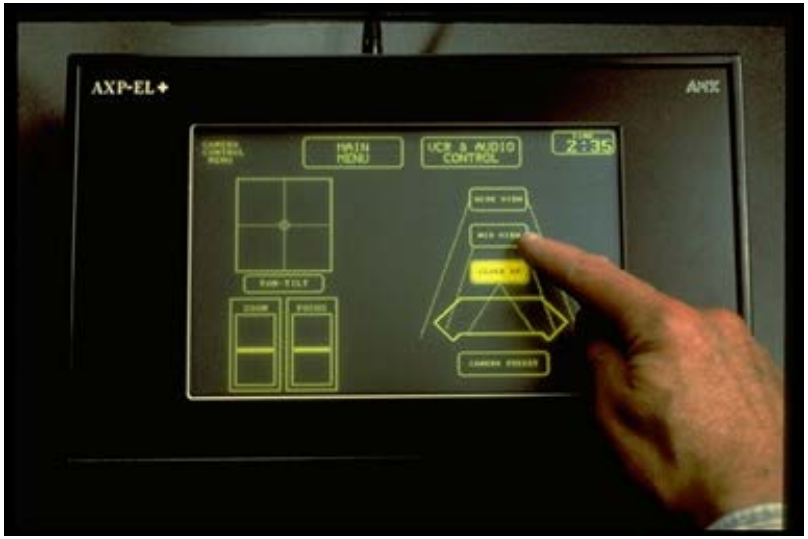
When queried, locator devices return a position and/or orientation

- Tablets
- Virtual Reality Trackers
  - Data Gloves
  - Digitisers

Input devices

- Light Pens
- Voice Systems
- Touch Panels
- Camera/Vision based
- Which is best?





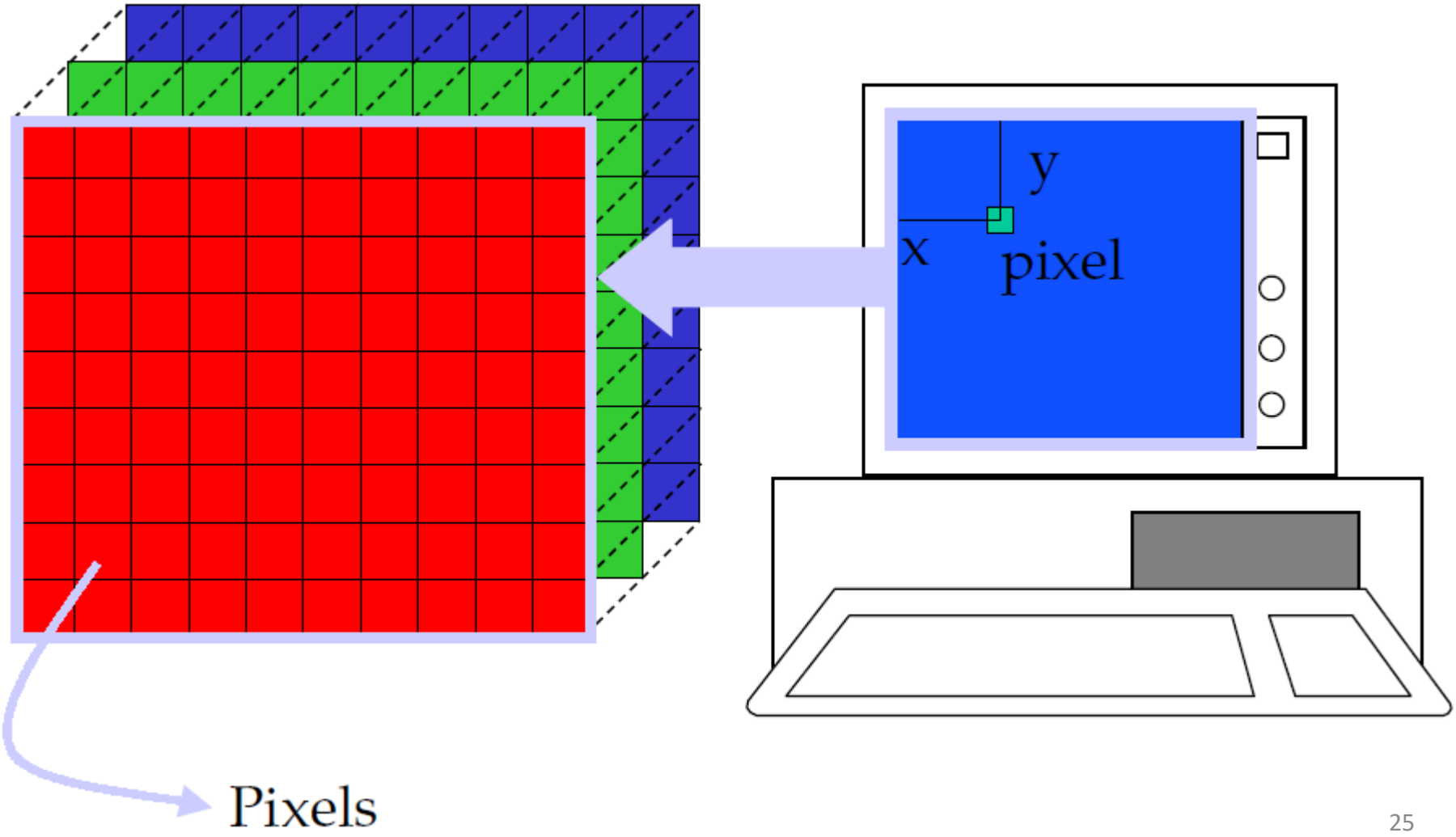
Input devices for games

Framebuffer

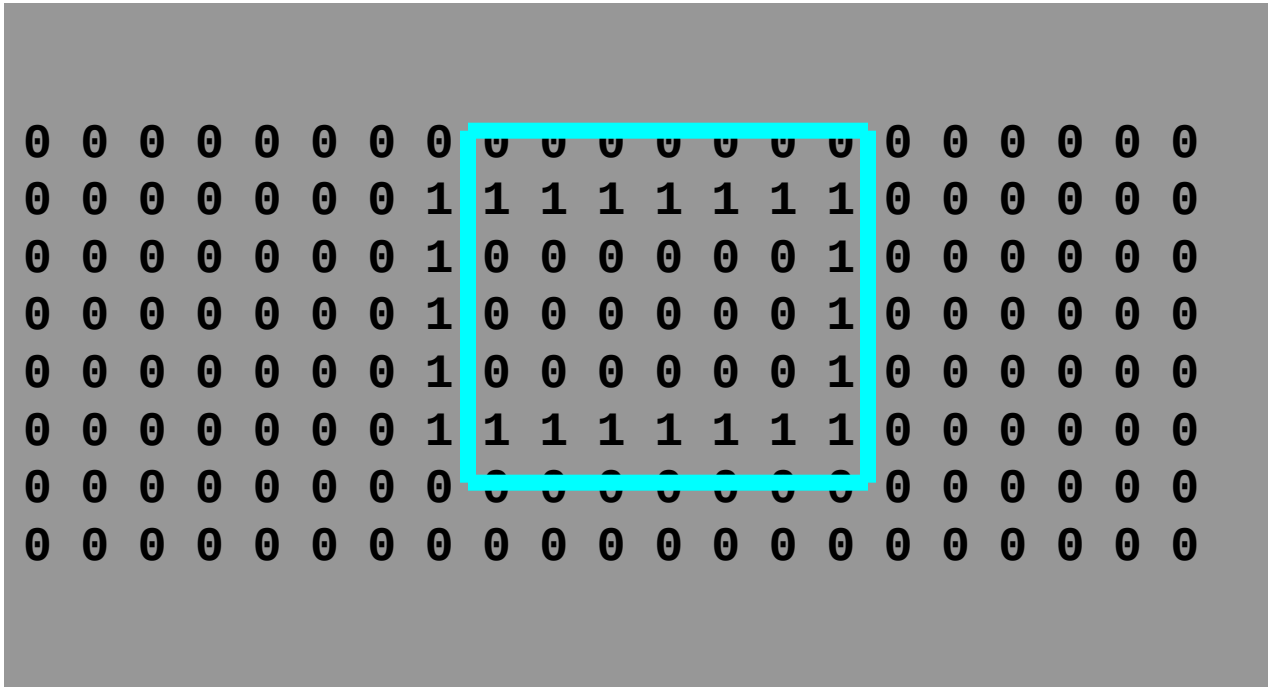
Framebuffer – A block of memory, dedicated to graphics output, that holds the contents of what will be displayed.

Pixel - an element of the framebuffer.

Framebuffer



Framebuffer



Questions: How many pixels are there?

What is the largest image you can display?

How big is the framebuffer?

How much memory do we need to allocate for the framebuffer?

Framebuffer in memory

- If we want a framebuffer of 640 pixels by 480 pixles, we should allocate:

  framebuffer = 640*480 bits

- How many bits should we allocate?

  Q: What do more bits get you?

  A: More values to be stored at each pixel.
  Why would you want to store something other than a 1 or 0?

Framebuffer bit depths

- How many colours does 1 bit get you?
- How many colours do 8 bits get you?
  - Monochrome systems use this (green/grey scale)
- What bit depth would you want for your framebuffer?

bit depth - number of bits allocated per pixel in a buffer.

Framebuffer bit depths

- Remember, we are asking “how much memory we allocate to store the colour at each pixel?”
- Common answer:
  - 32 bits RGBA

Framebuffer bit depths

- 32 bits per pixel (true colour)
  - 8 bits for red, green, blue and alpha
  - potential for 256 reds, greens and blues
  - total colours: 16,777,216 (more than the eye can distinguish)

- Let’s look at Display Control Panel

Data type refresher

bit - a 0 or 1. Can represent 2 unique values

byte - 8 bits or 256 values

word - 32 bits or 4,294,967,296 values

int - 32 bits

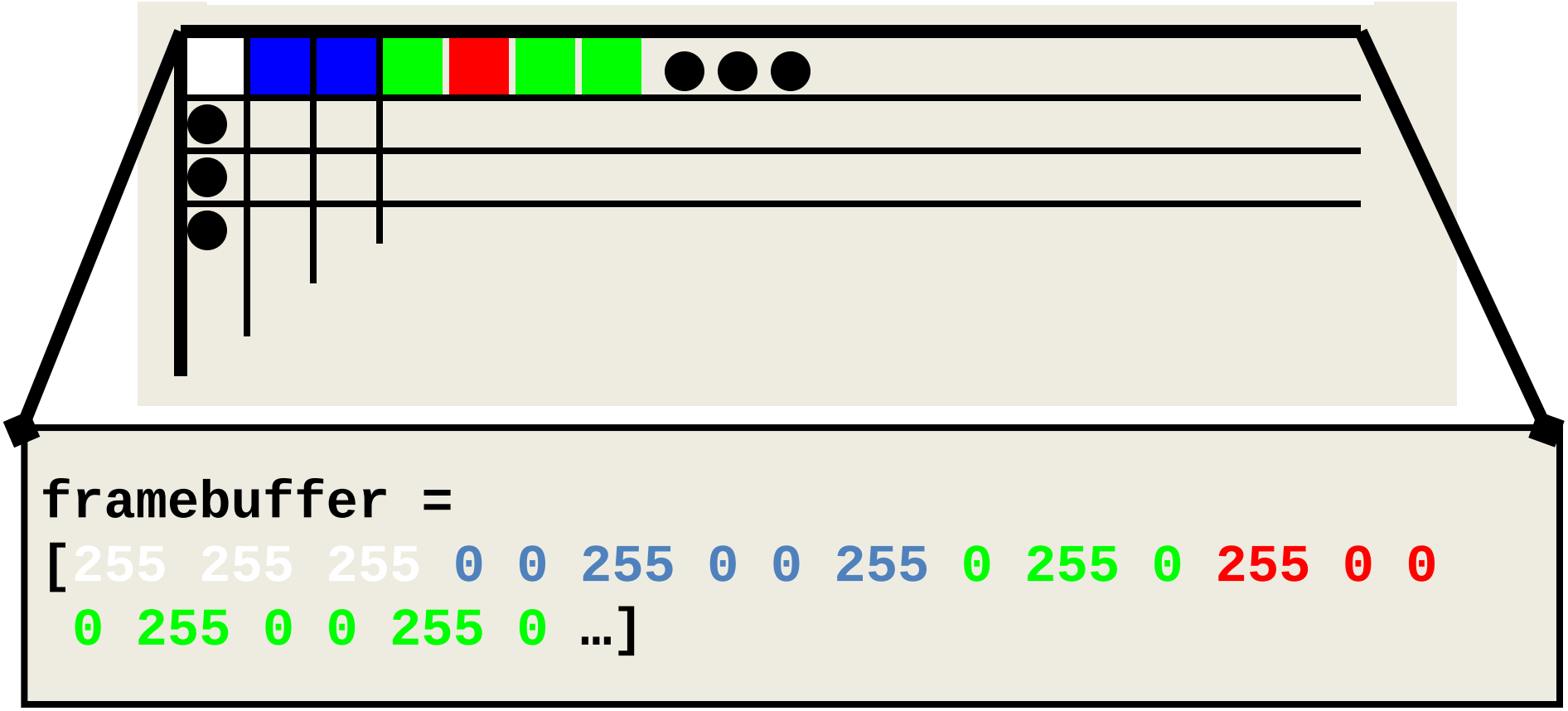
float - 32 bits

double - 64 bits

unsigned byte - 8 bits

Framebuffer bit depths

unsigned byte framebuffer [640*480*3];



```
framebuffer =
[255 255 255 0 0 255 0 0 255 0 255 0 255 0 0
 0 255 0 0 255 0 …]
```

Graphic card memory

- How much memory is on our graphic card?
  - 640 * 480 * 32 bits = 1,228,800 bytes
  - 1024 * 768 * 32 bits = 3,145,728 bytes
  - 1600 * 1200 * 32 bits = 7,680,000 bytes

- How much memory is on your graphics card?

Framebuffer -> monitor

The values in the framebuffer are converted from a digital (1s and 0s representation, the bits) to an analog signal that goes out to the monitor. This is done automatically (not controlled by your code), and the conversion can be done while writing to the framebuffer.

Image quality issues

- Screen resolution
- Colour
- Refresh rate
- Brightness
- Contrast
- Sensitivity of display to viewing angle

Pixels

- Pixel - The most basic addressable image element in a screen
  - CRT - Colour triad (RGB phosphor dots)
  - LCD - Single colour element
- Screen Resolution - measure of number of pixels on a screen (m by n)
  - m - Horizontal screen resolution
  - n - Vertical screen resolution

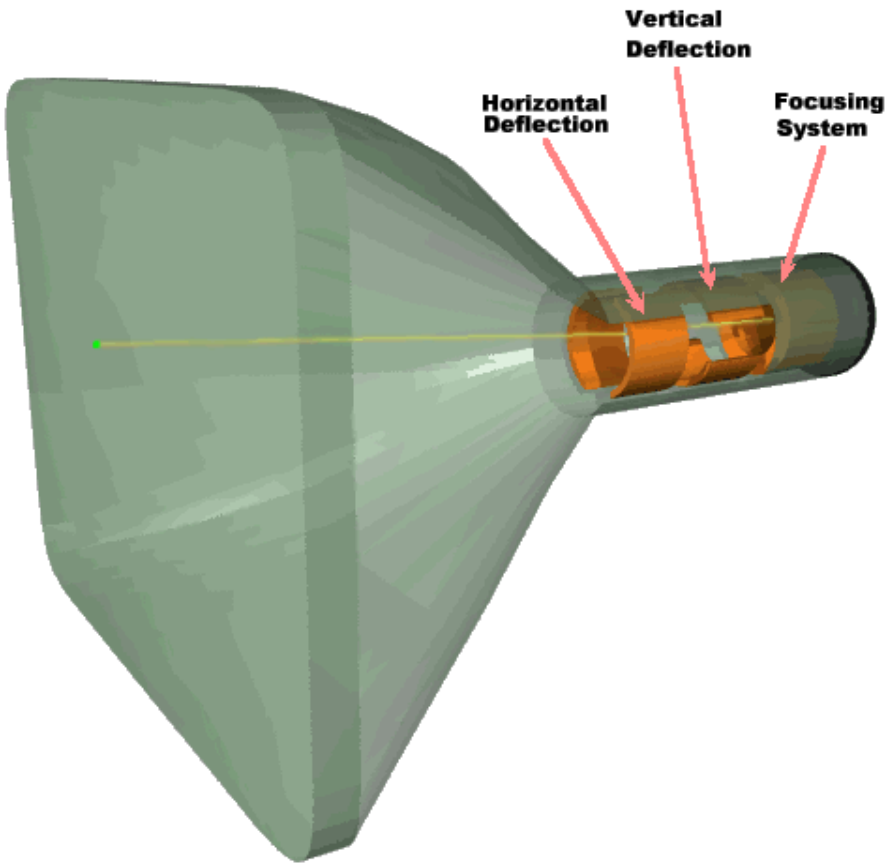
Video formats

- NTSC - 525x480, 30f/s, interlaced
- PAL - 625x480, 25f/s, interlaced
- VGA - 640x480, 60f/s, non-interlaced
- SVGA - 800x600, 60f/s non-interlaced
- RGB - 3 independent video signals and synchronisation signal, vary in resolution and refresh rate
- Interlaced - scan every other line at a time, or scan odd and even lines alternatively; the even scan lines are drawn and then the odd scan lines are drawn on the screen to make up one video frame.

Raster display

- Cathode Ray Tubes (CRTs), most “tube” monitors you might see. Used to be very common, but big and bulky.
- Liquid Crystal Displays (LCDs), there are two types transmissive (laptops, those snazzy new flat panel monitors) and reflective (wrist watches).

Cathode ray tubes (CRTs)



Heating element on the yolk.

Phosphor coated screen

Electrons are boiled off the filament and drawn to the focusing system.

The electrons are focused into a beam and “shot” down the cylinder.

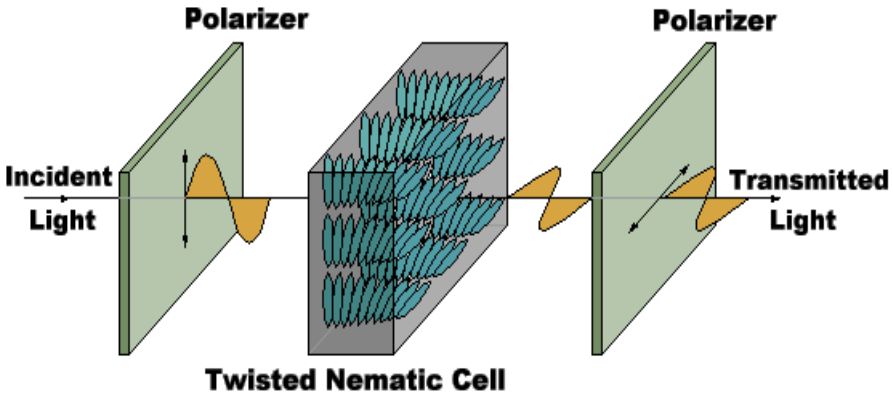
The deflection plates “aim” the electrons to a specific position on the screen.

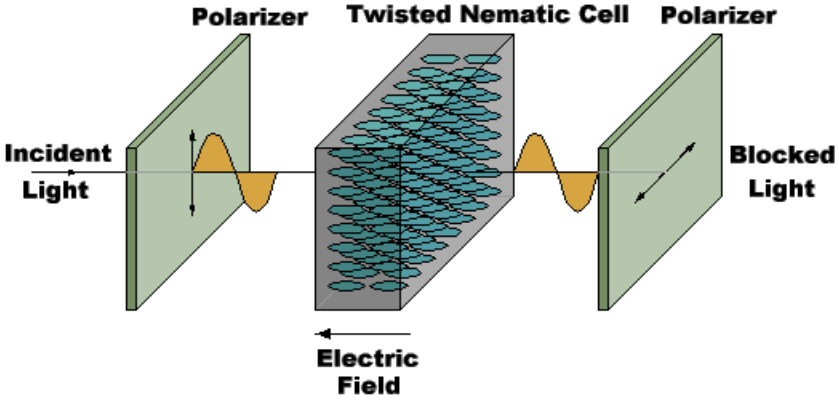
Cathode ray tubes (CRTs)

- Strong electrical fields and high voltage
- Very good resolution
- Heavy, not flat

Liquid crystal displays (LCDs)

- Also divided into pixels, but without an electron gun firing at a screen, LCDs have cells that either allow light to flow through, or block it.





Liquid crystal displays (LCDs)

- Liquid crystal displays use small flat chips which change their transparency properties when a voltage is applied.
- LCD elements are arranged in an n x m array called the LCD matrix.
- The level of voltage controls grey levels.
- LCDs elements do not emit light; use backlights behind the LCD matrix.
- Colour is obtained by placing filters in front of each LCD element.
- Image quality dependent on viewing angle.

Advantages of LCDs

- Flat
- Light weight
- Low power consumption

What is graphics software?

- Graphics drivers
- Graphics libraries
- Graphics editors
- Geometric modellers
- VR modellers
- Games
- Scientific visualisation packages
- …

Graphics software

- How to talk to the hardware?

  Algorithms, Procedures, Toolkits & Packages

  (Low Level  High Level)

- Programming API (helps to program, for our labs)
  - OpenGL (our focus)
  - JOGL (Open GL for Java)
  - OpenCV
  - Direct X, …
- Special purpose software (not our focus)
  - Excel, Matlab, …
  - AutoCAD, Studio Max, Unity, Maya, …
  - Medical visualisation, modelling, …

OpenGL

- First introduced in 1992.
- The OpenGL graphics system is a software interface to graphics hardware (GL stands for Graphics Library).
- For interactive programs that produce colour images of moving three-dimensional objects.
- It consists of over 200 distinct commands that you can use to specify the objects and operations needed to produce interactive three-dimensional applications.

OpenGL

- OpenGL is designed as a streamlined, hardware-independent interface to be implemented on many different hardware platforms.
- Similarly, OpenGL does not provide high-level commands for describing models of three-dimensional objects.
- With OpenGL, you must build up your desired model from a small set of geometric primitives - points, lines, and polygons.
- With OpenGL, you can control computer-graphics technology to produce realistic pictures or ones that depart from reality in imaginative ways.
- OpenGL has become the industry standard for graphics applications and games.

Summary

- These are interesting times for computer graphics.
  - Commodity graphics cards are highly capable.
  - New algorithms, long-offline algorithms are becoming possible.
  - Hard to keep up, even for “experts”.
- What’s pushing the technology curve?
  - Games
  - Movies
  - Interactive graphics applications

Topics covered today and …

- Computer Graphics and its main topics
- Graphics Hardware
  - Input, Processing and Output Devices
  - Framebuffers
  - Pixels and Screen Resolution
- Graphics Software
  - Techniques (Algorithms, Procedures)
  - Programming Library / API (OpenGL, JOGL, OpenCV, DirectX)
  - Not our focus: High level Interactive Systems (Maya, Studio Max, Unity, AutoCAD, and so on)
- Think about …
  - What are possible bottlenecks in system performance of a graphics system?
  - How can you achieve realism in a graphics system?
  - How do we combine the software and hardware to maximise performance?