Part IA Graphics Notes

1 Background

1.1 Images

- An image is a two dimensional function, with each point having an intensity and colour
- A digital image is a sampled and quantised version of the real image
  - Limits the number of different intensities that can be stored
  - Each value in a digital image is called a pixel
  - Sampling resolution is usually measure in pixels-per-inch or dots-per-inch
- For human consumption 8 bits of intensity is sufficient however some applications use 10, 12 or 16 bits
- Colour is usually stored as 3 numbers of 5-16 bits each

1.2 Human Vision

- The retina is an array of light detection cells
  - The retina preprocesses the signals: averaging them, performing colour signal processing and local edge detection
  - Two types of light detecting cell, rods (light intensity) and cones (colour)
  - Three types of cones sensitive to short, medium and long wavelengths
  - Far fewer short wavelength receptors, 3:2 ratio long:medium
- The fovea is the high resolution area of the retina
  - Cones densely packed, providing a high resolution
  - Mostly rods outside the fovea: lower resolution, principally monochromatic and provides peripheral vision
- The optic nerve takes signals from the retina to the visual cortex in the brain
- Cones
  - The three types of cones have a corresponding response function $l(\lambda)$ (long), $m(\lambda)$ (medium) and $s(\lambda)$ (short)
  - The overall intensity response (y) of the cones can be calculated, where $P(\lambda)$ is the function of the incident light and $r(\lambda)$ is the cone's response function

$$y(\lambda) = l(\lambda) + m(\lambda) + s(\lambda)$$

$$y = k \int P(\lambda) y(\lambda) d\lambda$$

    or

$$y = k \int P(\lambda) r(\lambda) d\lambda$$

  - The following are sent to the brain

$$luminance = long + medium + short$$

$$long - medium = red\text{-}green$$

$$long + medium - short = yellow\text{-}blue$$

  - The response values for the three cone types are calculated as follows

$$l = k \int P(\lambda) l(\lambda) d\lambda \qquad m = k \int P(\lambda) m(\lambda) d\lambda \qquad s = k \int P(\lambda) s(\lambda) d\lambda$$

  - As a result of these equations different spectra can produce the same result in the brain, meaning it can be fooled
  - Different amounts of red, green and blue lights can generate a wide range of responses in the human eye

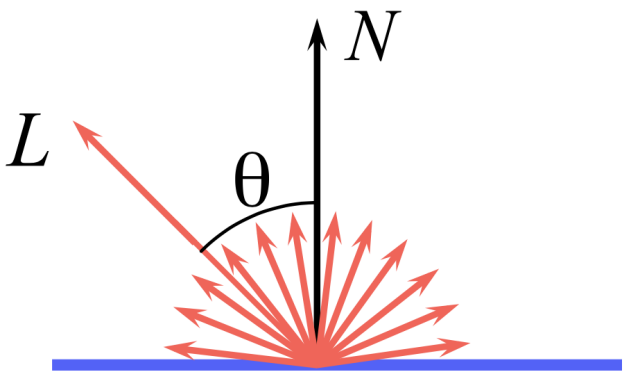
1.3 Storing Images

- 8-bits is the de facto standard for greyscale image, 16 also used
- 24 bits used for colour images
  - 3 bytes, one red, green and blue
- Often stored with each colour in a different plane
  - Planes can store other information, eg. alpha plane or z-buffer
- The frame being displayed currently is stored in the frame buffer
  - Consists of DRAM (sometimes called Video RAM or VRAM)

2 Rendering

2.1 Reflection and Shading

- The colour of a pixel depends on the lighting, shadows and the properties of the surface material
- Surfaces reflect in two different ways: Specular reflection and diffuse reflection (Lambertain reflection)
- The surface of a an object can absorb some wavelengths of light
- The wavelengths reflected specularly and diffusely by a surface can be different
- When calculating the shading of a surface the following assumptions are made
  - The is only diffuse reflection
  - All light falling on a surface comes directly from a light source - Cured by approximating specular reflection
  - No object casts a shadow on any other - Cured using shadow maps or ray tracing
  - Light sources are considered to be infinitely distant - Cured by adding local lights at the expense of more calculations
- The colour of a flat surface will be uniform across it, dependant only on the colour and position of the object and light sources
- **Diffuse Shading**



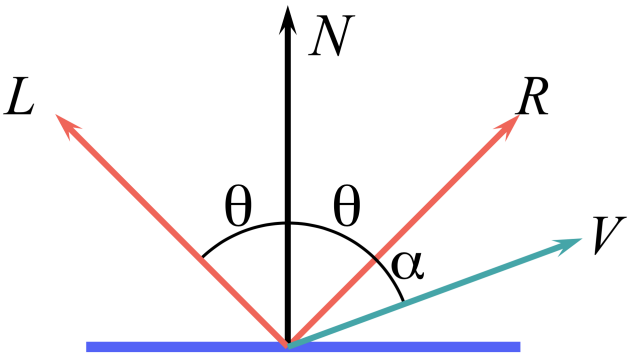
- Is calculated using the following equation

$$\begin{aligned} I &= I_l k_d cos\theta \\ &= I_l k_d (N \cdot L) \end{aligned}$$

  Where

  * L is the normalised vector pointing to the light source
  * N is the normal to the surface
  * I_l is the intensity of the light source
  * k_d is the proportion of light that is diffusely reflected by the surface
  * I is the intensity of the light reflected by the surface
- Can have different I_l and k_d for different wavelengths
- If $cos\theta < 0$ then the light is behind the plane
- One sided only the side in the direction of the normal vector can be illuminated, if $cos\theta < 0$ then both sides are black
- Two sided the sign of $cos\theta$ determines which side of the polygon is illuminated, need to invert sign of the intensity for the back side

- **Specular Reflection**



- Is calculated using the following equation

$$\begin{aligned} I &= I_l k_s cos^n \alpha \\ &= I_l k_s (R \cdot V)^n \end{aligned}$$

  Where

  - L is a normalised vector in the direction of the light source
  - R is the vector of perfect reflection
  - N is the normal to the surface
  - V is a normalised vector pointing at the viewer
  - I_l is the intensity of the light source
  - k_d is the proportion of the light that is reflected specularly by the surface
  - n is Phong's roughness coefficient
  - I is the intensity of the specularly reflected light

- Combined into one equation

$$I = I_a k_a + \sum_i I_l k_d (N \cdot L) + \sum_i I_l k_s (R \cdot V)^n$$

  Which summed over all light sources in the scene

2.2 Ray Tracing

- Given a set of 3D objects shoot a ray from the eye through the centre of each pixel and see what surface it hits
- Computational expensive
- Easily handles reflection, refraction, shadows and blur
- The basic algorithm

```
for (every pixel in the screen plane) {
    Determine ray from eye through centre of pixel
    for (each object in scene) {
        if (ray intersects object) {
            if (the intersection is the closest (so far) to the eye) {
                record intersection point and object
            }
        }
    }
    Calculate and set the pixel's colour
}
```

- The equation of a ray

$$P = O + sD,\ s \geq 0$$

- The equation of a plane

$$P \cdot N + d = 0$$

- The equation of a sphere

$$(P - C) \cdot (P - C) - r^2 = 0$$

- Solve the equation so to find if and where the ray and object intersect
  - For the plane

$$s = \frac{d + N \cdot O}{N \cdot D}$$

  - For a sphere

$$
\begin{aligned}
a &= D \cdot D \\
b &= 2D \cdot (O - C) \\
c &= (O - C) \cdot (O - C) - r^2 \\
d &= \sqrt{b^2 - 4ac} \\
s &= \begin{cases} \frac{-b \pm d}{2a}, & d \in \mathbb{R} \\ No\ intersection, & d \notin \mathbb{R} \end{cases}
\end{aligned}
$$

- Once the closest intersection has been found the colour of the surface must be calculated.
  - Calculate the normal to the plane at the intersection point
  - Shoot rays from that point to all of the light sources and calculate the specular and diffuse reflection
  - For each of these rays test if they intersect another objects, if they do then the object is in shadow
  - If a surface is totally or partially reflective then new rays are spawned recursively to find the contribution of the reflection
- Objects can be transparent, in this case rays pass through them
  - A refractive index can be modelled by altering the direction of rays
- If one ray through the centre is used then aliased form, these are
  - Edges are stepped
  - Small objects are missed completely
  - Thin objects are missed or split into small pieces
- Forms of anti-aliasing
  - **Super-sampling** - Send multiple rays through each pixel and average the result, comes in different types
    - Regular grid - Divide the pixel into sub-pixels and average, can still result in aliasing unless a lot of sub-pixels are used
    - Random - Shoot N rays at random points through the pixel, replaces aliasing with noise artifacts which the eye doesn't pickup as well
    - Poisson disk - Shoot N rays randomly but no two rays can be closer than ϵ, better than pure random but very hard to implement
    - Jittered - Divide pixel into N sub-pixels and shoot one ray through a random point in the sub-pixel, approximation of Poisson disk, better than purely random sampling and easy to implement
  - **Adaptive super-sampling** - Send a few rays and test the difference between their results and fire more if it's too large
  - Distributed ray tracing allows for a number of effects
    - Distributed over the area of a pixel allows for anti-aliasing
    - Distributed rays going to a light source allows area light sources and soft shadows
    - Distribute the camera position over some area allows for simulation of a camera with a finite focal length
    - Distribute samples over time allows for motion blur
    - Distribute rays to calculate specular reflection, can calculate effect of non-perfect reflection, need many rays per pixel

- Indirect illumination
  - Diffuse to specular - handled by distributed ray tracing
  - Specular to specular - handled by distributed ray tracing
  - Diffuse to diffuse - Handled by radiosity
  - Specular to diffuse - No usable algorithm
- Ray tracing is computationally expensive

3 Graphics Pipeline

3.1 Polygon mesh models

- Shapes are converted in to polygons, triangles are used as they must be planar
- GPUs are optimised to work with triangles
- Homogeneous 3D coordinates are used

$$(x,\ y,\ z,\ w) \equiv \left(\frac{x}{w},\ \frac{y}{w},\ \frac{z}{w}\right)$$

3.2 Transformations

- Translations are not commutative
- **Scale** - Around the origin by factor m_x in x and m_y in y

$$\begin{bmatrix} x' \\ y' \\ w' \end{bmatrix} = \begin{bmatrix} m_x & 0 & 0 \\ 0 & m_y & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ w \end{bmatrix}$$

- **Rotate** - About the origin by angle θ

$$\begin{bmatrix} x' \\ y' \\ w' \end{bmatrix} = \begin{bmatrix} cos\theta & -sin\theta & 0 \\ sin\theta & cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ w \end{bmatrix}$$

- **Shear** - Parallel to the x axis by factor a

$$\begin{bmatrix} x' \\ y' \\ w' \end{bmatrix} = \begin{bmatrix} 1 & a & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ w \end{bmatrix}$$

- **Translation** - Translation by x_0 and y_0

$$\begin{bmatrix} x' \\ y' \\ w' \end{bmatrix} = \begin{bmatrix} 1 & 0 & x_0 \\ 0 & 1 & y_0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ w \end{bmatrix}$$

- Rotations in 3D can happen about different axis
  - Z axis

$$\begin{bmatrix} x' \\ y' \\ z' \\ w' \end{bmatrix} = \begin{bmatrix} cos\theta & -sin\theta & 0 & 0 \\ sin\theta & cos\theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ w \end{bmatrix}$$

- Y axis

$$\begin{bmatrix} x' \\ y' \\ z' \\ w' \end{bmatrix} = \begin{bmatrix} cos\theta & 0 & -sin\theta & 0 \\ 0 & 1 & 0 & 0 \\ sin\theta & 0 & cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ w \end{bmatrix}$$

- X axis

$$\begin{bmatrix} x' \\ y' \\ z' \\ w' \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & cos\theta & -sin\theta & 0 \\ 0 & sin\theta & cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ w \end{bmatrix}$$

Model Transformation

- Transforms an model into the desired size and position
- Order:
  1. Scale
  2. Rotate
  3. Translate
- Work backwards when rotating and take the inverse

Perspective Projection

- Transforms a 3D world into a 2D image
- **Parallel** - $(x,\ y,\ z) \mapsto (x,\ y)$
- **Perspective** - $(x,\ y,\ z) \mapsto (x\frac{x}{z},\ y\frac{y}{z})$
- Projection is a matrix operation

$$\begin{bmatrix} x' \\ y' \\ \frac{1}{d} \\ \frac{z}{d} \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & \frac{1}{d} \\ 0 & 0 & \frac{1}{d} & 0 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

- We assume:
  - Screen centred at $(0,\ 0,\ d)$
  - Screen parallel to the xy-axis
  - z-axis through screen
  - y-axis up and x-axis to the right
  - Camera at origin
- Camera specified in world coordinated
  - Eye at $(e_x,\ e_y,\ e_z)$
  - Look point (centre of screen) at $(l_x,\ l_y,\ lz)$
  - Up along vector $(u_x,\ u_y,\ u_z)$
  - Up perpendicular to vector $\overrightarrow{el}$
- Order of transformations
  1. Translate eye to origin

2. Scale so $\overrightarrow{el}$ distance is d
3. Align $\overrightarrow{el}$ with z axis
   (a) Rotate around z axis so is in positive xy plane
   (b) Rotate around x axis so is aligned with z axis
4. Align up vector with the positive y-axis

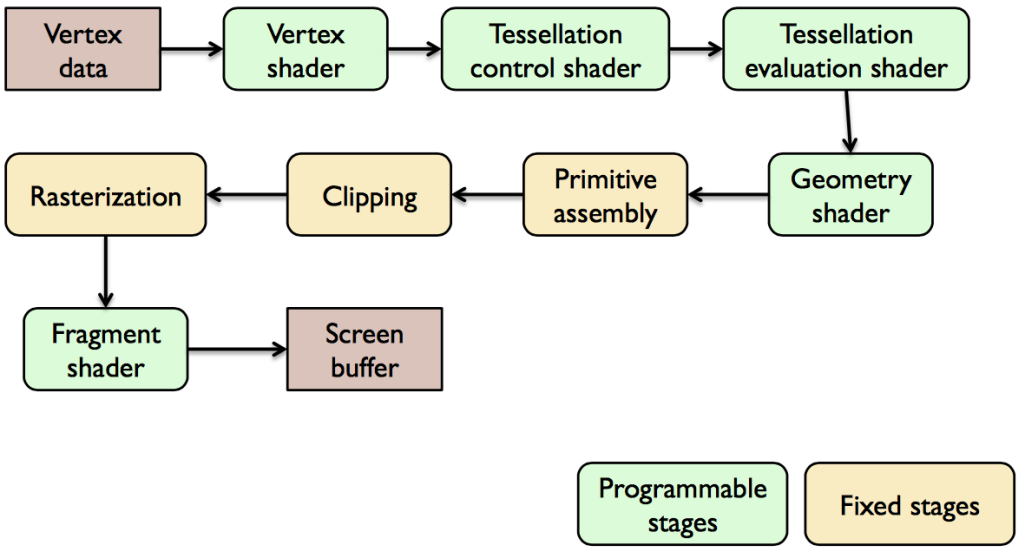
- All transformations can be pre-multiplied together

3.2.1 Illumination and Shading

- Gouraud Shading
  - Calculate the diffuse illumination at each vertex
  - The normal of each vertex is used to calculate the normal
  - Colour is interpolated across the polygon
- Phong Shading
  - Normals are interpolated across a polygon
  - Interpolated normals are used to calculated specular reflection

4 Graphics Pipeline

4.1 OpenGL Pipeline



- **Vertex Shader** - Processing of vertices, normals and UV texture coordinated
- **Tessellation Control Shader and Tessellation Evaluation Shader** - (Optional) Create new primitives by tessellating existing primitives
- **Geometry Shader** - (Optional) Operate on tessellated geometry. Can create new primitives
- **Primitive Assembly** - Organises vertices into primitives and prepares them for rendering
- **Clipping** - Remove or modify vertices so that they all lie within the viewport (view frustum)

- **Rasterisation** - Generate fragments to be drawn for each primitive. Interpolates vertex attributes
- **Fragment Shader** - Computes colour per each fragment, can lookup colour in the texture or modify pixel's depth value

4.2 Buffers

- Double buffer - One buffer displayed, the other written to. Buffers swapped every 1/60 of a second
- Triple buffer - Cycles through the buffers, larger delay between writing and displaying
- If buffers swapped while the elements are being drawn to the screen the image will tear - solved by V-Sync

5 Technology

5.1 Colour Spaces

- Munsell's artist's colour classification
  - Three axis
    - Hue - Dominant colour
    - Value - Brightness
    - Chroma - Vividness/dullness
  - Highly irregular
- Three standard primaries X, Y, Z
- The CIE chromaticity diagram
  - Used to define a colour space
  - Pure colours lie on the outer curve
  - Points outside the curve don't exist
  - Chromacitity is defined in terms of x, y, z

$$x = \frac{X}{X+Y+Z}, \qquad y = \frac{Y}{X+Y+Z}, \qquad z = \frac{Z}{X+Y+Z}$$

5.2 Outputs

- Liquid Crystal Displays
  - Twist the polarisation of light
  - Can become transparent or opaque
  - colour is achieved using filters
- Digital Micromirror Devices
  - Reflect light differently depending on the luminance of a pixel
- Electrophorectic Displays
  - Small transparent capsules
  - Filled with dark oil and white particles
  - Current attracts or repels particles changing colour of display
- Printers

  - Ink jet - Sprays ink onto paper
  - Laser - Uses a laser to charge a drum, this picks up charged toner which is pressed on paper
  - Commercial offset - Image of the whole page is put on a roller; repeatedly inked and pressed on paper, print thousands of the same thing
- Half toning - Divide image into cells and draw a spot in each, the size of the spot varying
- Dithering - Noise is deliberately added, more realistic but photocopies badly