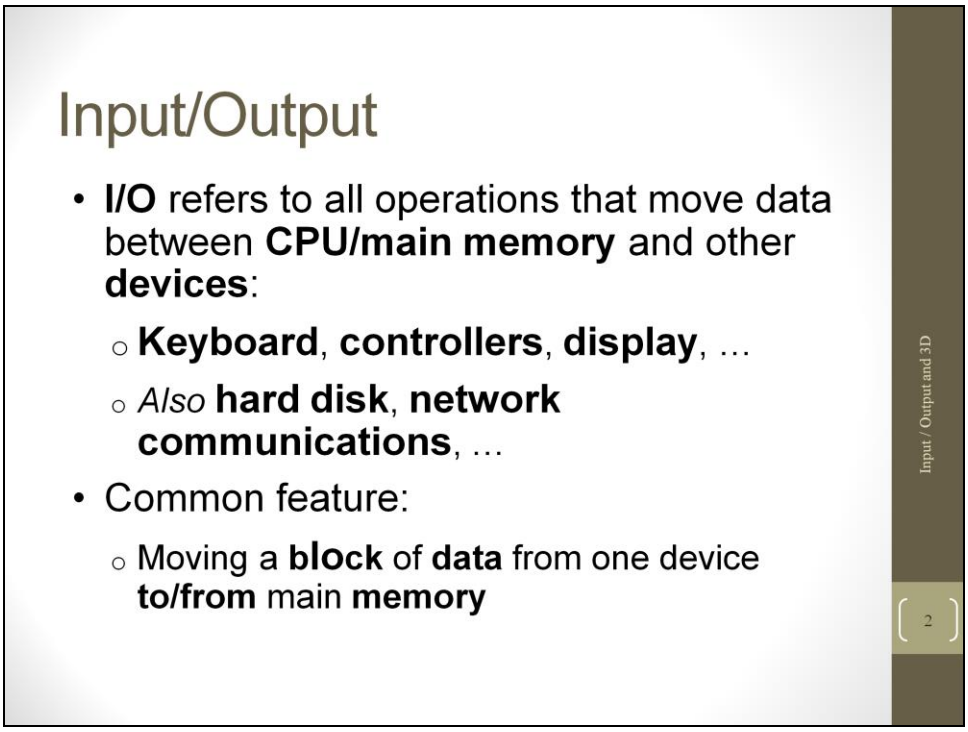
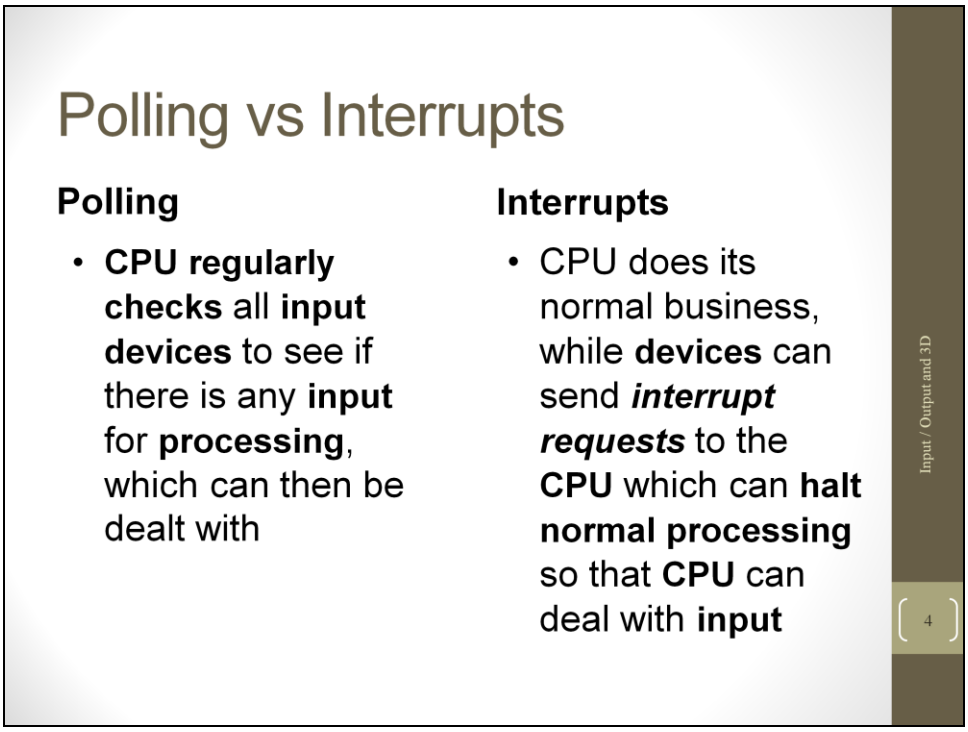

Computing Systems
Input Devices, 3D & Gaming
Input / Output and 3D
BY SA



Most of this lecture will focus on the input devices, but some more detail on I/O strategies will be included.

Keyboard input. The most basic common input device on modern computers.
But what happens when a key is pressed? What happens inside the computer and – how is the key press detected and reacted to?
The machines BIOS generates a *scan code* when they keyboard is pressed (a different code for each key as it is pressed down or released).
When a key is pressed an interrupt is generated telling the processor that there is a keyboard input waiting.
The CPU then can halt the current program,

and load an interrupt service routine (ISR) to deal with the input.
Once the ISR has completed, normal operation resumes

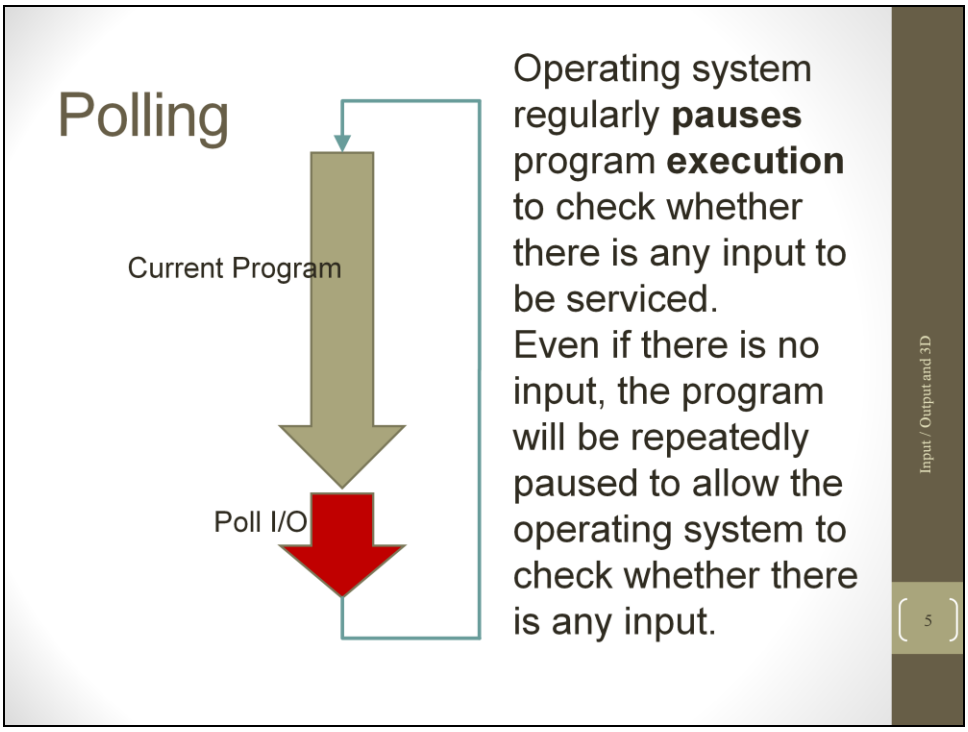


Two methods for dealing with input: Polling vs Interrupts
So if eg a calculation in under way, polling will pause the calculations at regular intervals to poll the input devices.
OR using interrupts, the calculation proceeds until an interrupt is received by the processor.

Interrupts
- CPU / hardware level
- Conceptually Related to Event driven programming

Polling

CPU/ hardware level
Conceptually related to Game/program loop
Aka Programmed I/O



From http://en.wikipedia.org/wiki/Polling_%28computer_science%29

Polling, or polled operation, in computer science, refers to actively sampling the status of an external device by a client program as a synchronous activity.

Polling is most often used in terms of input/output (I/O), and is also referred to as polled I/O or software driven I/O.

Polling is sometimes used synonymously with busy-wait polling (Busy waiting).

In this situation, when an I/O operation is required the computer does nothing other than check the status of the I/O device until it is ready, at which point the device is accessed.

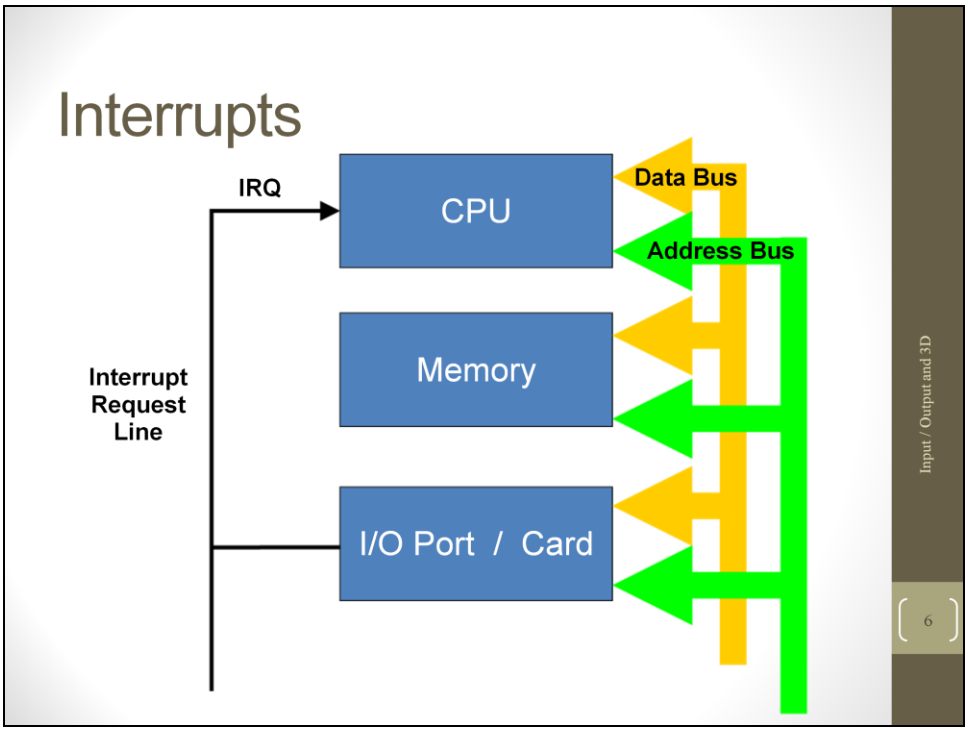
In other words the computer waits until the device is ready.

Polling also refers to the situation where a device is repeatedly checked for readiness, and if it is not the computer returns to a different task. Although not as wasteful of CPU cycles as busy-wait, this is generally not as efficient as the alternative to polling, interrupt driven I/O.

In a simple single-purpose system, even busy-wait is perfectly appropriate if no action is possible until the I/O access, but more often than not this was traditionally a consequence of simple hardware or non-multitasking operating systems.

Polling is often intimately involved with very low level hardware. For example, polling a parallel printer port to check whether it is ready for another character involves examining as little as one bit of a byte. That bit represents, at the time of reading, whether a single wire in the printer cable is at low or high voltage. The I/O instruction that reads this

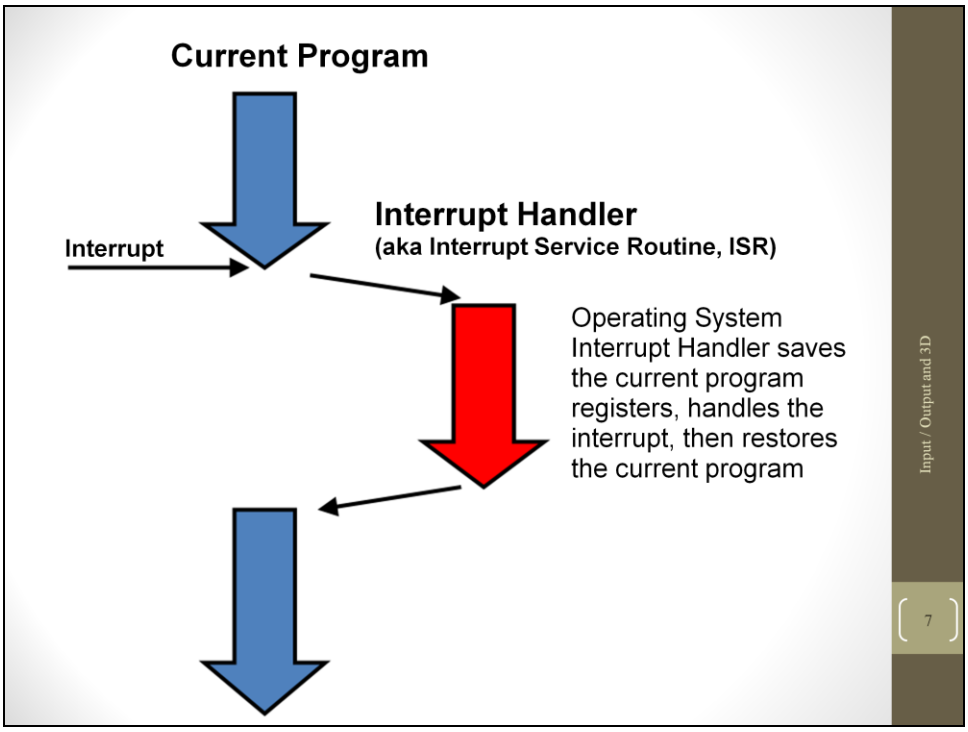
byte directly transfers the voltage state of eight real world wires to the eight circuits (flip flops) that make up one byte of a CPU register.



At any time during processing an interrupt can be sent to the CPU – this is via a separate connection from the normal data or address bus.

From http://en.wikipedia.org/wiki/Interrupt :
In computing, an **interrupt** is an asynchronous signal indicating the need for attention or a synchronous event in software indicating the need for a change in execution.
A *hardware interrupt* causes the processor to save its state of execution and begin execution of an interrupt handler. Software

interrupts are usually implemented as instructions in the instruction set, which cause a context switch to an interrupt handler similar to a hardware interrupt.
Interrupts are a commonly used technique for computer multitasking, especially in real-time computing. Such a system is said to be interrupt-driven.[1]
An act of *interrupting* is referred to as an interrupt request (IRQ).



Interrupt can cause CPU launch a handler routine (ISR) as soon as the current machine code instruction has finished execution. (CPU can also ignore an interrupt)

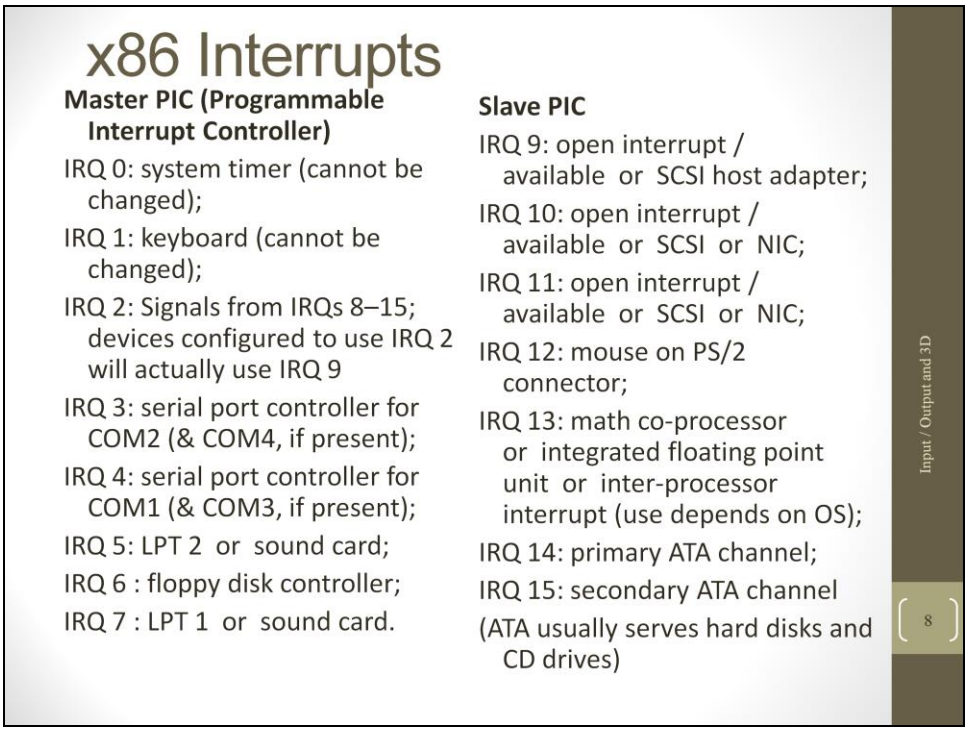
More complexity appears in the system to deal with multiple interrupts, e.g. to award priority for different interrupts or what happens when an interrupt arrives in the middle of dealing with a previous interrupt.

- e.g. if one interrupt is continuously being raised, this could prevent other interrupts or the main program from being executed.

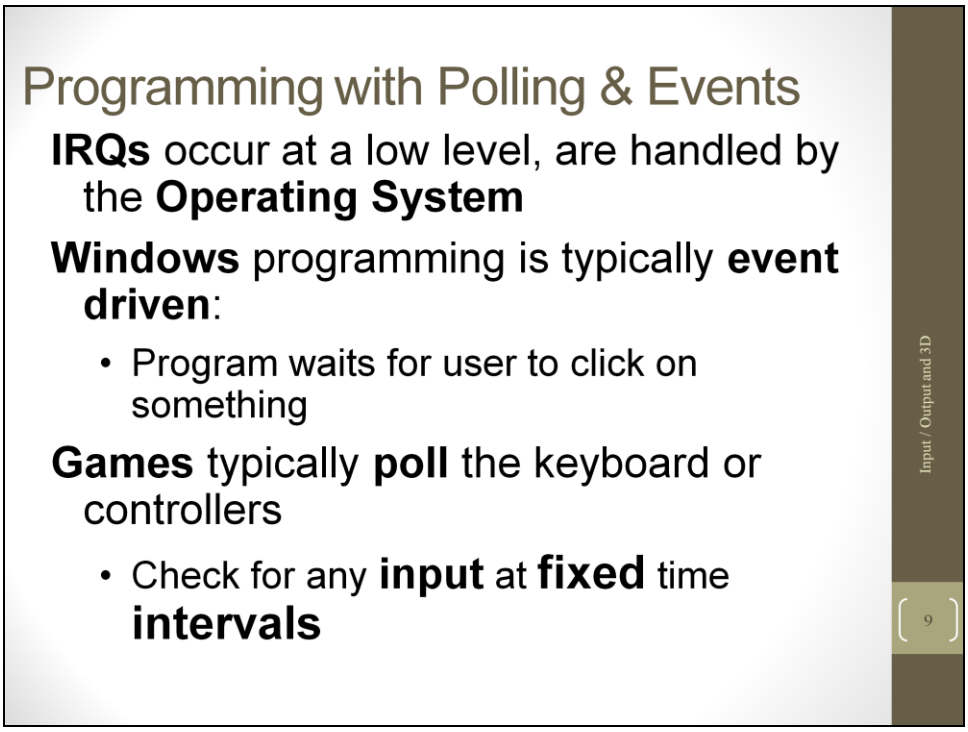
From http://en.wikipedia.org/wiki/Interrupt_handler

An **interrupt handler**, also known as an **interrupt service routine** (**ISR**), is a callback subroutine in an operating system or device driver whose execution is triggered by the reception of an interrupt. Interrupt handlers have a multitude of functions, which vary based on the reason the interrupt was generated and the speed at which the interrupt handler completes its task.

An interrupt handler is a low-level counterpart of event handlers. These handlers are initiated by either hardware interrupts or interrupt instructions in software, and are used for servicing hardware devices and transitions between protected modes of operation such as system calls.



PIC is the Programmable Interrupt Controller. Source: http://en.wikipedia.org/wiki/Interrupt_request. Note that many of the IRQs are fixed and based on legacy devices (e.g. LPT 2, COM3 etc) – these often refer to I/O ports that are no longer present on modern computers. May have other devices mapped to them. Modern OS and hardware supports more interrupts, but still will be limited in number and many will be hardwired to particular devices. Interrupt requests, shown above, it is dated.



Windows programs are event driven as user may click on any of a huge number of controls (buttons, scrollbars, etc) at any time. To poll these would be very wasteful – not least because such events are very infrequent, and the OS needs to respond in a timely fashion to any of these events

Games need to be able to respond rapidly but they also need to be able to draw complex scenes and model rich physical worlds without unexpected interruptions. Using polling, the programmer is able to control the amount of time spent on checking input. As long as input can be polled regularly, the program should remain responsive. Inputs will also occur frequently and regularly, so checking input regularly is not as wasteful as it might be.

Event Based programming

From http://en.wikipedia.org/wiki/Event_%28computing%29

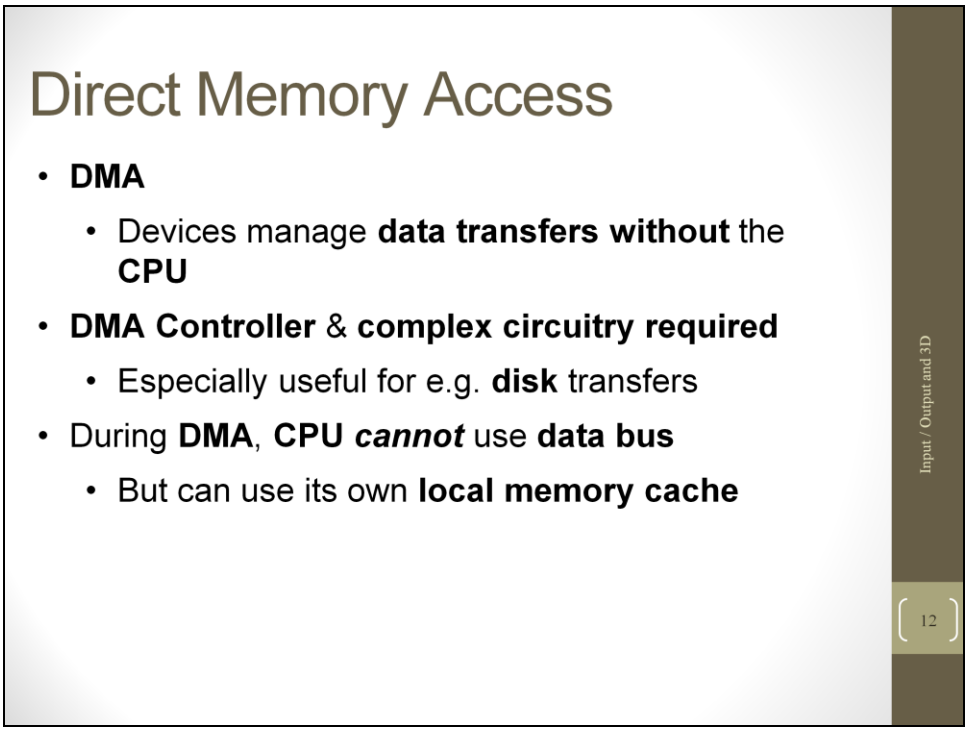
In computing an event is an action that is usually initiated outside the scope of a program and that is handled by a piece of code inside the program. Typically events are handled synchronous with the program flow, that is, the program has one or more dedicated places where events are handled. Typical sources of events include the user (who presses a key on the keyboard, in other words, through a keystroke). Another source is a hardware device such as a timer. A computer program that changes its behaviour in response to events is said to be event-driven, often with the goal of being interactive.

Windows Programming

- Windows programs are **event driven** as user may click on any of a huge number of **controls** (**buttons**, **scrollbars**, etc.) at ***any*** time.
- To **poll** these would be very **wasteful** – not least because such events are very **infrequent**, and the **OS** needs to respond in a **timely** fashion to any of these **events**.
- **Windows** uses ***interrupts***.

Why do games use polling?

- Games need to be able to respond **rapidly** but they also need to be able to draw complex scenes and model rich physical worlds ***without unexpected interruptions***.
- As long as input can be ***polled*** regularly, the program should remain ***responsive***.
- Inputs will also occur ***frequently*** and ***regularly***, so checking input regularly is not as wasteful as it might be.

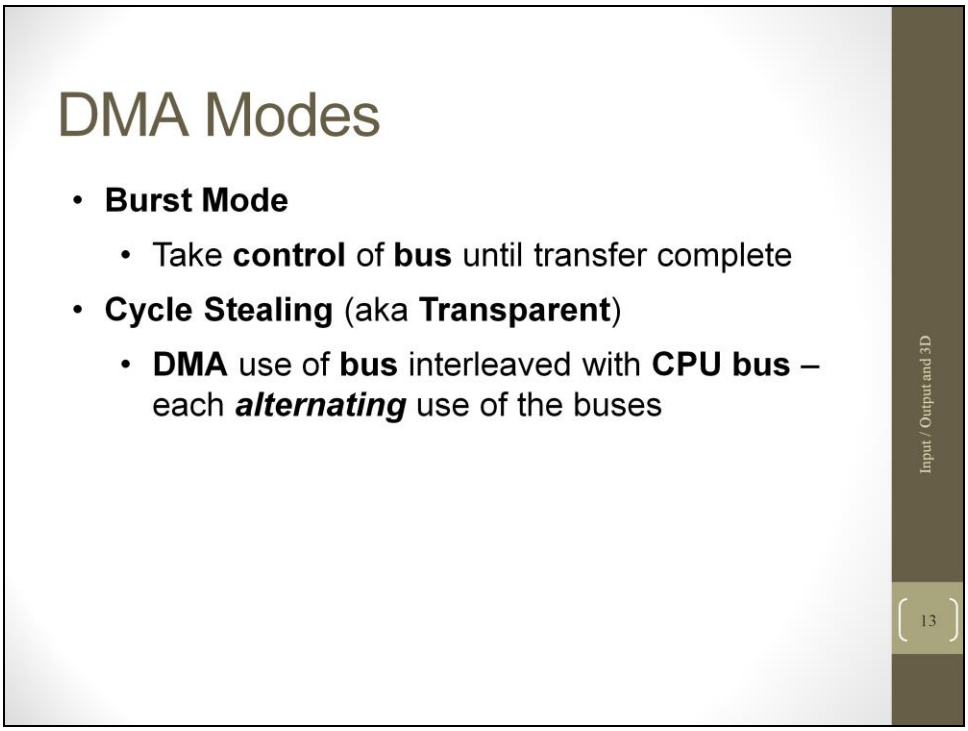


For good speed for I/O DMA is used. Devices can manage the data transfers without the CPU having to be in charge of moving data, this keeps the CPU busy. With DMA the CPU can start the transfer but does not have to be processing it, it can be doing it's own thing. Separate controller called the DMA controller with special circuitry, can handle this instead.

DMA also used in modern multiprocessor architectures as one (complex) example:

In e.g. The PS3 cell, there are 8 processing elements (1 PPU "Power Processing Element"

+ 7 SPU “Synergistic Processing Element”) in the CPU chip. This eight core processor allows different processing elements to pass data to one another using a complex architecture that allows for massive data throughput on the CPU itself.



Key DMA modes → Burst Mode

Here the **DMA** machine simply takes over control of the bus makes the data transfer at top speed, then hands control back to the CPU. This is fast, but the CPU is stopped dead for the duration of the transfer.

SUMMARY:

Take control of bus.

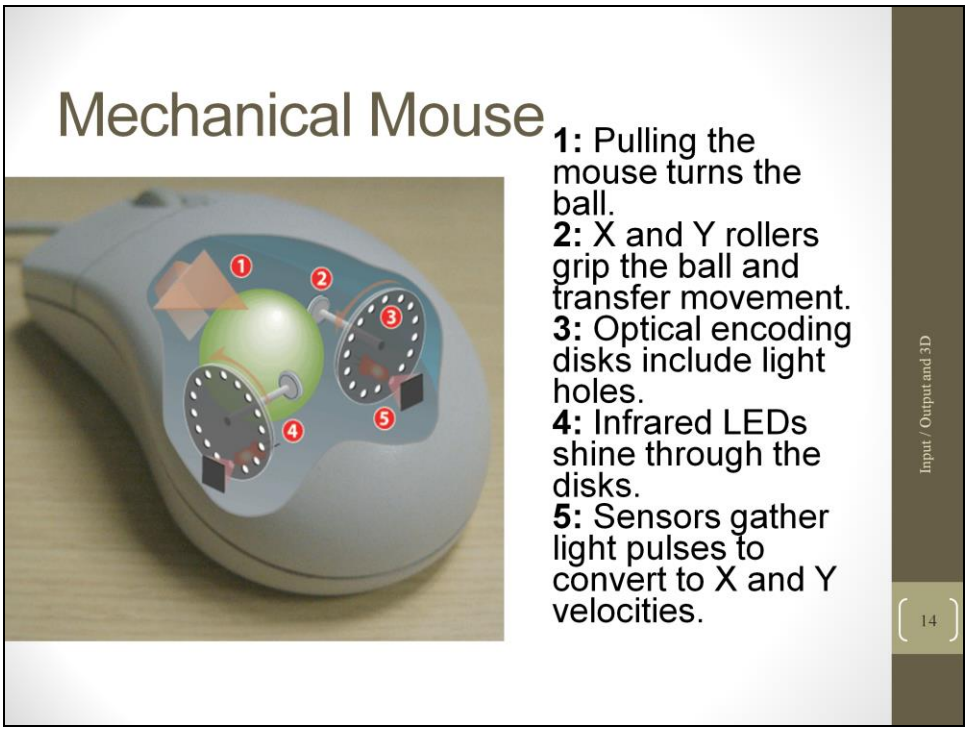
Stop the CPU

Send all the data to Memory

Restore bus control to CPU

Key DMA modes → Cycle Stealing → sometimes called transparent DMA

In **cycle stealing** mode the DMA controller has the ability to "**steal**" bus cycles for its own data transfer, stopping the CPU. For example, the DMA controller may take every second cycle off the CPU to use the bus Cycle stealing less used now (?) due to mixture of high speed CPU & memory and lower speed devices – and higher CPU memory cache capacities.



Mouse mats sometimes required as ball can slip on low friction surfaces.

Optical mouse

http://en.wikipedia.org/wiki/Optical_mouse

An optical mouse uses a light-emitting diode and photodiodes to detect movement relative to the underlying surface, unlike wheeled mice which use a set of one rolling ball and two chopper wheels for motion detection.

Older optical mice required special marked mouse mats to work, current optical mice work on most surfaces – but not all

Trackpad is just one of many touch devices. For current computers and handheld devices, touchscreen devices have become common place. There are a wide range of different technologies used for trackpads and touchscreens, some of which support ‘multi touch’ – capable of tracking multiple touches and movements at the one time. Older and cheaper technologies are only capable to detecting one point of contact at a time – and can be confused by accidental swipes with other fingers during use.

Gamepads such as the PS3 DualShock 3 combine multiple forms of input and also some output

Inputs: Multiple buttons. Some of which may be able to detect not just on/off but how hard the button is being pressed (this is often termed analog input, but will be converted to a digital value before being sent to the console)

On the controller shown, two sticks can be pushed in different directions. These analogue sticks can be pushed in any direction, just slightly or to an extreme. The value detected by

the console will be a direction and a strength.

A further four shoulder buttons cannot be seen from this angle – giving a large range of inputs for a simple hand held device.

Internal accelerometer tilt sensors can detect the orientation of the gamepad, allowing users to control in game action simply by tilting the device. Controller is not only used for Input but also gives some output.

Output: Modern controllers often include some form of Haptic force-feedback. This will use coils which react to an electric current making the whole controller shake and rumble, and can be controlled by the program running.

Like most current controllers, this is a wireless controller. These tend to use Bluetooth signals to communicate with a nearby console.

WiiMote works with a (misnamed) sensor bar.

Sensor bar sends IR signals, which are sensed by the WiiMote and help it determine its position relative to the sensor bar. Wii MotionPlus provides more accurate orientation data.

Game play combines use of buttons on WiiMote with ‘natural’ gestures – such as ‘rowing’ with the WiiMote. Opened up a lot of new games. It is a simple controller.

X- box Kinect – body motion control alternative to controller. ‘Controllerless’ (sort of)

Combines 3D depth sensors and visual camera sensors to allow it to locate and extract player position and stance from what it sees (difficult to do without the 3D data)

Also includes a microphone array – allowing not just speech commands to be given, but also sound localisation.

Kinect → RGB camera in the middle, an infrared laser projector and a sensor for range

finding. It can build up a 3D depth image of what is in front of it, this allows it to identify people standing and moving in front of it. It also has a microphone which can recognise speech, and also speech localisation, where sound is coming from. One of the fastest selling gadgets in the UK.

Modern smartphones have touch control (as in trackpad), but on screen directly.

Combined with few buttons, touch is the main control interface.

Also have additional sensors: **accelerometers/compass** to determine orientation and motions, **GPS** to determine position, **cameras** for visual input.

Modern smartphones are now rich and complex devices. Some androids have full keyboards.

Google has a program which allows

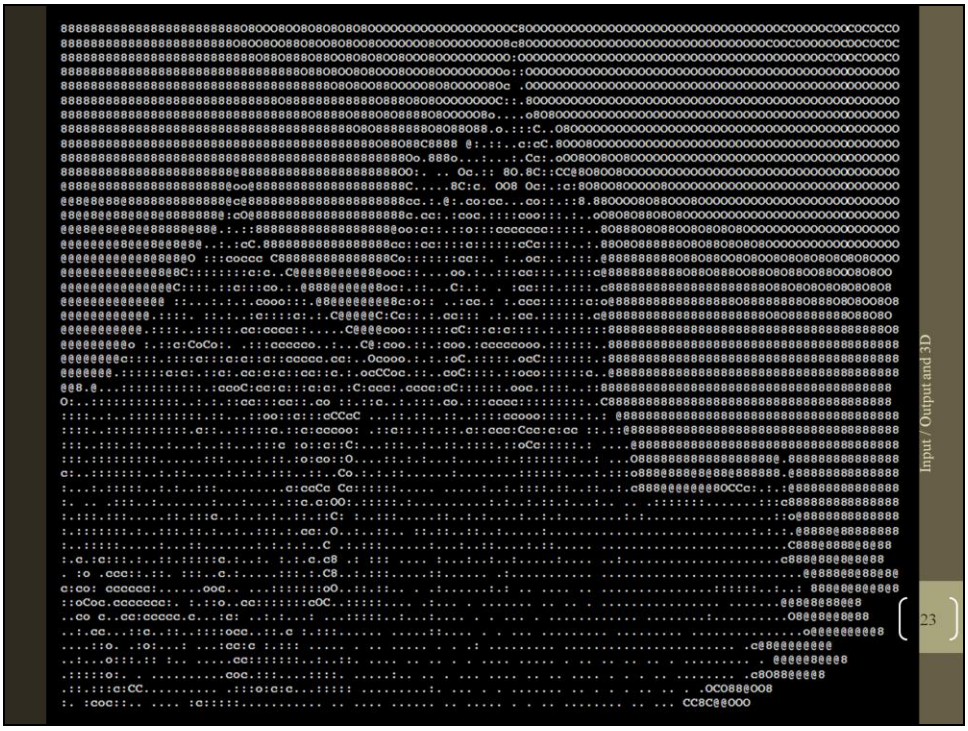
peripherals to connect to androids devices using USB, therefore gamepads and other devices, will be able to be used with android phones and pads.

Quick quiz

- Join the ‘Socrative’ app ‘Room 642124’ and try the quick quiz.

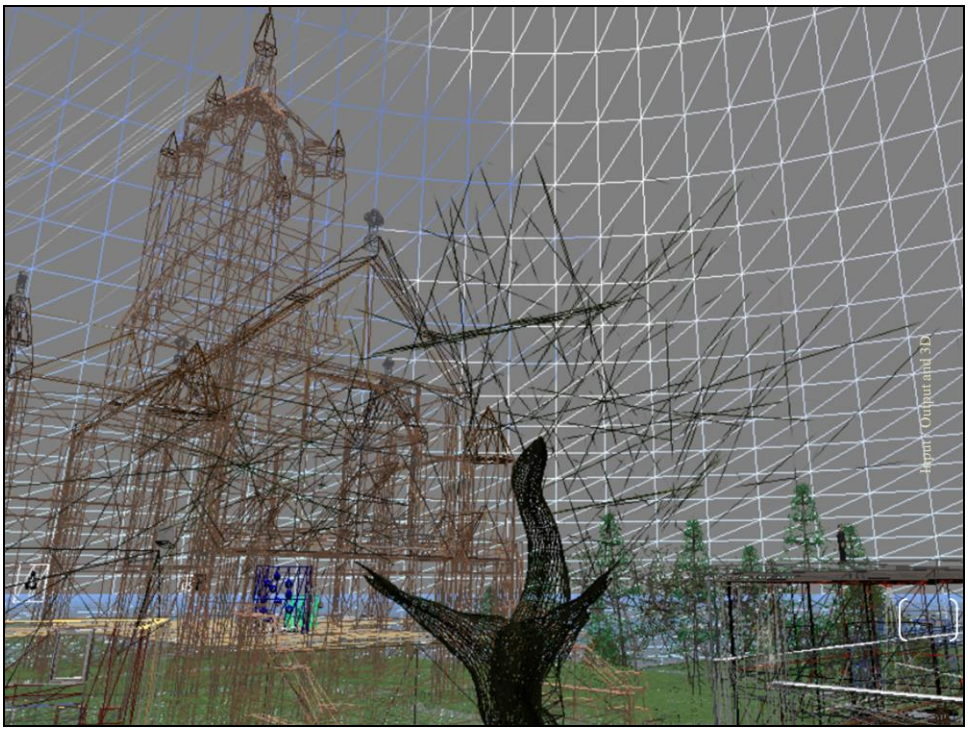
Graphics

- Until late 70's most computers were limited to text output
- Early graphics displays could be ***vector*** or ***raster*** based
  - ***Vector*** displays would control the ray in a cathode ray tube to draw lines
  - ***Raster*** displays split screen into rows and columns of *pixels* (picture elements)
    - Screenbuffer holds colour/gray values for all pixels



ASCII art of Coats memorial church in Paisley. Early form of graphics art. Image is made up of ASCII symbols.

Here is a simple 3D scene of the same church.



And here is a wireframe view of the same scene.

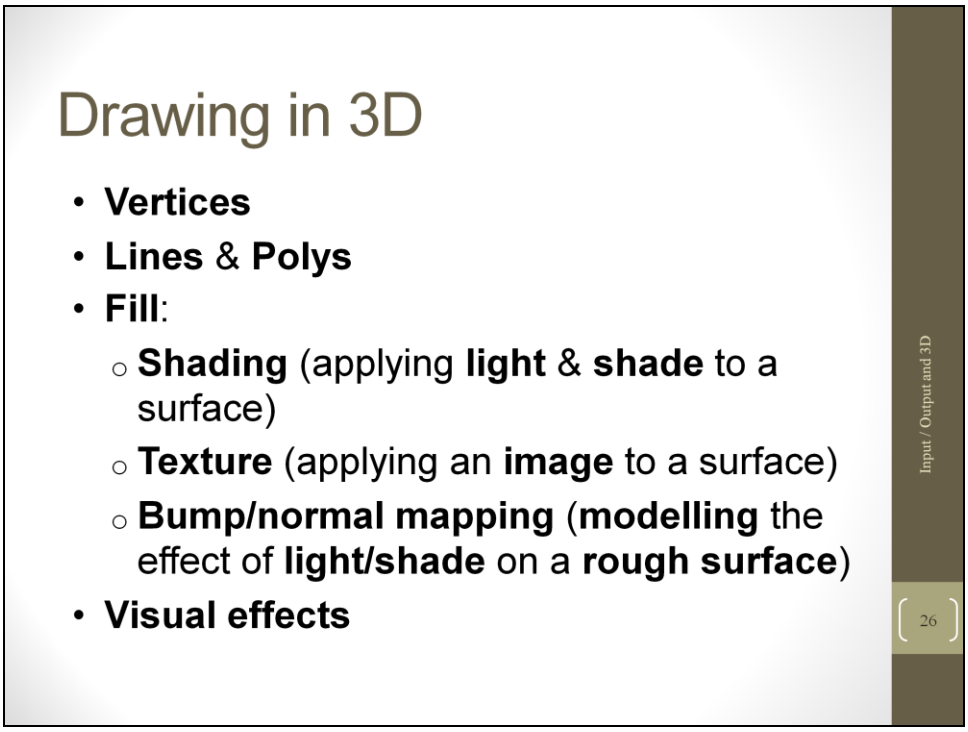
Each **vertex (corner)** of each line or **polygon (poly, *n*-sided shape)** represented with 3 or 4 values to represent its position in a three dimensional space.

If you are programming graphics you'll learn more about this in other courses

Displaying **3D graphics** required **rendering** (drawing) lots and lots of **lines** and **polys**.

The surfaces of each shape has been turned of.

Triangles are used as the images are guaranteed to be flat.

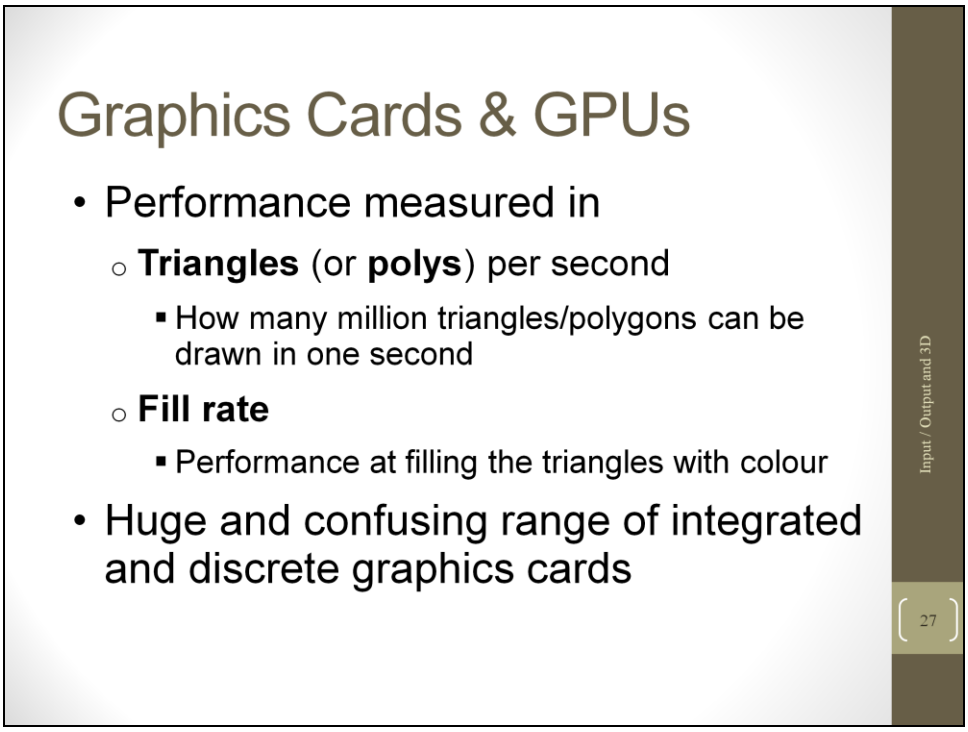


A single object might require thousands or tens of thousands of vertices (corners), which need connected to form lines and polygons (polys)
These then need filled – not with solid colour, but with textures – 2D graphics images
Lighting and shading need applied as does 'bump mapping' – additional textures are used that model the bumps on a surface and these are used to modify the output of the lighting calculations to show areas of greater and lesser shadow on the surface.

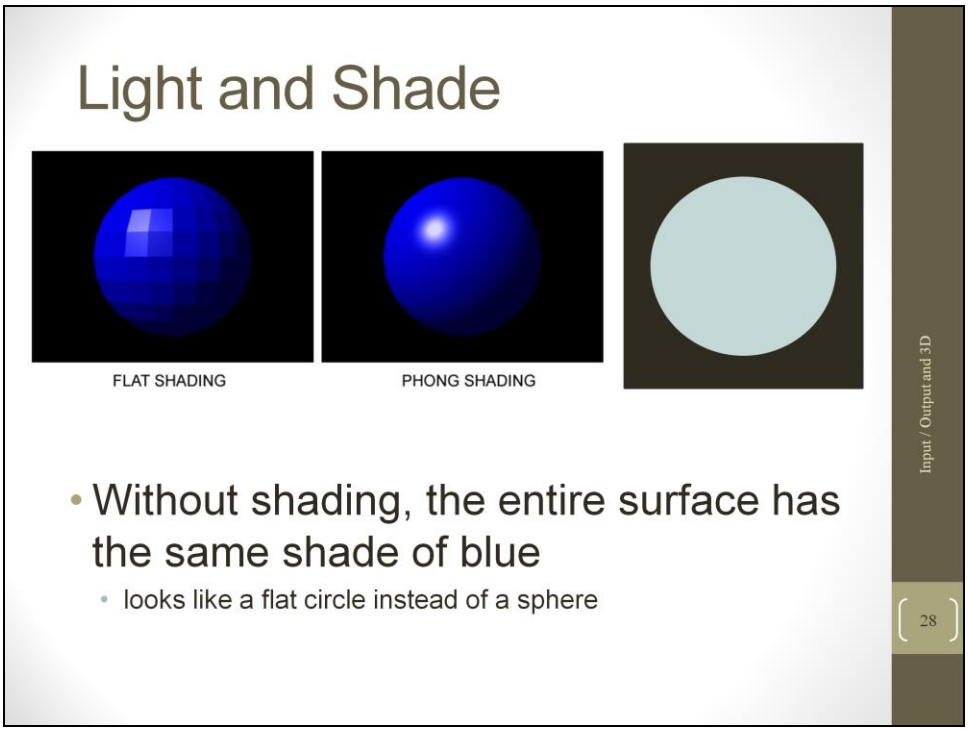
ALL of these steps involve geometric calculations – vertices themselves are represented simply as points in 3D space – with 3 values to represent the position in x,y and z coordinates. (A fourth value is actually also used… more on this in later courses on graphics for those that will be doing 3D programming.

Then there are also a huge range of advanced visual effects that can be generated… at every step geometric calculations are required that operate on each and every of the tens of thousands of vertices being modeled, and which need to calculate the fill values for surfaces modeled with just a few vertices.

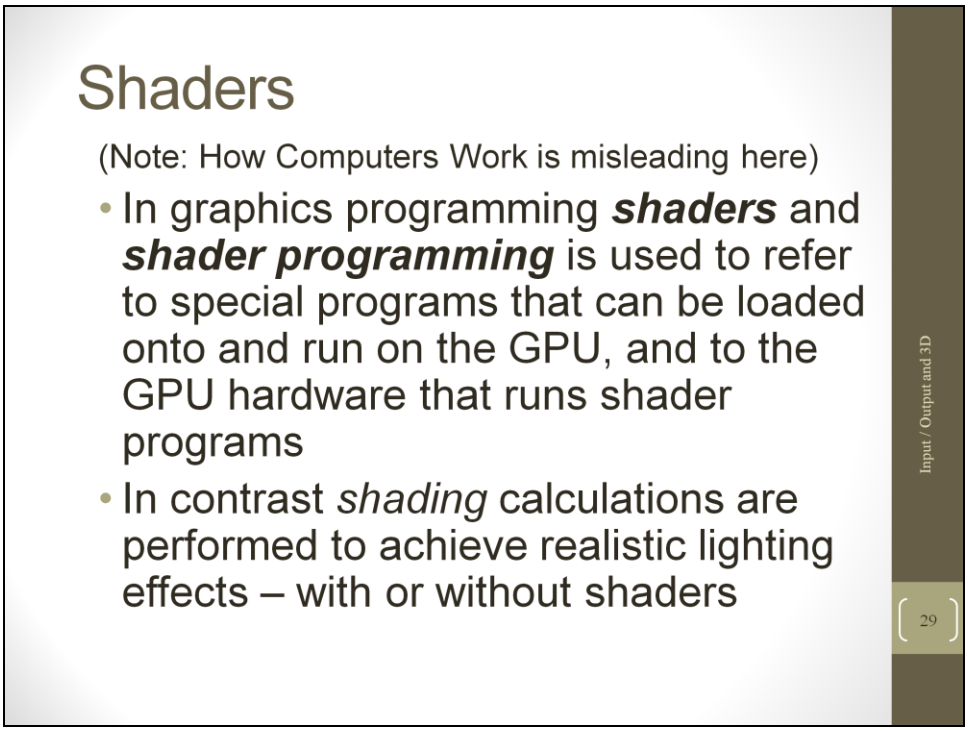
This is too involved for a typical CPU !



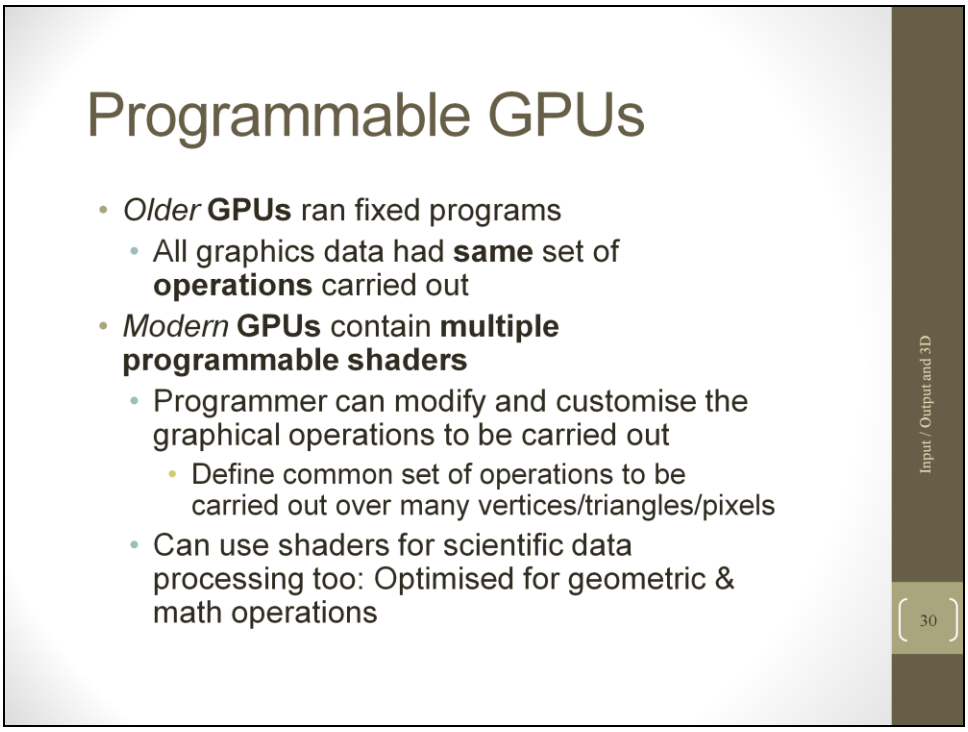
GPUs use designs that are built to efficiently perform the geometric types of operations required for 3D graphics, performing the same calculations for the many thousands of points/vertices/surface points being rendered in one drawing operation.
More specific design for using one instruction on large batches of data allows far greater performance at graphics tasks than a CPU could do



This can be confusing... But shading is used to illuminate objects. Flat colour would hide the shape otherwise.



This can be confusing... But shading is used to illuminate objects. Flat colour would hide the shape otherwise.



Nvidia CUDA and OpenCL are APIs that allow programmers to use the power of GPUs to perform more general computing tasks. The GPUs are still not designed as general CPUs, and are designed primarily to perform math operations on large arrays of numbers. But modern GPUs are themselves Turing Complete. Some modern supercomputing cluster designs use small numbers of CPUs to support processing carried out on large numbers of GPUs

If GPUs are so good for problem solving, why aren't they used instead of CPUs?

- **GPUs** have far *more* **processor cores** than **CPUs**, but because each GPU *core* runs significantly **slower** than a **CPU** core and do not have the features needed for modern operating systems, they are not appropriate for performing most of the processing in everyday computing.
- They are most **suited** to **compute-intensive** operations such as **video** processing and **physics simulations**.

Quick quiz

- Join the ‘Socrative’ app ‘Room 642124’ and try the quick quiz.

Further Reading

- Wikipedia:
  - Interrupt, Direct Memory Access
  - Computer Mouse, Trackpad, Touchscreen
  - Kinect, WiiMote, DualShock, iPhone
  - Computer Graphics, Shading, Shader
- PCH
  - Chapter 10

Image Credits:

Title Image, Optical mouse, ASCII art picture of Coats memorial church (Paisley, UK), Screenshot of Coats memorial church in Second Life, Wireframe image of Coats memorial church in Second Life – CC-BY-SA Daniel Livingstone, 2011

Keyboard – CC-BY-SA Michael Maggs, http://commons.wikimedia.org/wiki/File:QWERTY_keyboard.jpg

Mechanical Mouse cutaway , ©with permission for reuse for any purpose, by Jeremy Kemp, http://commons.wikimedia.org/wiki/File:Mouse-mechanism-cutaway.png

Apple MacBook Trackpad, CC-BY-SA Highway of Life, http://en.wikipedia.org/wiki/File:Macbook_pro_trackpad.jpg

PS3 Dualshock 3 – Public Domain by Evan Amos http://commons.wikimedia.org/wiki/File:PS3-DualShock3.jpg

WiiMote – Public Domain by Greyson Orlando, http://commons.wikimedia.org/wiki/File:Wii_Remote_Image.jpg

XBox 360 and Kinect – CC-BY James Pfaff http://www.flickr.com/photos/84606909@N00/4702296194/

Android HTC Desire phone, CC-BY Irwing Swang, http://en.wikipedia.org/wiki/File:HTC_Desire_-_Sense_2.1.jpeg

Flat and Phong Shading, Public Domain by Jalo, http://en.wikipedia.org/wiki/File:Phong-shading-sample.jpg