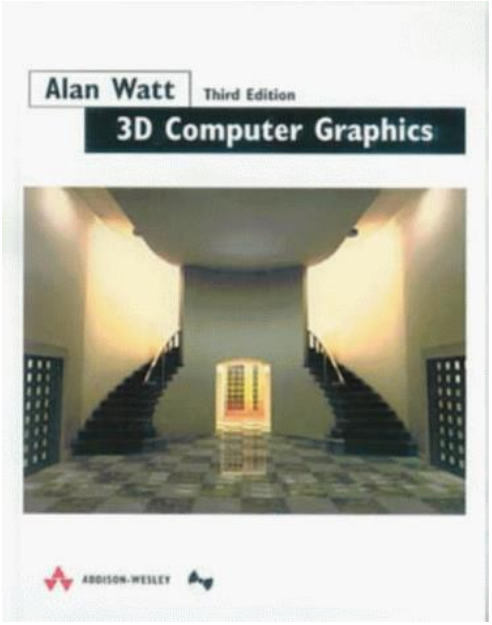
COM3503/4503/6503: 3D Computer Graphics
Lecture 1: Introduction…



Dr. Steve Maddock

s.maddock@sheffield.ac.uk

Room G011, Regent Court

https://staffwww.dcs.shef.ac.uk/people/S.Maddock

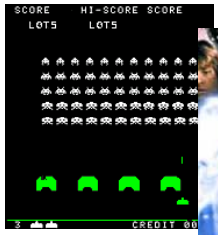
…to the module

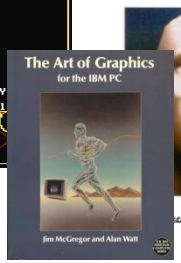
…to computer graphics

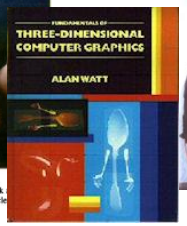
Outline

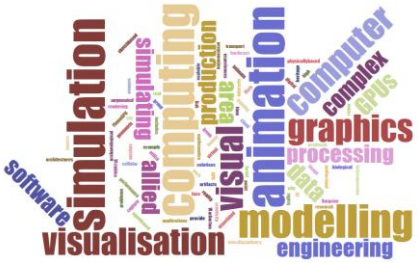
- Beginnings
- Organisation
- Where does CG fit in?
- What does CG involve?
- Outline of lectures
- Software practicalities
- Summary

Beginnings…









Pong, 72

Space Invaders, 78

Star Wars: A New Hope, 78

"Help me Obi-Wan…"

The Wall, 80

Tron, 82

Elite, 84

The Art of Graphics for the IBM PC, 84

Siggraph, 87

Watt, 89

Gremlin Interactive, 98

Current:
Head of Visual Computing Research Group

@stevemaddock

Outline

- Beginnings
- Organisation
- Where does CG fit in?
- What does CG involve?
- Outline of lectures
- Software practicalities
- Summary

The course website…

https://staffwww.dcs.shef.ac.uk/people/S.Maddock/campus_only/com3503/

- Aim: lecture slides posted before lecture
- May update some after lecture

The University Of Sheffield.

COM3503/4503/6503: 3D Computer Graphics

Home Lectures Labs Assessment Resources

COM3503/4503/6503: 3D Computer Graphics

Welcome to this module on 3D Computer Graphics. In lectures, I will be covering a range of fundamental topics that will give you the knowledge to work in many areas of computer graphics and help you to understand some of the techniques used in 3D animated films and games. In lab classes, you will be learning how to use the OpenGL API. The lab classes will ground some of the theory given in lectures and help you to understand the use of specific algorithms.

Resources

The University Of Sheffield.

COM3503/4503/6503: 3D Computer Graphics

Home | Lectures | Labs | Assessment | Resources

Resources

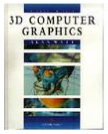
There are a range of good books on 3D Computer Graphics, each covering the relevant theory in this module. You will find it helpful to read at least one book. There are plenty of copies of Watt's books in the University library. Whilst this is a fairly old book, it covers the theory of 3D Computer Graphics well. My lecture notes build on the material in this book, enhancing it with more recent work.

A. Watt, "3D Computer Graphics (3rd edition)", Addison-Wesley, 2000

JOGL 2.0

The lab classes will provide exercise sheets to help you learn how to use JOGL. However, the following resources may also be useful:

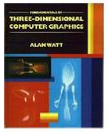
- jogamp - JOGL 2.x project home
- The jogamp JOGL Tutorial



A. Watt, "3D Computer Graphics (2nd edition)", Addison-Wesley, 1993

OpenGL

On the module we will be using modern OpenGL, which includes the programmable pipeline. (The most recent version is version is 4.5.) The tutorials below are done with C++ in mind. Nonetheless, all the relevant OpenGL can be reused in a Java program with a few changes.



A. Watt, "Fundamentals of 3D Computer Graphics", Addison-Wesley, 1989

- opengl.org
- John Kessenich, Graham Sellers, Dave Shreiner. OpenGL Programming Guide: The Official Guide to Learning OpenGL, Version 4.5 with SPIR-V Paperback (ninth edition), Addison Wesley, 2016
- Learn OpenGL by Joey de Vries - **I recommend this tutorial** for learning modern OpenGL. There is also a downloadable book.
- Anton's OpenGL 4 Tutorials - Also useful, and has an accompanying book
- Modern OpenGL tutorial by Alexander Overvoorde
- Free tutorials for modern Opengl (3.3 and later) in C/C++
- wikibooks: OpenGL programming
- Learning Modern 3D Graphics Programming by Jason L. McKesson

Maths for computer graphics

- immersive linear algebra by J. Ström, K. Åström, and T. Akenine-Möller
- Vector maths tutorial
- Essence of Linear Algebra: video tutorial series by Grant Sanderson on the mathematics of transformations and linear algebra.
- Dunn, F. and I.Parberry, 3D Math Primer for Graphics and Game Development, AK Peters/CRC Press, 2nd edition, 2011.
- Vince, J.A., "Mathematics for Computer Graphics (Undergraduate Topics in Computer Science)", (Fourth edition), Springer, 2014.

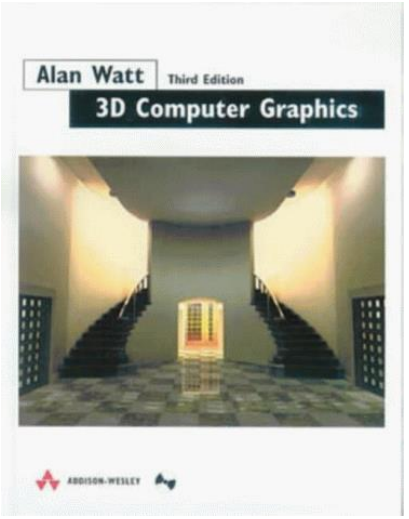
Java links

- Oracle's Java page - A good starting point to find many other Java resources.
- Oracle's Java tutorial - Lots of practical examples of using all the Java classes.
- Horstmann, C.S. and G. Cornell, "Core Java ..." - I recommend these books for learning to program in Java. They also include good material on building GUIs.

General

- Glossary of computer graphics terms
- GuerrillaCG - a non for profit organisation dedicated to teaching the fundamentals of computer graphics.
- Edinburgh online graphics dictionary
- Brown University cs123 resources
- Tools for movie generation and image processing
- Unity game engine (free download)
- jMonkeyEngine: 3D games engine using Java and OpenGL
- Blender - "free open source 3D content creation suite, available for all major operating systems under the GNU General Public License".
- Autodesk Education Community - Free autodesk software including 3ds Max, AutoCAD and Maya
- regentcourt.zip - a Google SketchUp model of Regent Court
- WebGL path tracing (Use Chrome web briwser)
- ACM SIGGRAPH
- Eurographics

Accessibility



Assessment

- Both assignment and exam
- All assignments will be available on MOLE
- All assignments must be submitted through MOLE

	Credits	**Programming Assignment** *Weeks 5-11* *(Handin: week 11, 3pm, Wed 6 Dec)*	**Individual research study** *Weeks 5-12* *(Handin: week 12, 3pm Wed 13 Dec)*	**Exam** *Sometime in weeks 13-15*
com3503	10	40%		60%
com4503	15	27%	33%	40%
com6503	15	27%	33%	40%

Reflections on last year

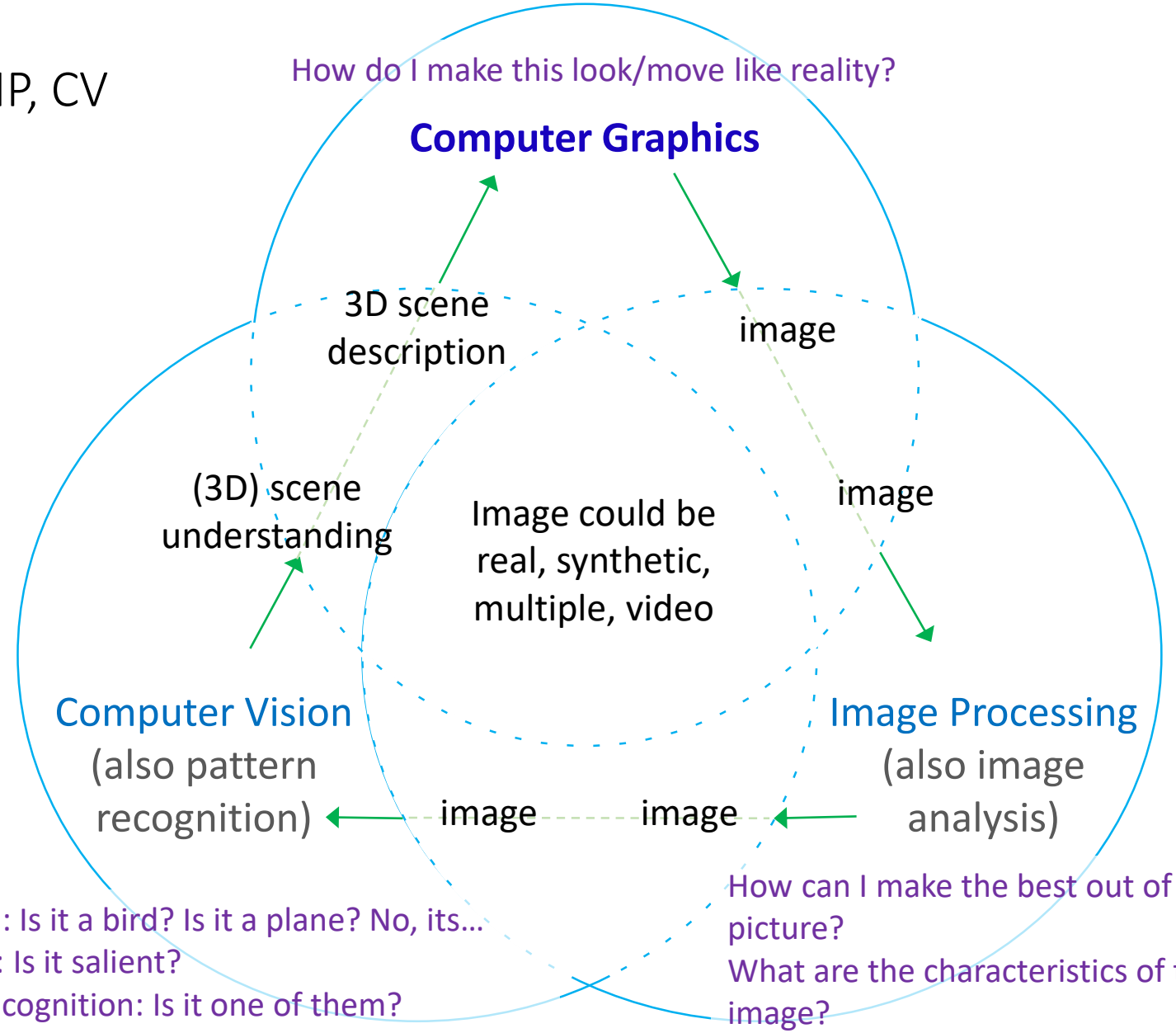
	Very Dissatisfied	Dissatisfied	Satisfied	Very satisfied
Com3503 (19/53 responses)		5%	42%	53%
Com4503 (1/3 responses)			67%	33%
Com6503 (13/17 responses)			54%	46%

- Lots of positive comments.
- But, a few students found the course challenging

Outline

- Beginnings
- Organisation
- Where does CG fit in?
  - Aspects of visual computing
  - Wider use of computer graphics
- What does CG involve?
- Outline of lectures
- Software practicalities
- Summary

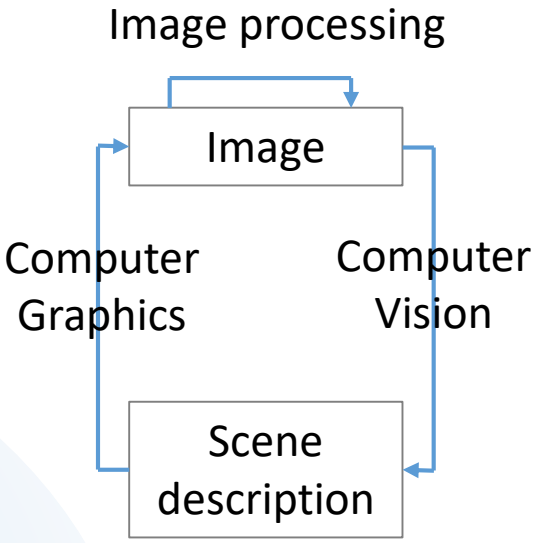
CG, IP, CV

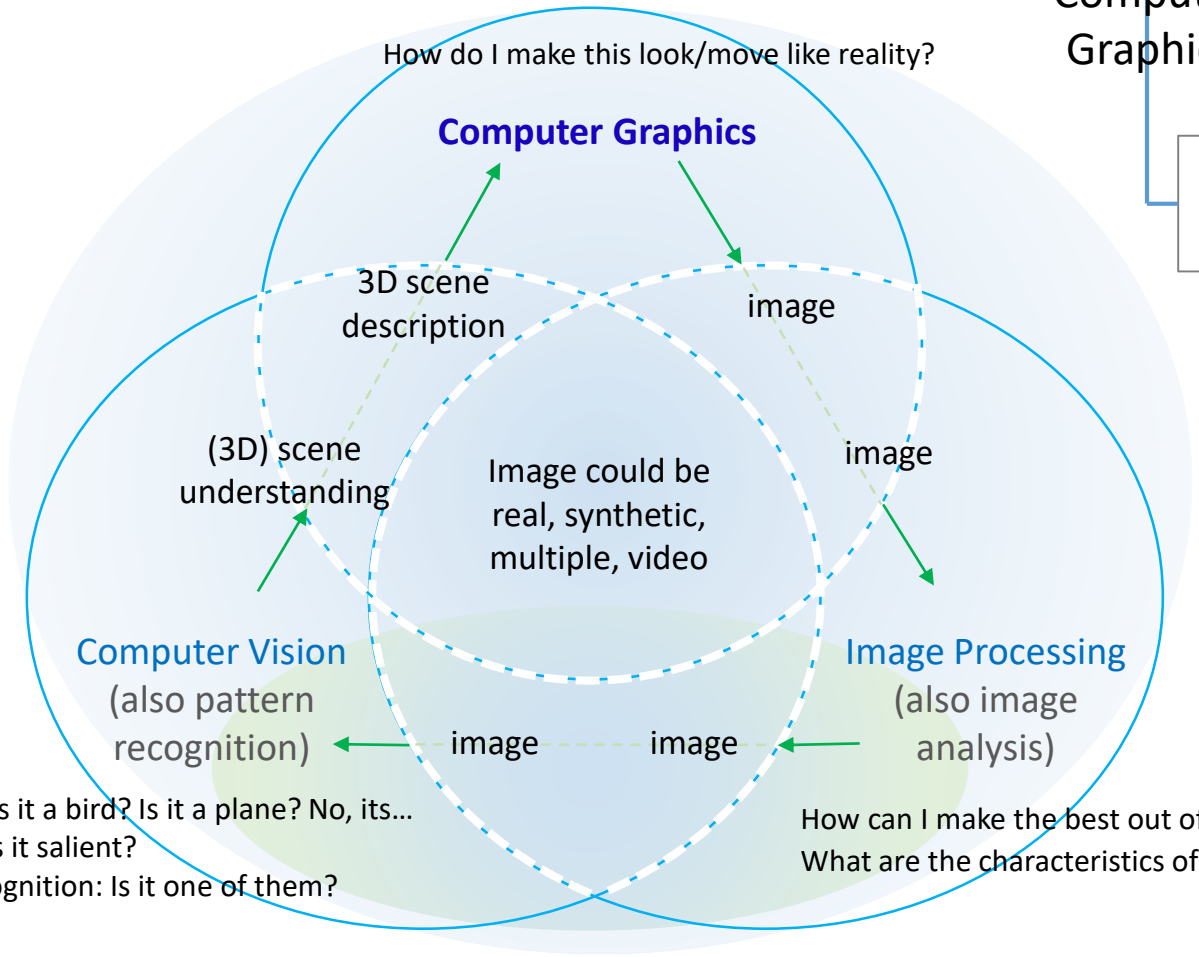


High-level: Is it a bird? Is it a plane? No, its…
Low-level: Is it salient?
Pattern recognition: Is it one of them?

How can I make the best out of a bad picture?
What are the characteristics of this image?

CG, IP and CV





High-level: Is it a bird? Is it a plane? No, its…
Low-level: Is it salient?
Pattern recognition: Is it one of them?

How can I make the best out of a bad picture?
What are the characteristics of this image?

1. Uses of computer graphics

- Training & testing: experiments in the real world can be expensive and dangerous
- Design – enables spaces/objects to be investigated before construction
- Understanding using scientific visualisation
- Enhancing the existing world – mix models with the real world: special effects, augmented reality
- Entertainment and pleasure – games, TV and films, art

The Diamond

The Virtual Hole in the Road link2

Outline

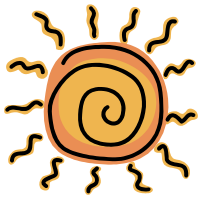
- Beginnings
- Organisation
- Where does CG fit in?
- What does CG involve?
  - Comparison with a camera
- Outline of lectures
- Software practicalities
- Summary

2. Real world versus synthesis

- In the real world, light transport happens naturally

Reflection

Reflections and Absorption

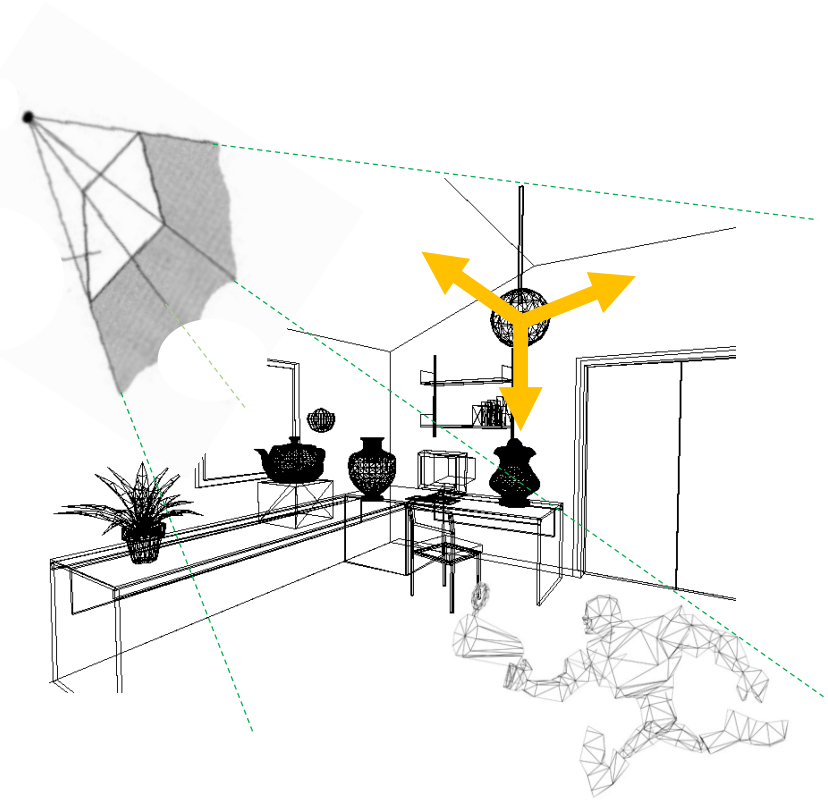


Illumination

Gold-painted Royal Mail post box celebrating heptathlete Jessica Ennis's 2012 Olympic gold medal (http://www.goldpostboxes.com/)

2. Real world versus synthesis

- In computer graphics we have to do everything…
- Geometric Modelling
  - Define a set of objects and arrange them in a scene
- Position a synthetic camera
- Define the lighting
- Calculate the lighting for each object
  - Different approaches simplify this in different ways
- Convert the 3D scene into a 2D view
- Animate the scene
- Requires: Lots of maths and programming



Maths

Online linear algebra book:
http://immersivemath.com/ila/index.html

- 2D and 3D coordinates
- Vectors
- Matrices
- …

Matrix algebra

$$\mathbf{A}=\begin{pmatrix} a_{11} & \cdots & a_{1n} \\ \vdots & \ddots & \vdots \\ a_{m1} & \cdots & a_{mn} \end{pmatrix}$$

$$A(BC) = (AB)C$$
$$AB \neq BA$$
$$AI = IA = A$$
$$(AB)^T = B^T A^T$$
$$A^{-1}A = I$$

$$a + b = b + a$$
$$(a + b) + c = a + (b + c)$$
$$\lambda(a + b) = \lambda a + \lambda B$$
$$\dots$$
$$v = (x_1, x_2, x_3)$$
$$\|v\| = \sqrt{(x_1^2 + x_2^2 + x_3^2)}$$

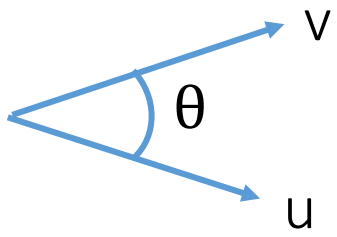
Dot product

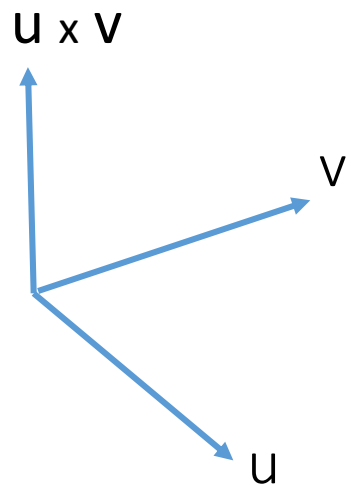
$$u.v = u_1v_1 + u_2v_2 + u_3v_3$$
$$u.v = \|u\|\|v\|cos\theta$$

Cross product

$$\|u \times v\| = \|u\|\|v\|sin\theta$$

$$\begin{pmatrix} a \\ b \\ c \end{pmatrix} \times \begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{pmatrix} bz - cy \\ cx - az \\ ay - bx \end{pmatrix}$$





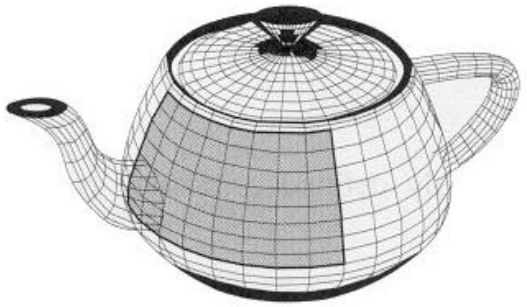
Outline

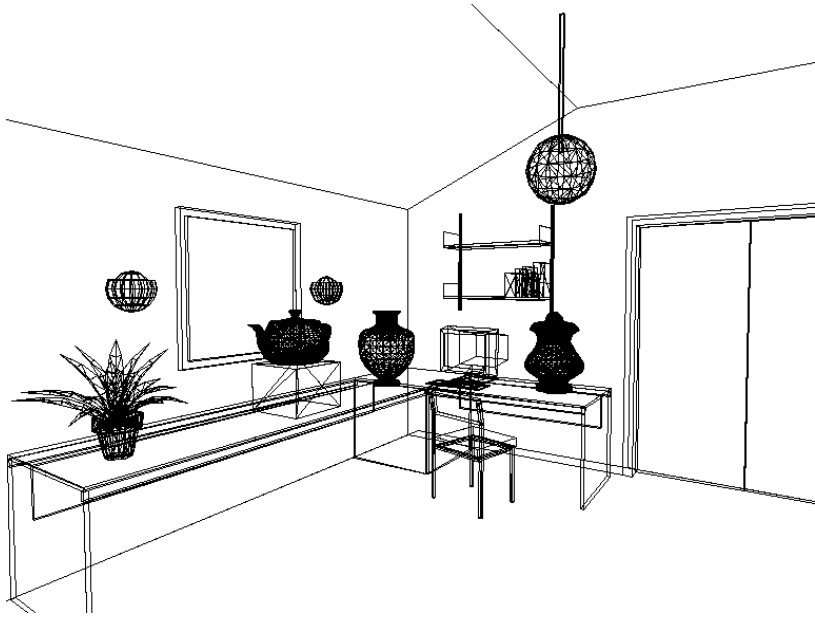
- Beginnings
- Organisation
- Where does CG fit in?
- What does CG involve?
- Outline of lectures – Dept intranet:
  - http://www.dcs.shef.ac.uk/intranet/teaching/public/modules/level3/com3503.html
  - …4503.html, 6503.html
- Software practicalities
- Summary

3. Modelling

(~6 lectures)

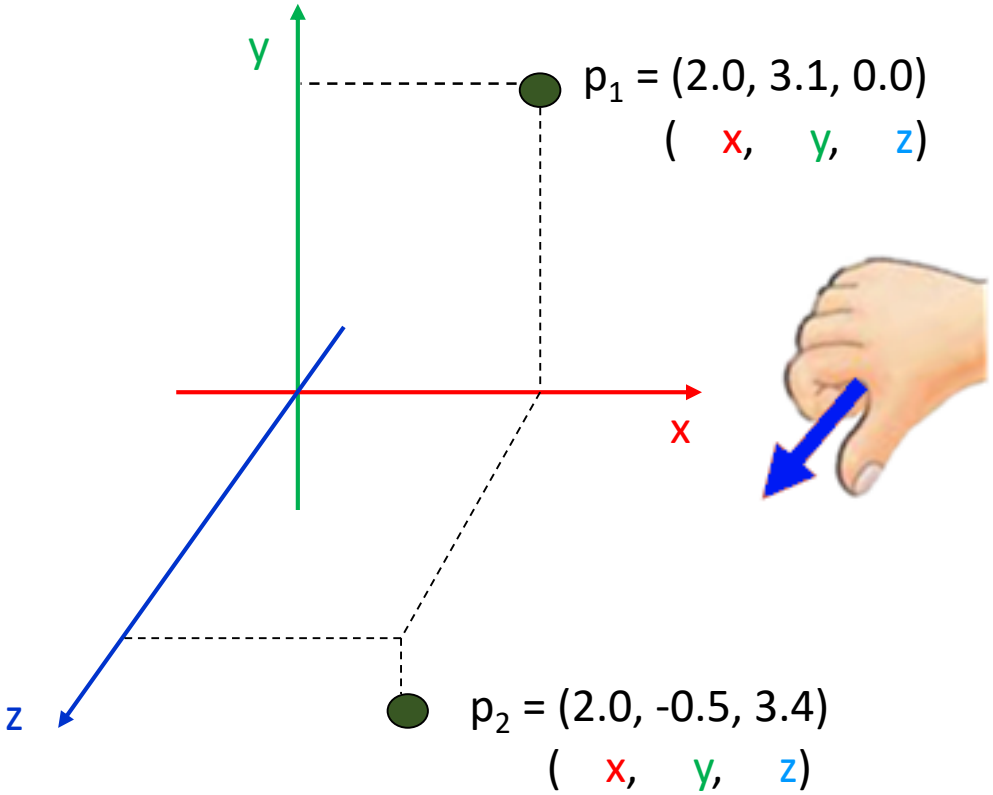
- A 3D model is a geometric description of a real-world object
- Lots of alternative representations
  - Polygons, patches, CSG, implicit surfaces, procedural approaches, space subdivision, …
- The models are arranged in a scene
  - Scene graphs, transformations, projections





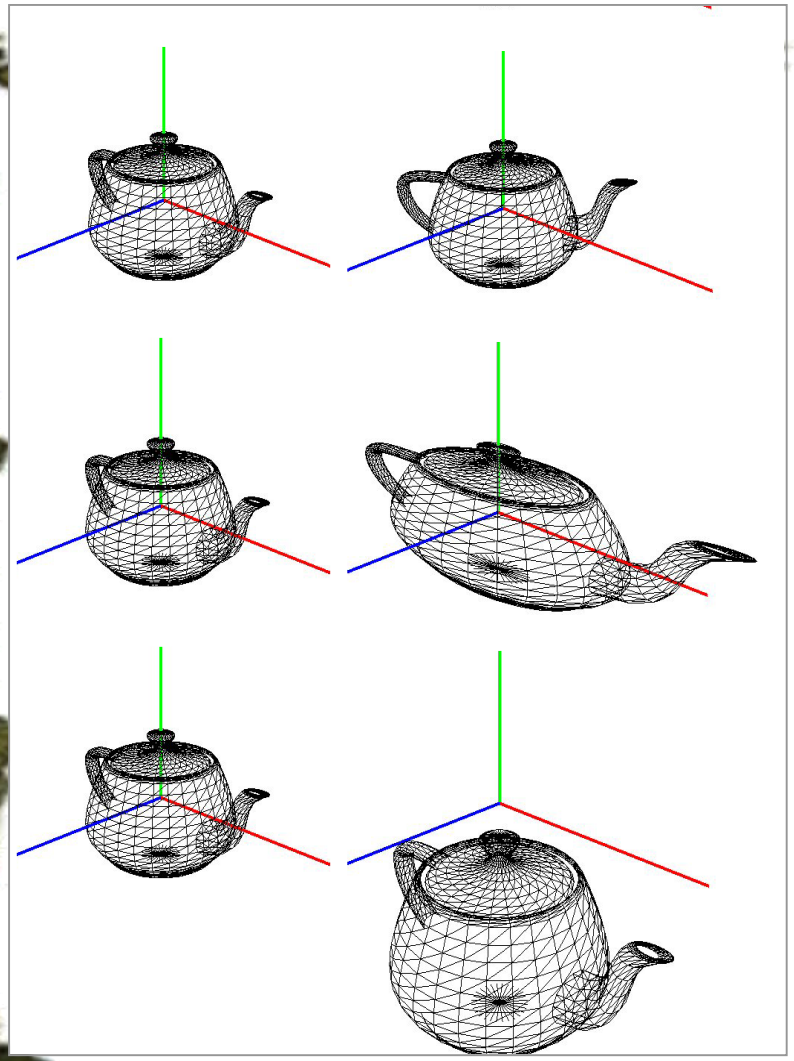
3.1 Coordinate systems and transformations

- Coordinate systems are used to describe locations of points
  - Polygon vertices are represented as position vectors, e.g. (2.0, 3.1, 0.0)
- OpenGL adopts a right hand coordinate system (RHS)



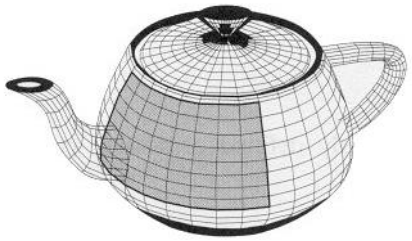
3.1 Coordinate systems and transformations

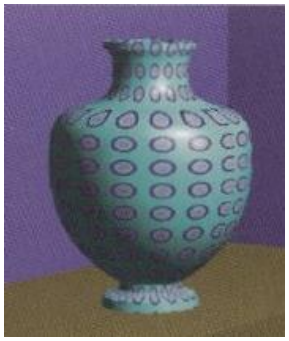
- Common transformations are rotation, scale and translation
- Transformations are represented as matrices
  - Composition of transformations – matrix multiplication
  - Linear algebra and homogeneous coordinates
- We'll also use transformations to build scenes and hierarchical objects

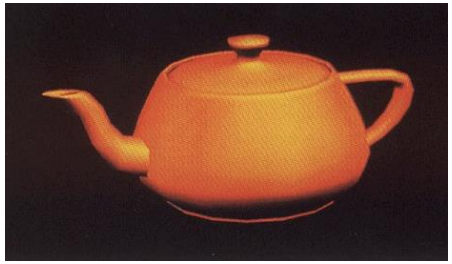


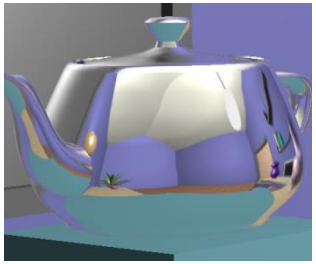
3.2 Material properties

- Specify material properties of each object
  - colours, textures









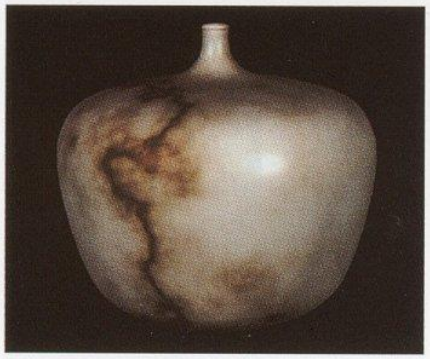
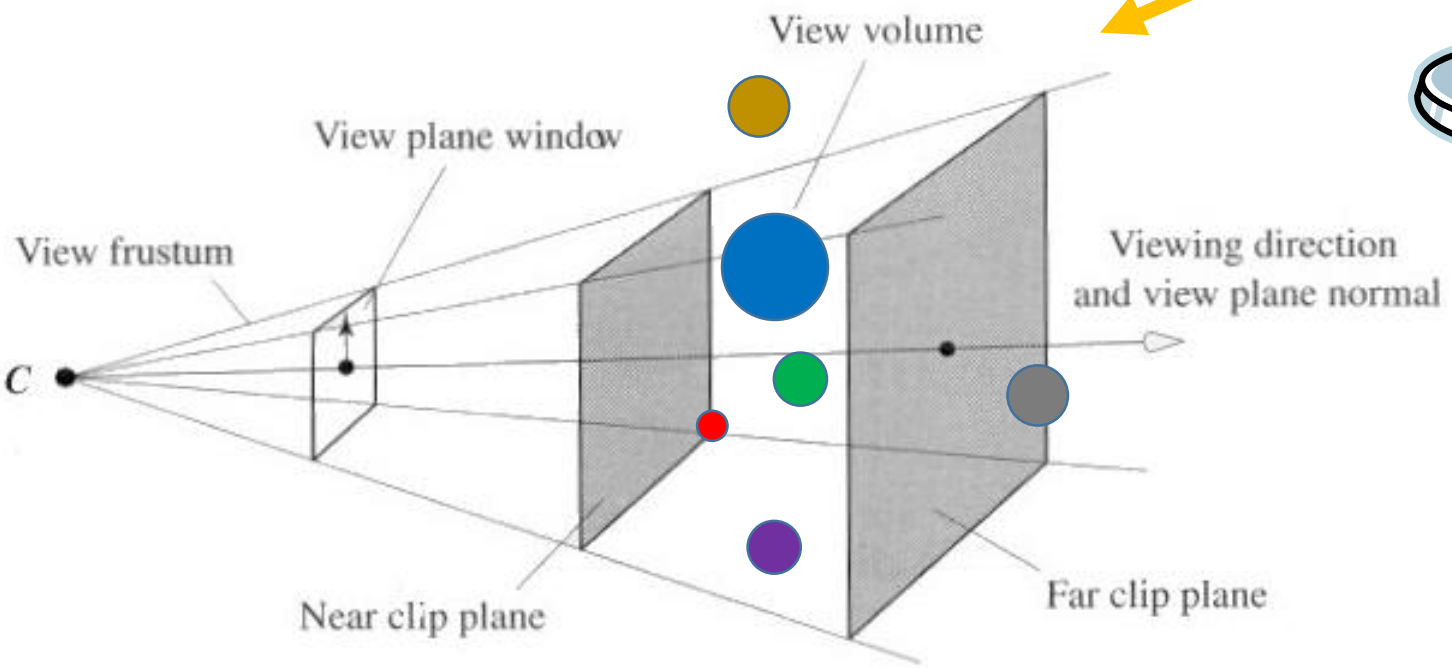
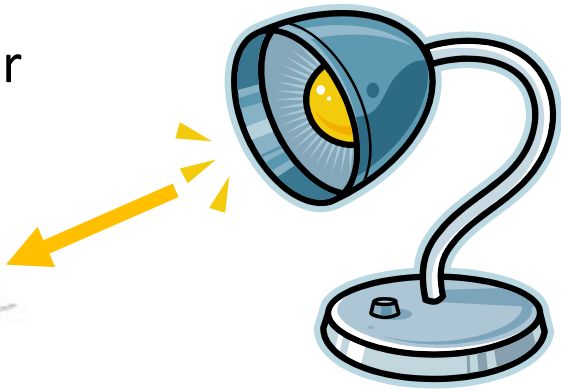


Figure 8.21 Imitating marble – the classic example of three-dimensional procedural texture.

3.3 Lights and cameras

- We specify a camera position, a view direction and an up vector
  - Create a view space, i.e. position of everything relative to the camera
- We specify a lighting position, direction and other parameters





4. Rendering

(~10 lectures)

- Convert the 3D scene into a 2D view
  - View space and perspective projection
  - Hidden surface removal – decide which objects can be seen

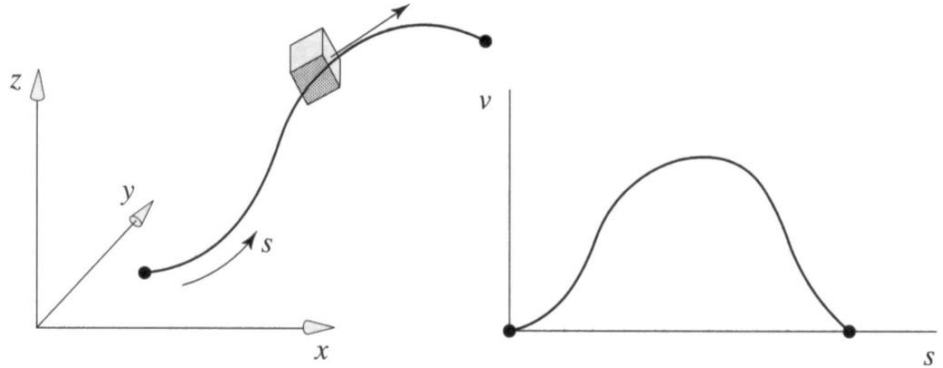
4. Rendering

- Calculate the lighting variation over each object's surface
  - Consider local and global illumination
  - Texture and shadows
  - Different approaches deal with this in different ways – interactive rendering, photorealistic rendering, artistic rendering

5. Animation

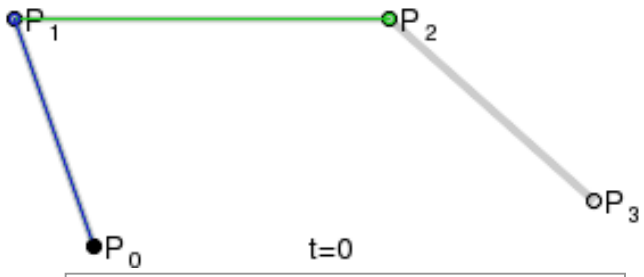
(~3 lectures)

- How to update parts of the scene, e.g. position, hierarchy, colour, lighting, etc.

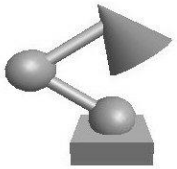


Explicit path characteristic

Explicit motion characteristic



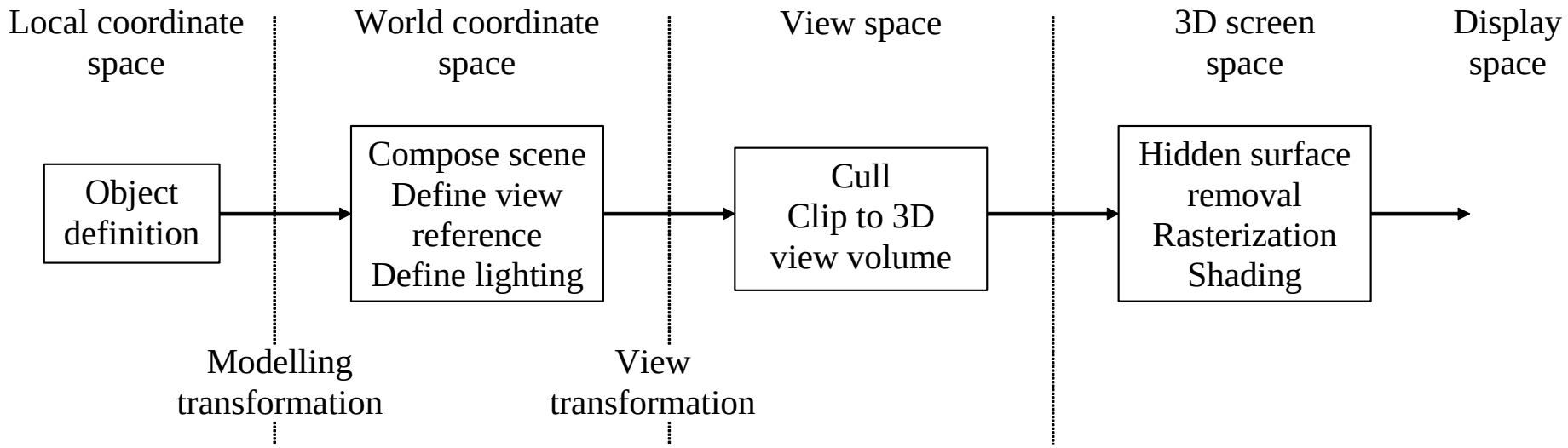
http://en.wikipedia.org/wiki/Bezier_curves



Knep et al, 95

6. The graphics pipeline

- For interactive rendering, we can consider the following high-level stages:



- For an advanced rendering approach such as ray tracing, aspects of this change

Outline

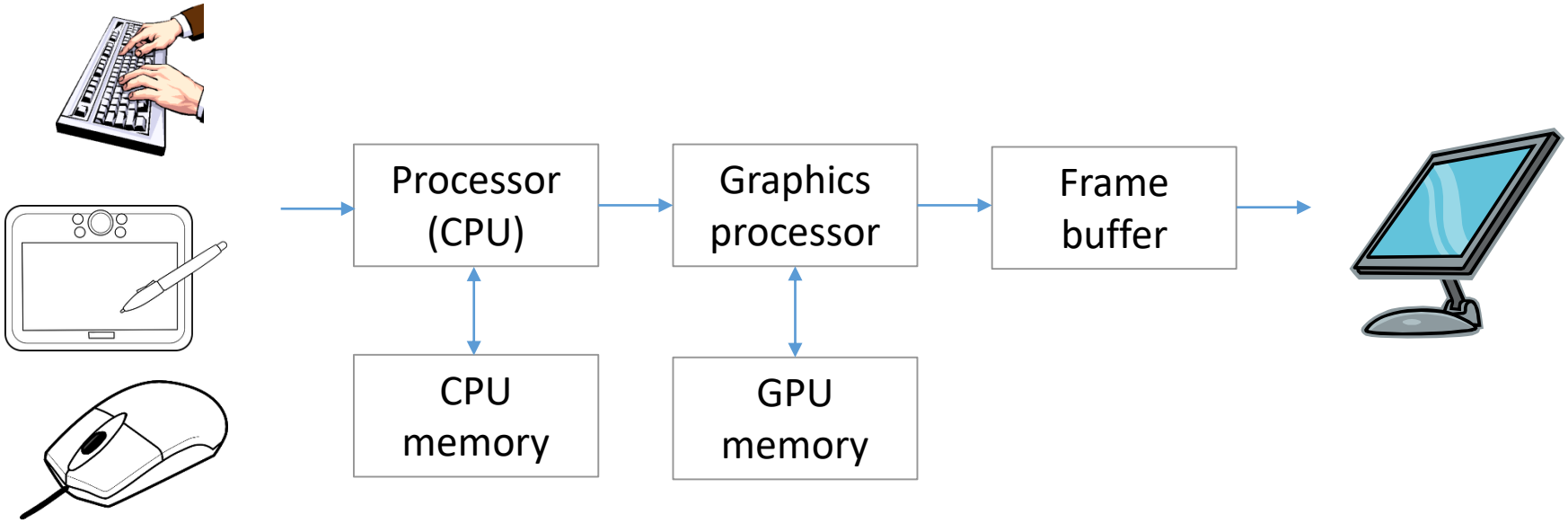
- Beginnings
- Organisation
- Where does CG fit in?
- What does CG involve?
- Outline of lectures
- Software practicalities
- Summary

7. Software for 3D computer graphics

- Use software tools to create and render models
  - 3ds Max, Blender, Cinema 4D, Maya, Mudbox, Poser, Rhinoceros 3D, SketchUp, Softimage, SolidWorks, ZBrush
- Use games engines to create and develop
  - Blender, CryEngine, Euphoria, jMonkeyEngine, Torque, Unity, Unreal
- Specialist tools for particular applications
  - AutoCAD, NUKE, VTK
- Use a graphics API
  - Direct3D, Java 3D, JOGL, LWGL, OGRE, OpenGL (,WebGL), OpenSceneGraph, RenderMan
- More often than not a mixture of tools is used to suit the application

- OpenGL is a hardware-independent, OS-independent, vendor-neutral computer graphics rendering API
  - We will cover JOGL, the Java binding for the OpenGL API

8. A basic graphics system



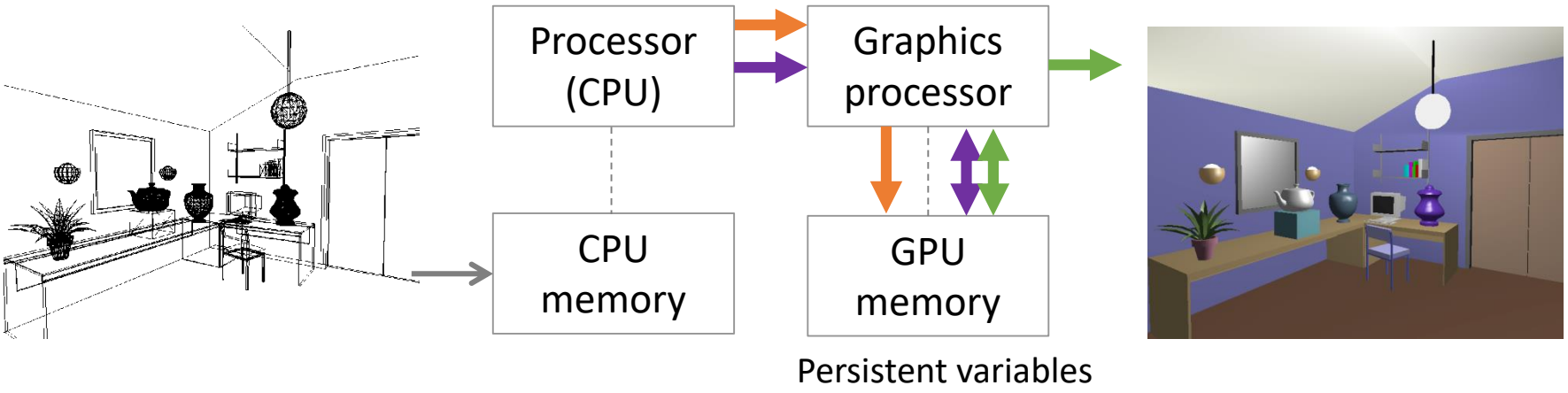
Input devices

Image formed in frame buffer

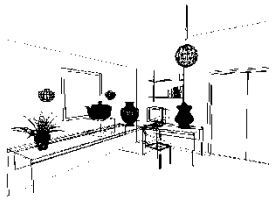
Output device

8.1 OpenGL Fixed-function pipeline

- State-based rendering – use API functions to:
  - Set up the state: Send information to change the OpenGL state machine (e.g. information about the camera and lights in a 3D world)
  - Pass in data, e.g. points, lines and polygons
  - Data is modified by existing state and then passed on as pixels to be displayed on the screen.



8.1 Fixed-function pipeline



3D geometry

3D API commands

OpenGL API

Application

CPU

GPU

vertex connectivity data

assembled primitives

pixel coordinates

pixels

GPU front end

Primitive assembly

Rasterization and interpolation

Raster operations

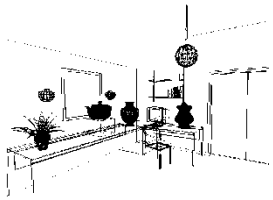
Frame Buffer

Vertex pipeline

Fragment pipeline

Transform geometry from 3D to 2D; Generate fragments from 2D geometry; Combine fragments to produce image

8.2 Programmable pipeline



3D geometry

CPU

GPU

3D API commands

OpenGL API

Application

vertex connectivity data

assembled primitives

pixel coordinates

pixels

GPU front end

Primitive assembly

Rasterization and interpolation

Raster operations

Frame Buffer

vertices

Programmable vertex processor

processed vertices

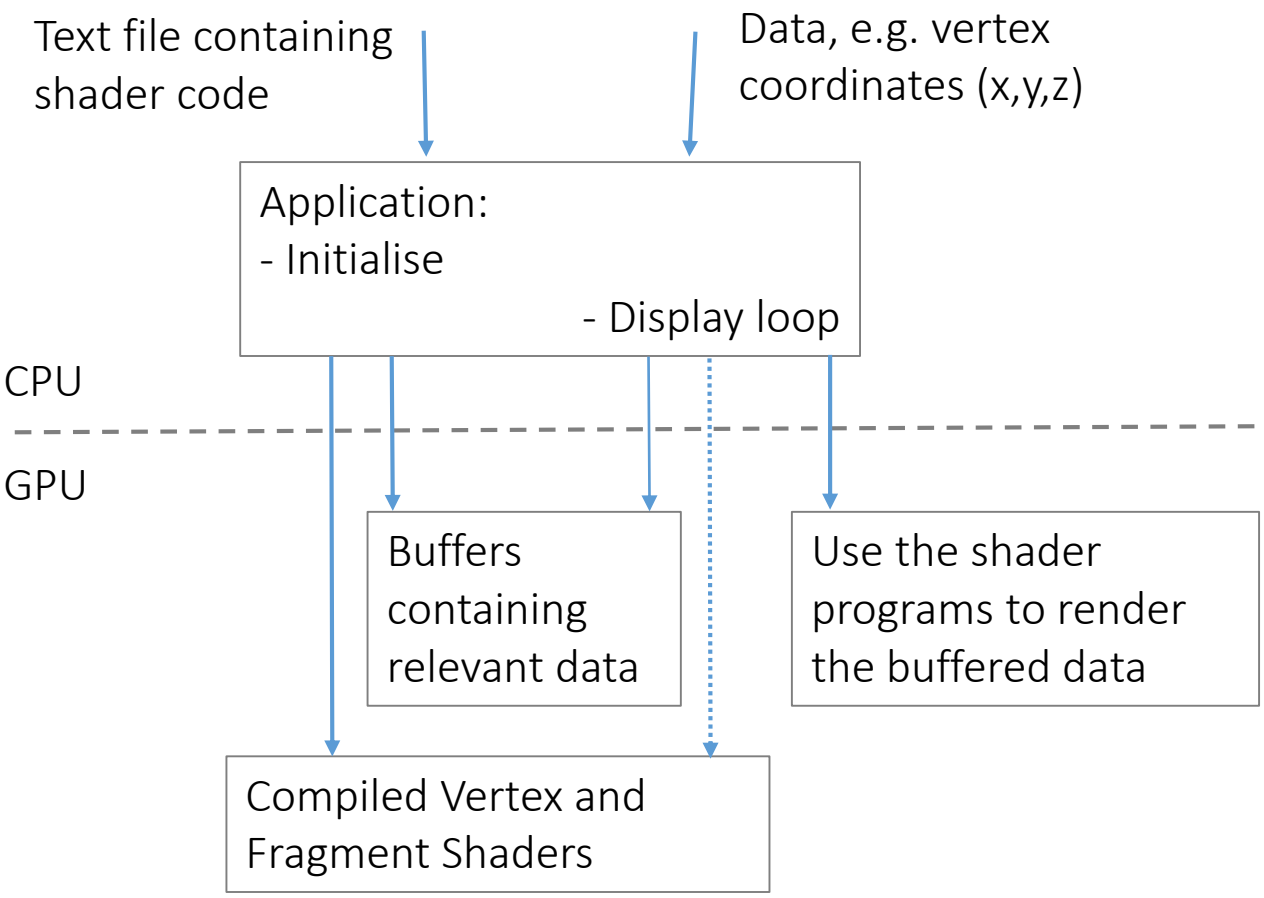
fragments

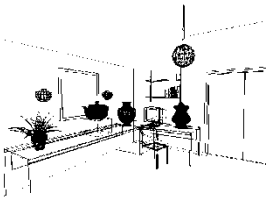
Programmable fragment processor

processed fragments

Tessellation shader, Geometry shader

8.2 Programmable pipeline





3D geometry

Rendered image

9. The assignment (More in week 5)

- Java
- OpenGL – provides an API for drawing objects specified in 3D
- JOGL – Java binding for OpenGL
  - http://jogamp.org/
  - JOGL 2, not 1.x
- Programming environment – your choice
- *However:* Assignments must compile and run from the command line (on a Windows machine)

OpenGL versions	
1.0-1.5	Fixed function pipeline
2.0-2.1	Support for programmable shaders
3.0	Adopts deprecation model (fixed function deprecated), but retains backward compatibility
3.x	Fixed function pipeline and associated functions removed (but can be accessed using a compatibility context)
4.x	Geometry and tessellation shaders
ES 1.x	Fixed-function version (stripped down) [ES - embedded systems]
ES 2.x	Programmable shader version
ES 3.x	Geometry and tessellation shaders

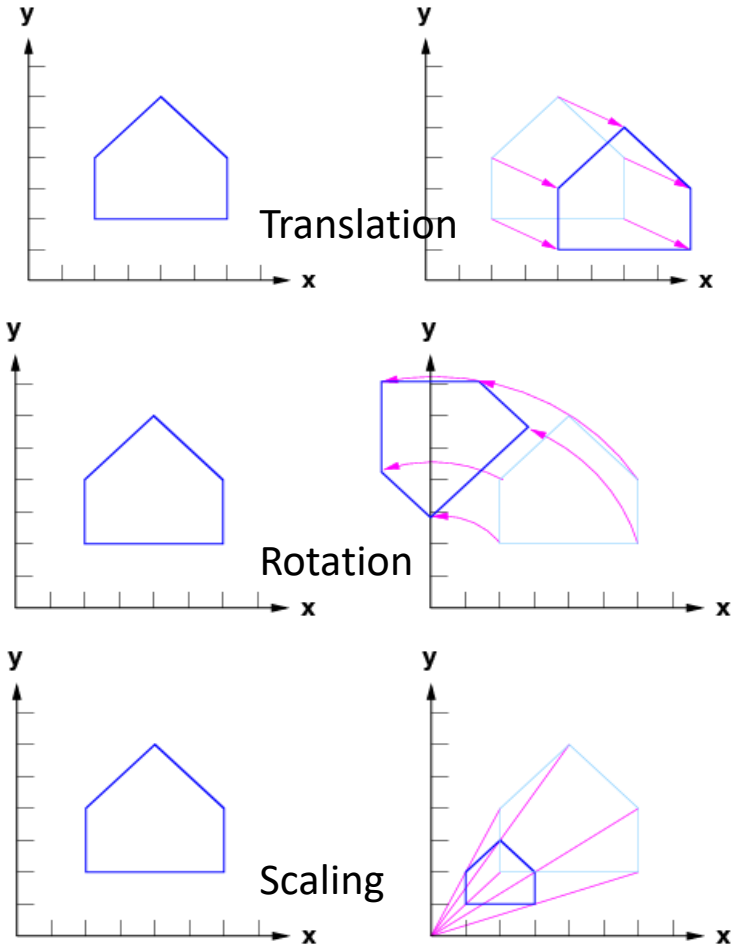
10. Summary

- The computer graphics pipeline gives us a 'track' to follow
- OpenGL is a cross-platform standard that provides an API for rendering objects specified in 3D
- *This Module:* Main focus is on a pipeline for rendering polygon mesh objects
- Towards the end of the semester: more advanced rendering approaches based on a global consideration of light illumination and reflection

- **Next lecture:** transformations
  - Please make sure you understand the material in the Appendix before the next lecture.
  - Read through the relevant chapters of http://immersivemath.com/ila/index.html and http://chortle.ccsu.edu/vectorlessons/

Appendix. Two-dimensional (2D) Affine Transformations

- The most common modelling and animation transformations we use are Translation, Rotation and Scaling
- Properties:
  - Parallel lines remain parallel
  - Collinearity: points remain on a line in same order (which means we can just transform endpoints of lines and join them to remake the lines)
  - Ratios of distance: midpoints remain as midpoints



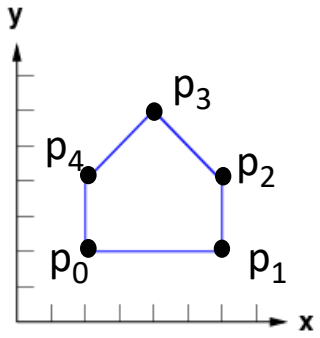
A.1 Translation

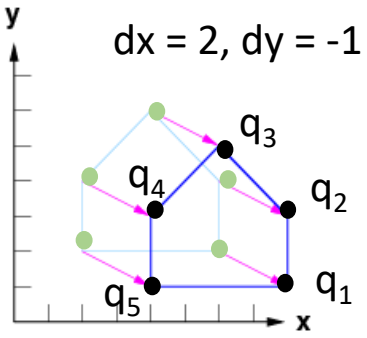
- Translate a point (x_i,y_i) to position (x'_i,y'_i) by distance (dx,dy)

$$x'_i = x_i + dx$$
$$y'_i = y_i + dy$$

- Given dx = 2, dy = -1, consider p_1:

$$x'_1 = 6 + \quad 2 \quad = 8$$
$$y'_1 = 2 + (-1) = 1$$





- Using vectors:

$$q_i = p_i + T \qquad \text{where } T = \begin{pmatrix} 2 \\ -1 \end{pmatrix}$$

$$\begin{pmatrix} x'_i \\ y'_i \end{pmatrix} = \begin{pmatrix} x_i \\ y_i \end{pmatrix} + \begin{pmatrix} dx \\ dy \end{pmatrix}$$

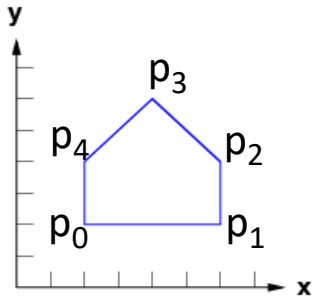
A.2 Scaling

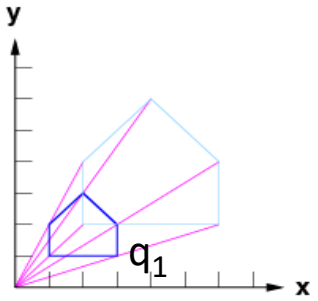
- Enlarge or shrink points:

$$x'_i = s_x\, x_i$$
$$y'_i = s_y\, y_i$$

- For p_1:

$$x'_i = 0.5 * 6 = 3$$
$$y'_i = 0.5 * 2 = 1$$





- Matrix arithmetic

$$q_i = S\, p_i \qquad \text{where } S = \begin{pmatrix} 0.5 & 0 \\ 0 & 0.5 \end{pmatrix}$$

$$\begin{pmatrix} x'_i \\ y'_i \end{pmatrix} = \begin{pmatrix} s_x & 0 \\ 0 & s_y \end{pmatrix} \begin{pmatrix} x_i \\ y_i \end{pmatrix}$$

- If s_x=1 and s_y=1, then its the identity matrix: $I_2 = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$

Reminder of matrix multiplication

- Matrix multiplication: ‘row x column’
- Example:
  - Assume array index starts at 0
  - result(0,0) = row ‘0’ of matrix a x column ‘0’ of matrix b

$$result = \begin{pmatrix} a_{0,0} & a_{0,1} \\ a_{1,0} & a_{1,1} \\ a_{2,0} & a_{2,1} \end{pmatrix} \begin{pmatrix} b_{0,0} & b_{0,1} & b_{0,2} & b_{0,3} \\ b_{1,0} & b_{1,1} & b_{1,2} & b_{1,3} \end{pmatrix}$$

$$= \begin{pmatrix} a_{0,0} \times b_{0,0} + a_{0,1} \times b_{1,0} & ? & ? & ? \\ ? & ? & a_{1,0} \times b_{0,2} + a_{1,1} \times b_{1,2} & ? \\ ? & ? & ? & ? \end{pmatrix}$$

- 3x2 matrix, `a`, multiplied by 2x4 matrix, `b`, gives a 3x4 matrix, `result`

A.3 Translation, scaling, rotation

$$q_i = p_i + T$$

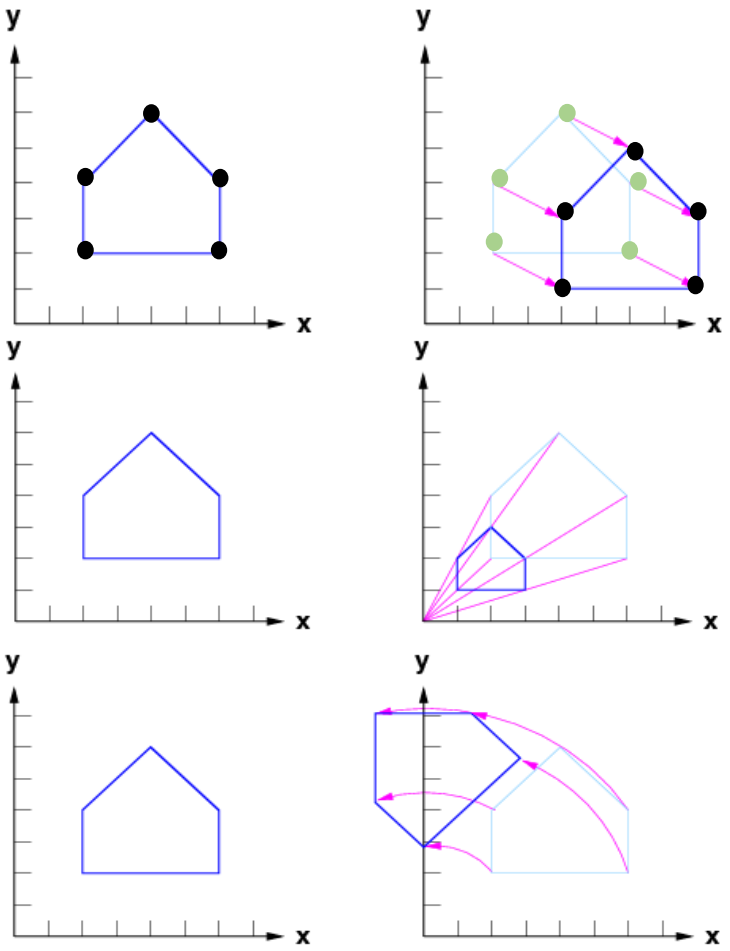
$$\begin{pmatrix} x'_i \\ y'_i \end{pmatrix} = \begin{pmatrix} x_i \\ y_i \end{pmatrix} + \begin{pmatrix} dx \\ dy \end{pmatrix}$$

$$q_i = S\, p_i$$

$$\begin{pmatrix} x'_i \\ y'_i \end{pmatrix} = \begin{pmatrix} s_x & 0 \\ 0 & s_y \end{pmatrix} \begin{pmatrix} x_i \\ y_i \end{pmatrix}$$

$$q_i = R\, p_i$$

$$\begin{pmatrix} x'_i \\ y'_i \end{pmatrix} = \begin{pmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{pmatrix} \begin{pmatrix} x_i \\ y_i \end{pmatrix}$$

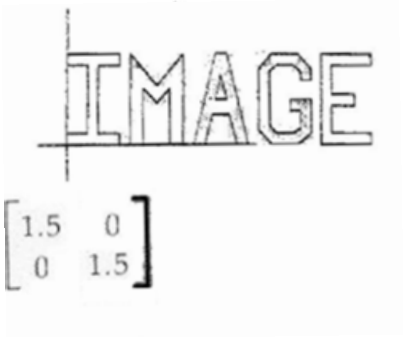


A.4 Respect to the origin

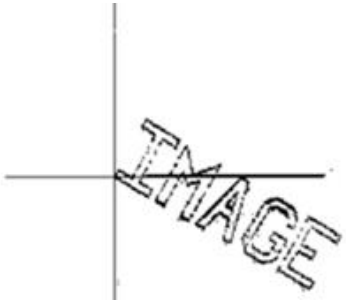
- All transformations are ‘with respect to’ the origin
- The point at position (0,0) does not move for scaling
- The point at position (0,0) does not move for rotation

Identity $\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$

$\begin{bmatrix} 0.5 & 0 \\ 0 & 2 \end{bmatrix}$



$\begin{bmatrix} 1.5 & 0 \\ 0 & 1.5 \end{bmatrix}$



Clockwise rotation $\begin{bmatrix} 0.866 & 0.5 \\ -0.5 & 0.866 \end{bmatrix}$

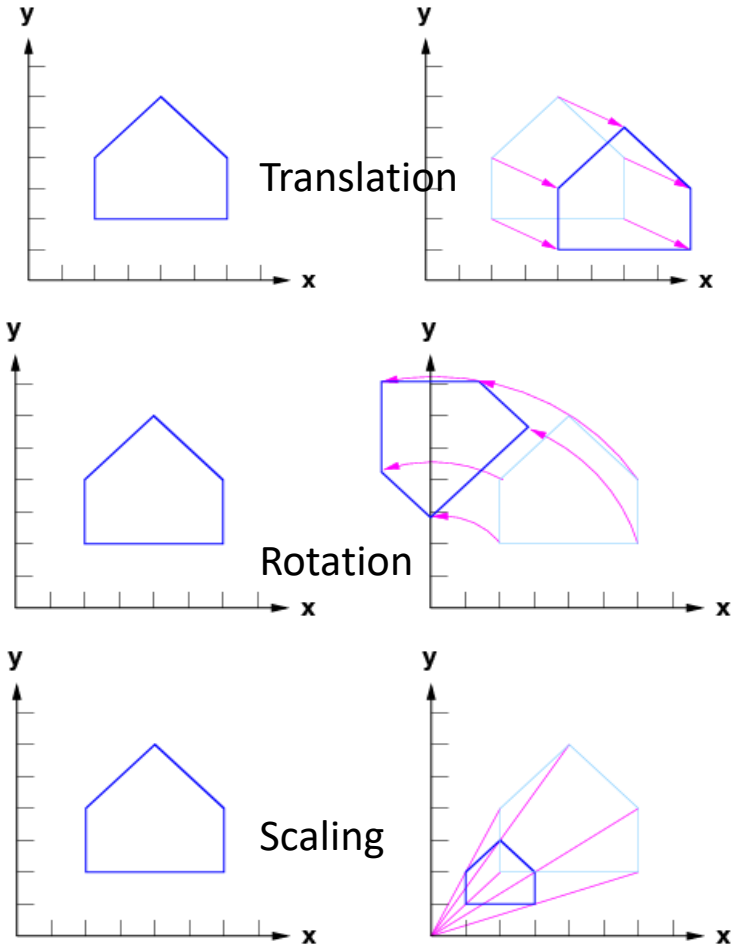
A.5 Summarising…

- A vertex is represented as a vector
- Transformation is achieved using matrix arithmetic

$$\begin{pmatrix} q_x \\ q_y \end{pmatrix} = \begin{pmatrix} a & b \\ c & d \end{pmatrix} \begin{pmatrix} p_x \\ p_y \end{pmatrix} + \begin{pmatrix} t_x \\ t_y \end{pmatrix}$$

New position	Rotate and/or Scale	Old position	Translate

- *Issue:* Translation is a separate matrix operation



A.6 Homogeneous coordinates

- (x, y) → (wx, wy, w) for any constant w<>0
- The vertex representation is augmented with an extra '1': (x, y, 1)
- The matrix representation becomes 3x3 for a 2D system

$$M = \begin{bmatrix} s_x & 0 & 0 \\ 0 & s_y & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

scale

$$M = \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

rotate (anti-clockwise)

$$M = \begin{bmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{bmatrix}$$

translate

- And we have $q = Mp$ where $M = \begin{bmatrix} a & b & t_x \\ c & d & t_y \\ 0 & 0 & 1 \end{bmatrix}$

A.7 Translation can be done as matrix multiplication

- Instead of:

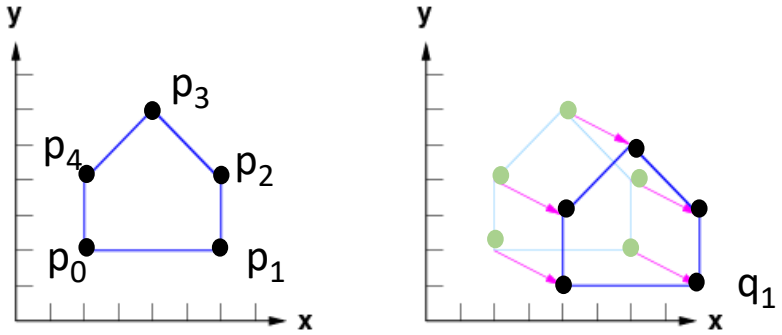
$$\begin{pmatrix} q_x \\ q_y \end{pmatrix} = \begin{pmatrix} a & b \\ c & d \end{pmatrix} \begin{pmatrix} p_x \\ p_y \end{pmatrix} + \begin{pmatrix} t_x \\ t_y \end{pmatrix}$$

- We can use: $q = Mp$ where $\mathrm{M} = \begin{pmatrix} \mathrm{a} & \mathrm{b} & \mathrm{t_x} \\ \mathrm{c} & \mathrm{d} & \mathrm{t_y} \\ 0 & 0 & 1 \end{pmatrix}$

- Example, for p1, (6,2,1):

$$q_1 = \begin{bmatrix} 1 & 0 & 2 \\ 0 & 1 & -1 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 6 \\ 2 \\ 1 \end{bmatrix}$$

$$q_1 = \begin{bmatrix} 8 \\ 1 \\ 1 \end{bmatrix}$$



A.8 Composing transformations

- Unified view: multiply matrices.
- Given two transformations A and B, and a set of points p, the new set of points q is:

$$q = BAp$$
$$M = BA$$
$$q = Mp$$

- **Note**: *Order matters*: AB is generally not the same as BA

Order matters

- Consider scaling (A) and translating (B) a set of points

$$\begin{matrix} M = BA \\ M = AB \end{matrix} \longrightarrow q = Mp$$

- Scale then translate, M = BA

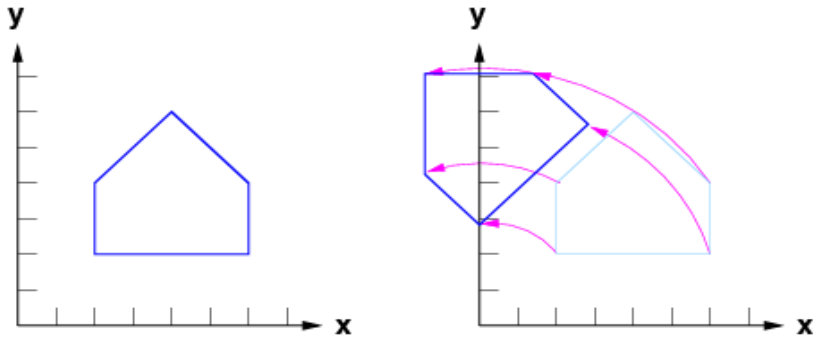
$$M = \begin{bmatrix} 1 & 0 & 2 \\ 0 & 1 & -1 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 2 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 2 & 0 & 2 \\ 0 & 2 & -1 \\ 0 & 0 & 1 \end{bmatrix}$$

- Translate then scale, M = AB

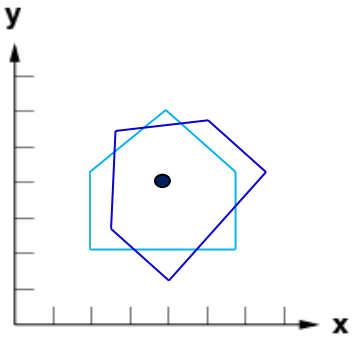
$$M = \begin{bmatrix} 2 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 2 \\ 0 & 1 & -1 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 2 & 0 & 4 \\ 0 & 2 & -2 \\ 0 & 0 & 1 \end{bmatrix}$$

A.9 Rotation about an arbitrary point

- Rotation matrix rotates points about the origin of the coordinate system



- If we want to rotate about an arbitrary point, we need to use a combination of transformations



A.10 Rotation about an arbitrary point

- *General plan*: Translate to world origin (T), rotate (R), and translate back again (T^{-1})

$$q_i = T^{-1}\left(R(T\,p_i)\right)$$

$$q_i = (T^{-1}\,R\,T)\,p_i$$

- Combining these:

$$M = T^{-1}RT = T^{-1}(RT)$$

$$q_i = M\,p_i$$