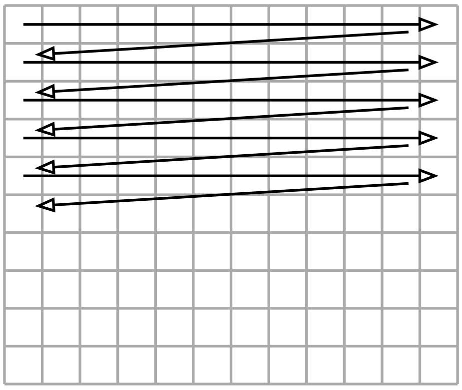
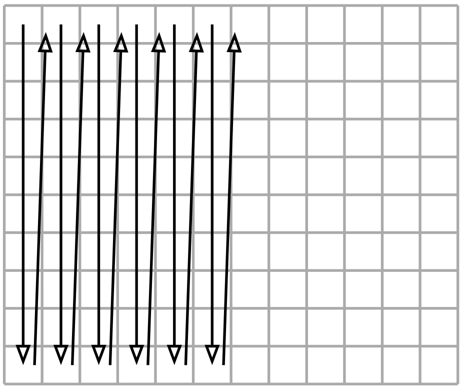
Introduction to Graphics

Image Representation

Row Major $i(x, y) = x + y \cdot n_{\text{cols}}$

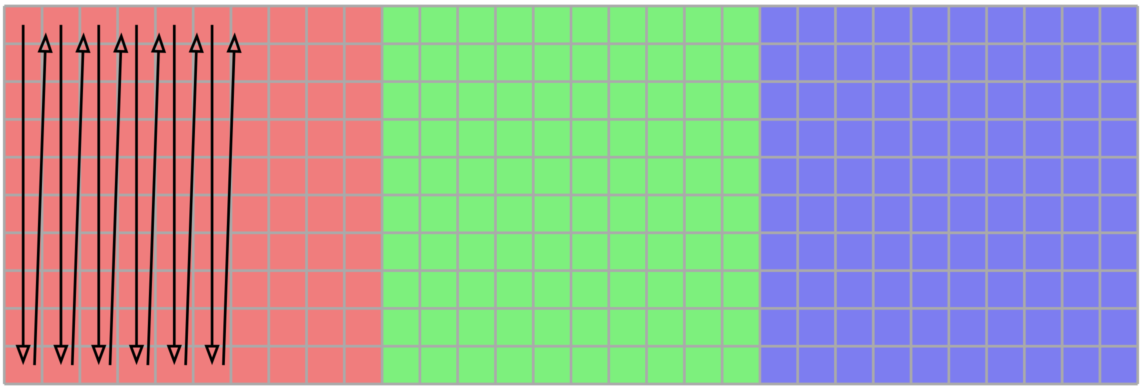


Column Major $i(x, y) = x \cdot n_{\text{rows}} + y$



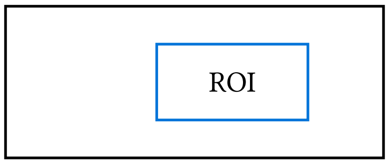
RGB representations:

- **Interleaved row major** $i(x, y, c) = 3x + 3y \cdot n_{\text{cols}} + c$ all colours of a pixel next to each other.
- **Planar column major** $i(x, y, c) = x \cdot n_{\text{rows}} + y + cxy$



Padded images are used if an algorithm requires all pixels have neighbouring pixels.

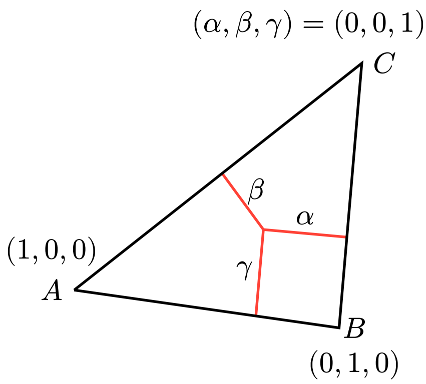
$$i(x, y, c) = i_{\text{first pixel}} + xs_x + ys_y + cs_c$$



- **Colour banding** is visible when there are not enough bits to represent colour.
- **Dithering adds noise** to reduce banding.

Ray Tracing

Barycentric coordinates (α, β, λ) gives a point in a triangle if $0 \leq \alpha, \beta, \gamma \leq 1$ where $P = \alpha A + \beta B + \gamma C$



Finding Intersection

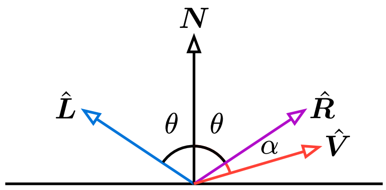
- Ray-sphere: solve for s in $\left(s\hat{\boldsymbol{d}} + (\boldsymbol{o} - \boldsymbol{c})\right)^2 - r^2 = 0$

and choose the closer solution

- Ray-plane: $\boldsymbol{s} = -(\boldsymbol{a} + \boldsymbol{n} \cdot \boldsymbol{o}) / \left(\boldsymbol{n} \cdot \hat{\boldsymbol{d}}\right)$
- Ray-triangle: additionally check barycentric coordinate.

$$\text{ray} : \boldsymbol{r} = \boldsymbol{o} + s\hat{\boldsymbol{d}}$$
$$\text{plane} : \boldsymbol{r} \cdot \boldsymbol{n} + \boldsymbol{a} = 0$$
$$\text{sphere} : (\boldsymbol{r} - \boldsymbol{c})^2 = r^2$$

Phong's Shading Algorithm



$$I_{\text{specular}} = I_l k_s \cos^n \alpha = I_l k_s \left(\hat{\boldsymbol{R}} \cdot \hat{\boldsymbol{V}}\right)^n$$

- I_l the light intensity
- k_s proportion of light reflected specularly
- n is the roughness factor of the surface.

$$\begin{aligned} I_{\text{total}} &= \text{ambient} + \text{diffuse} + \text{specular} \\ &= I_a k_d + \sum_i I_i k_d \hat{\boldsymbol{L}} \cdot \widehat{\boldsymbol{N}} + \sum_i I_i k_s \left(\hat{\boldsymbol{R}} \cdot \hat{\boldsymbol{V}}\right)^n \end{aligned}$$

Sampling Method	**Description**
Single point sampling	Samples at the center of pixels
Super sampling	Goal is the remove artifact • Random sampling: samples random points in a pixel • Poisson disc samping: reject rays less than distance ε from each other. • Jitter sampling: divide pixel into a grid, then random sample on each cell
Distributed sampling	Achieve effects such as antialiasing, area light, depth of field, motion blur.

Rasterisation

1. Model surface as polyhedrons (connected polygon surfaces)
2. Apply transformations to project plane on screen
3. Fill pixels with colour of the nearest visible polygon

Let homogenous coordinates (x, y, w) represent $(x/w, y/w)$ in 2D cartesian coordinates.

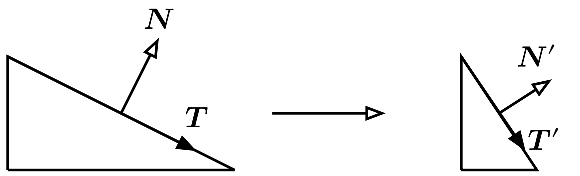
2D Transformation	***Homogenous* Matrix**
Scale by m	$\begin{pmatrix} m & 0 & 0 \\ 0 & m & 0 \\ 0 & 0 & 1 \end{pmatrix}$
Rotate by θ	$\begin{pmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{pmatrix}$
Translate by (x, y)	$\begin{pmatrix} 1 & 0 & x \\ 0 & 1 & y \\ 0 & 0 & 1 \end{pmatrix}$

3D Transformation	***Cartesian* Matrix**
Rotate by θ about x axis	$\begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos\theta & -\sin\theta \\ 0 & \sin\theta & \cos\theta \end{pmatrix}$
Rotate by θ about y axis	$\begin{pmatrix} \cos\theta & 0 & \sin\theta \\ 0 & 1 & 0 \\ -\sin\theta & 0 & \cos\theta \end{pmatrix}$
Rotate by θ about z axis	$\begin{pmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{pmatrix}$

Model-View-Projection Transformations

1. To transform a cylinder at origin with radius 1, height 1, oriented in direction of $(0, 0, 1)$
   i) Apply scale S
   ii) Apply rotation R
      - Find R_1 which orients desired cylinder in direction $(0, y, z)$
      - Find R_2 which orients desired cylinder in direction $(0, 0, 1)$
      - $R = (R_1)^{-1}(R_2)^{-1}$
   iii) Apply translation T

Find transformation G so $\boldsymbol{N}'$ is normal to $\boldsymbol{T}'$



$$T' = MT \qquad N' = GN$$

$$(GN) \cdot (MT) = 0$$

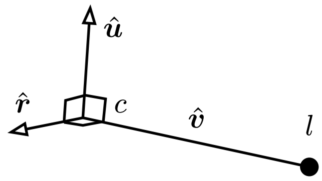
$$N^T G^T MT = 0 \quad (T \text{ is transpose})$$

If $G^T M = I$ then $N^T G^T MT = N^T T = 0$, so $G = M^T$.

To attach object B to A, make a **scene graph**.

i) Apply scale to A
ii) Apply scale, rotation and translation to move B to where it will attach to A
iii) Apply rotation and translation to both A and B

2. Transform objects to a viewing coordinate system:
   - Before: camera centred at $\boldsymbol{c}$, directed at $\boldsymbol{l}$, the direction of up is $\boldsymbol{u}$
   - After: camera centred at origin, directed at $(0, 0, 1)$, up is $(0, 1, 0)$

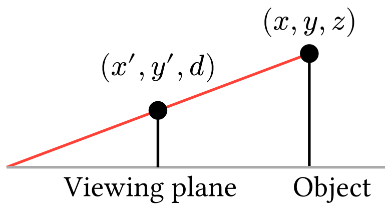


$$\hat{\boldsymbol{v}} = \frac{l - c}{|l - c|} \qquad \hat{\boldsymbol{r}} = \frac{\boldsymbol{u} \times \hat{\boldsymbol{v}}}{|\boldsymbol{u} \times \hat{\boldsymbol{v}}|} \qquad \hat{\boldsymbol{u}} = \hat{\boldsymbol{v}} \times \hat{\boldsymbol{r}}$$

$$\text{viewing coordinates} = \begin{pmatrix} \hat{\boldsymbol{r}}_x & \hat{\boldsymbol{r}}_y & \hat{\boldsymbol{r}}_z & 0 \\ \hat{\boldsymbol{u}}_x & \hat{\boldsymbol{u}}_y & \hat{\boldsymbol{u}}_z & 0 \\ \hat{\boldsymbol{v}}_x & \hat{\boldsymbol{v}}_y & \hat{\boldsymbol{v}}_z & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 & -\boldsymbol{c}_x \\ 0 & 1 & 0 & -\boldsymbol{c}_y \\ 0 & 0 & 1 & -\boldsymbol{c}_z \\ 0 & 0 & 0 & 1 \end{pmatrix} \times \text{scene coordinates}$$

3. Projection to viewing plane.

$$\begin{pmatrix} x' \\ y' \\ z' \\ w' \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1/d \\ 0 & 0 & 1/d & 0 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix} = \begin{pmatrix} x \\ y \\ 1/d \\ z/d \end{pmatrix}$$



Corresponding to $(xd/z, yd/z, 1/z)$, the z component is used for z-buffer.

The **rasterisation algorithm** goes as:

0. Initialise **colour buffer** with background colour
1. Find the MVP transformation matrix
2. Apply the matrix to the vertices of all triangles
3. For each **fragment** (candidate pixel), interpolate attributes with barycentric coordinates.
4. If fragment is closer to camera than pixels drawn so far, update **colour buffer** with fragment colour, and set value of **depth buffer**.

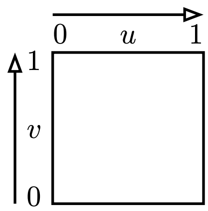
The OpenGL Pipeline

Step name	**Description**
Vertex shader	Apply MPV transformations to all vertices
Tesselation shader	Conditionally split surface into more polygons
Geometry shader	Create new geometry (e.g. fur and volumes)
(Non-programmable)	Clipping and rasterisation
Fragment shader	E.g. phong's shading algorithm

Shaders are written in **GLSL** and runs in parallel in the GPU.

Operations on aggregated types (e.g. `vec4`) are almost as fast as single values.

There are 1D, 2D and 3D textures. OpenGL uses a UV-map, a (u, v) coordinate is defined for each vertex so by interpolation, every surface point gets a value.



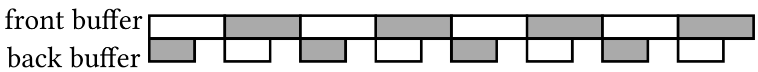
- Upscaling:
  - **Nearest neighbour** (blocky artifact)
  - **Bilinear interpolation** (blurry artifact)
- Downsampling:
  - **Area averaging**: averages the *texels* the pixel covers (slow)
  - **Mipmap**: stores textures in multiple resolution to avoid recalculation

Textures can be tiled so it wraps around, e.g. $T(2, 1) = T(1, 1) = T(0, 1)$

Mapping	Description
Bump mapping	Changes normal to affect shading
Displacement mapping	Changes the shape of an object
Environment mapping	Texture with infinite distances from the source of reflection, e.g. sky box

The **back buffer** is the one the GPU draws to, the **front buffer** is displayed to the screen.

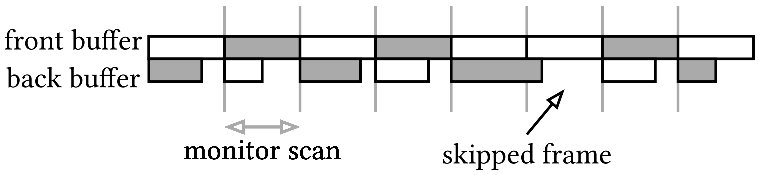
Call **swap** when done drawing.



Use 3 buffers so the GPU always have a buffer to draw to.

- **Tearing** happens if swap is called when the pixels are still being copied from buffer to screen.

  Vsync makes the GPU wait for a refresh cycle to complete before swapping.



- **Variable refresh rate** allows the GPU to control the timing of frames.

Human Vision

3 types of cone cells S, M, L are responsible for colour vision. For a particular light, the cone response for S is $R_s = \int L(\lambda) d\lambda$ where the light intensity with wavelength λ is $L(\lambda)$.

A percepted colour is entirely characterised by R_s, R_m, R_l.

- **Standard dynamic range** image encode only the colours that the display can show.
- **High dynamic range** tries to encode all visible wavelengths so the screen (OLED, laser, etc) can accurately display the encoded information.

There are matricies to convert between different SDR and HDR encodings.

Tone Mapping

- **Luma** is gamma corrected greyscale brightness.
- **Exposure** changes scene white

The **sigmoidal tone curve** mimics film.

$$R' = \frac{R^b}{(L_m/a)^b + R^b}$$

L_m is the median colour, a shifts the curve left and right, b is the steepness of the curve (contrast).