# Part IB Further Graphics Notes

## 1 Ray tracing

- Choose the colour of the pixel by firing a ray through and see what if anything it hits

### 1.1 GPU ray tracing

1. Use a vertex shader with no transformations
2. Set up OpenGL's geometry with only one quad taking up the entire view port
3. Bind coordinates to each vertex, have OpenGL interpolate coordinates for each pixel
4. Implement ray tracing in GLSL

#### 1.1.1 Ray marching

- Take a series of finite steps along the ray seeing if you've intersected something or exceeded the maximum number of steps
- Too large a step and step over objects
- An if statement within a for loops is very difficult for the GPU to optimise

#### 1.1.2 Signed distance fields

- Can give ray tracing good real time performance
- Define the function $d(p) =$ [distance to nearest object]
- At each step calculate the function $d$ and advance this far
- Define some $\epsilon$ such that if $d(p) < \epsilon$ then the ray has intersected that object
- A signed distance field measures the distance a point is from the surface it describes, a negative value indicates that the point is within the object
- Signed distance fields can be combined in the following ways
  - The union is given by the `min()` of the fields
  - The intersection is given by the `max()` of the fields
  - The difference, $A - B$ is given by `max(A, -B)`
  - To smoothly interpolate two surfaces use a polynomial function to give a smooth distance curve instead of discontinuous one
- To transform a singed distance field apply the inverse of the transform the input point
- Can apply a nonglobal transformation to the points, eg. a function which changes with the height of the point
- The normal to a signed distance field can be approximating by finding the gradient of the field

```
float d = getSDF(pt);
vec3 normal = normalize(vec3(
    getSDF(pt.x - ϵ, pt.y, pt.z) - d,
    getSDF(pt.x, pt.y - ϵ, pt.z) - d,
    getSDF(pt.x, pt.y, pt.z - ϵ) - d,
));
```

- Shadows
  - Trace a ray towards the light sources, if a signed distance field ever drops too close to zero then it is occluded
  - Get soft shadows by making the intensity of illumination a function of the minimum value of a signed distance field on the path.
- Repeat geometry by taking the mod of the point's coordinates along one or more of the axis

## 1.2 Detecting collisions

### 1.2.1 Vector method

- Rays are define parametrically as $P(t) = E + tD$ where $E$ is the coordinates of the eye and $D$ is the ray's normalised direction
- A polygon is a set of points $\{v_1, ..., v_n\}$
- The normal to the polygon is therefore $N = (v_n - v_1) \times (v_2 - v_1)$
- The equation for the plane is therefore $N \cdot (p - v_1) = 0$
- Subbing in the equation of a point gives $N \cdot (E + tD - v_1) = 0$
- This can then be rearanged to give

$$t = \frac{N \cdot v_i - N \cdot E}{D \cdot N}$$

### 1.2.2 Half-planes method

- Only works for convex polygons
- A half-plane is the set of all points which lie on one side of a line
- Each edge defines a half-plane covering the polygon
- If a point lies within all half-planes then it is within the polygon
- For each edge $v_i \to v_{i+1}$ rotate 90° anticlockwise about the normal to give $e_R$
- $e_R$ can easily be calculated by crossing $N$ with $e$
- If $e_R \cdot (P - V_i) < 0$ then the point does not lie within the polygon

## 1.3 Barycentric coordinates

- Defines points within a triangle
- Points are given three coordinates $(t_A, t_B, t_C)$
- If all of the coordinates are not the same sign then the point lies outside the triangle
- For a triangle with vertices $A$, $B$,$C$, barycentric coordinates $(t_A, t_B, t_C)$, and Cartesian coordinates $(x, y)$

$$x = \frac{(x_A \cdot t_A) + (x_B \cdot t_B) + (x_C \cdot t_C)}{t_A + t_B + t_C}$$

$$y = \frac{(y_A \cdot t_A) + (y_B \cdot t_B) + (y_C \cdot t_C)}{t_A + t_B + t_C}$$

## 1.4 Further effects

### 1.4.1 Shadows

- Fire a ray from $P$ to each light $L_i$ if it hits anything the $L_i$ does not contribute to the illumination of $P$ - this gives a hard edge
- Soft shadows
  - Give each light source a size, fire multiple rays from $P$ randomly distributed across the light source's area
  - The intensity of the illumination is a function the proportion of rays which reach the light source
  - The more rays, the smoother the result

### 1.4.2 Spotlights

- Create a spotlight shining along $S$ multiply the diffuse+secular term by $(max(L \cdot S, 0))^m$
- Increasing $m$ tightens the spotlight, but leaves the edges smooth
- Hard edged shadows us $((max(L \cdot S, 0) > cos(15°))\ ?\ 1 : 0)$

### 1.4.3 Transparency and refraction

- Create transparency by upon collision creating a new ray with $E_T = P$ and $D_T = D$
- To simulate refraction $D_T$ should be perturbed as defined by Snell's law

$$\frac{sin\theta_1}{sin\theta_2} = \frac{n_2}{n_1}$$

- Calculate the angle of incidence using the dot product of $D$ and $N$, to give

$$\theta_2 = sin^{-1}(\frac{n_1}{n_2}\ sin\ \theta_1)$$

- If $\left(\frac{n_1}{n_2}\ sin\ \theta_1\right) > 1$ then total internal reflection occurs
- At grazing angles some of the light is reflected, some refracted use the Fresnel equations to calculate this

### 1.4.4 Aliasing

- Attempting to sample a continuous function with a discrete function (pixels)
- Supersampling - fire multiple rays through each pixel and take the average
  - Rays can still clump together and cause aliasing
  - Can use jittered supersampling, split the pixel into sub pixels and fire a ray randomly into the pixel

## 1.5 Optimisations

### 1.5.1 Bounding volumes

- A bounding volume for a set of object is a volume which contains every object within that set
- Used to fast volumetric tests, eg. testing if a ray hits something
- Types of bounding volume

  - Axis-aligned bounding box - min and max of the x, y, and z coordinates
  - Bounding sphere - The maximum radius from the centre
- Can use bounding volumes in a hierarchy, allows you to discard parts of the scene which the ray will not reach

#### 1.5.2 Subdivision of space

- Split space into cells and for each store a list of all objects which overlap with it
- Rays can skip over empty boxes
- Cell size means that objects can overlap many cells or have lots of empty cells

#### 1.5.3 BSP trees

- BSP prepartitions the scene into objects in front of and behind a tree of planes
- By testing each plane in turn the set of objects can be found using a binary tree search
- Requires a slow preprocessing step and is therefore not very useful for nonstatic scenes

#### 1.5.4 kd-trees

- Similar to a BSP tree but all planes are aligned to the axis
- Build a binary search tree, where each node is a test of which side of a plane a point lies

#### 1.5.5 Bounding interval hierarchies

- Builds volumes around objects and then shrinks them to remove unused space
- Similar to kd-trees but the planes are moved to remove unused space
- Fast enough to be used in nonstatic scenes, is generated as a ray is fired

## 2 Computational Geometry

- Closed manifold - Exactly two triangles meet at each edge, and faces meeting at a vertex belong to a single connected loop of faces
- Manifold with boundary - At most two faces meet at each edge, faces meeting at a vertex belong to a single strip of faces
- A surface is oriented if every face is stored in a fixed order, and if vertices $i$, $j$ appear in faces $f1$ and $f2$ then in $f1$ they are in the order $i$, $j$ and in $f2$ they are $j$, $i$
- A surface is embedded if no vertex, edge, or face shares any point with anyother vertex, edge, or face
- A closed embedded surface must split 3D space into two parts, a bounded interior and unbounded exterior
- The genus of a surface is the number of nonintersecting closed curves that can be drawn on the surface without separating it
  - The number of coffee handles on it
- The Euler characteristic
  - Is a topological invariant

  - Poincaré formula, where $V$ is the number of vertices, $E$ is the number of edges, $F$ is the number of faces, $g$ is the genus, and $\chi$ is the Euler characteristic

$$V - E + F = 2 - 2g = \chi$$

  - The sum of the angle deficits of all vertices in a surface is $2\pi\chi$

$$\sum_S AD(v) = 2\pi\chi$$

- The Jordan curve theorem
  - Any closed cure separates the points on a plain into two distinct domains, with C as the boundary
  - This can be extended to a sphere, and any shape which is topologically equivelent to a sphere
- The medial axis of a surface is the set of all points on that surface that is closest to two or more points on the edge of the surface

Voronoi diagrams

- For a set of points $P$ divide the surface up into cells such that the cell associated with $P_i$ is closer to $P_i$ than any other point in $P$
- The Delaunay triangulation is the dual of the Voronoi diagram, it is a graph where each pair of points are only connected if their Voronoi cell's share an edge
- The set $P$ is called the generating point
- Where three or more edges meet is a Voronoi point, it is the centre of a circle which passes through the points associated with the incident edges
- The equiangularity of a triangulation of a set of points is a sorted list of all angles within that triangulation.
- The Delaunay triangulation is equiangular as it has the lexicographically largest equiangularity
- A circumcircle of a triangle is the unique circle such that all the triangle's vertices lie on the circle's circumference
- The empty circle property of a triangulation is that the circumcircles of all triangles within the triangulation can not contain a vertex from any other triangle in the triangulation.
- An n-dimensional Delaunay triangulation can be found by finding the (n+1)-dimensional convex hull and flattening it down to be n-dimensional (by removing the last coordinate)
- Can be calculated using Fortune's algorithm
  - Maintain a breach and a sweep line, advancing left-to-right
  - The breach line is the union of parabolas equidistant from the point and the sweep line
  - The intersection of parabola are edges of the voronoi diagram
  - Maintain a queue of events, namely the addition and removal of parabola

## 2.1 Normal to a vertex

- As a limit
  - The limit of the cross-product of two (non-colinear) vectors from $P$ to the points in $S$, the set of all points a distance $r$ from $P$ ,as $r$ tends to zero
- Weighted average
  - Take the a weighted average of the normals of each of the incident faces, weighted their face angle

### 2.2 Curvature

- Curvature is how unflat a surface is
- Gaussian curvature
  - The two directions a surface is curving most $k_1$ and $k_2$ are the principle curvature
  - The Gaussian curvature is the product of $k_1$ and $k_2$
  - The Gaussian curvature of a region is the ratio of the area of the surface produced by unit normal vectors of the region and the area of that region
  - The Gaussian curvature of any polyhedral mesh is zero except at the vertices where it is zero
- Angle deficit
  - Defined as $2\pi$ minus the sum of the face angle of all incident faces

$$AD(v) = 2\pi - \sum_F \alpha(F, v)$$

## 3 Bezier Curves

- A linear bezier curve, with control points $P_0$ and $P_1$, is defined as

$$P(t) = (1-t)P_0 + tP_1$$

- An nth order bevier curve can be found by linearly interpolating two (n-1)st order curves
- Bernstein polynomials for $0 \leq t \leq 1$ sum to 1

$$b_{v,n}(t) = \binom{n}{v} t^v (1-t)^{n-v}$$

- Can be drawn by drawing line segments for a fixed parameter $t$
  - Has the problem that the fixed parameter will not work well with every curve
- Adaptive method
  - Draw a straight line from $P_0$ and $P_n$
  - Tests if this is acceptable
  - If it is leave it, if not split in half and recurse
- A Bezier can be split into two curves, $Q$ and $R$, which when rendered will be the same as the original

$$\begin{aligned}
Q_0 &= P_0 \\
Q_1 &= \frac{1}{2}P_0 + \frac{1}{2}P_1 \\
Q_2 &= \frac{1}{4}P_0 + \frac{1}{2}P_1 + \frac{1}{4}P_2 \\
Q_3 &= \frac{1}{8}P_0 + \frac{3}{8}P_1 + \frac{3}{8}P_2 + \frac{1}{8}P_3
\end{aligned}$$

$$\begin{aligned}
R_0 &= \frac{1}{8}P_0 + \frac{3}{8}P_1 + \frac{3}{8}P_2 + \frac{1}{8}P_3 \\
R_1 &= \frac{1}{4}P_1 + \frac{1}{2}P_2 + \frac{1}{4}P_3 \\
R_2 &= \frac{1}{2}P_2 + \frac{1}{2}P_3 \\
R_3 &= P_3
\end{aligned}$$

- Can draw a Bezier curve using singed distance fields, with the function being the minimum distance to a line segment of the curve

- Overhause's Cubic is used to draw a bezier which passes through four provided points
  - For four data points $A$, $B$, $C$, and $D$ the curve has the following control points

$$\begin{aligned} P_0 &= B \\ P_1 &= B + \frac{C - A}{6} \\ P_2 &= C + \frac{D - B}{6} \\ P_3 &= C \end{aligned}$$

- Types of join
  - $C_n$ - mathematical continuity - continuous in all derivatives up to the $n^{\text{th}}$ derivative
  - $G_n$ - geometric continuity - each derivative up to the $n^{\text{nt}}$ has the same direction to its vector on either side of the join (but different magnitudes)
- To join two Bezier splines together with $C_1$ continuity need $P_3 = Q_0$ and $P_3 - P_2 = Q_1 - Q_0$
- B-Splines are an extension of Beziers which can have an arbitrary number of control points and each control point has an arbitrary weighting
- NURBS (nonuniform rational b-splines) a further extension knots in the knot vector do not have to be evenly spaced
- Bazier patches - a 2d extension, made up of 16 control points, given by

$$P(s,t) = \sum_{i=0} \sum_{j=0} b_i(s) b_j(t) P_{i,j}$$

  where $B_i(t) = \binom{4}{i} t^i (1-t)^{4-i}$
- The tensor product

$$\begin{pmatrix} A \\ B \\ C \end{pmatrix} \otimes \begin{pmatrix} D \\ E \\ F \end{pmatrix} = \begin{pmatrix} AD & AE & AF \\ BD & BE & BF \\ CD & CE & CF \end{pmatrix}$$

- Let the $A^n$ and $B^m$ be Bezier curves with order $n$ and $m$ respectively, these do not have control points, i.e. $A^n_i = B^n_i(t)$
  - The tensor product of $A$ and $B$ will be a bezier patch, when multiplied with a matrix of control points a bezier patch for those control points will be produced

# 4 Subdivision of surfaces

- Instead of building a parameterised curve repeatedly subdivide a set of course control points
- Will produce a smooth limit surface
- An interpolating scheme is one which retains and passes through the original control points
- An approximating scheme is one which builds a surface close to the control points

## 4.1 Chaikin

- One each edge insert new points at $\frac{1}{4}$ and $\frac{3}{4}$, deleting the old points
- This can be written as

$$P_{2i}^{k+1} = \left(\frac{3}{4}\right) P_i^k + \left(\frac{1}{4}\right) P_i^k$$
$$P_{2i+1}^{k+1} = \left(\frac{1}{4}\right) P_i^k + \left(\frac{3}{4}\right) P_i^k$$

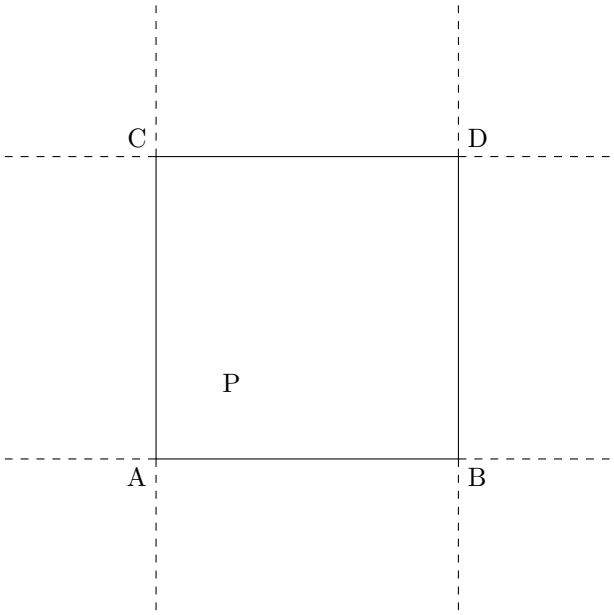
- This can be written as a vector equation

$$\begin{bmatrix} \vdots \\ P_{k+1}^{2i-2} \\ P_{k+1}^{2i-1} \\ P_{k+1}^{2i} \\ P_{k+1}^{2i+1} \\ P_{k+1}^{2i+2} \\ P_{k+1}^{2i+3} \\ \vdots \end{bmatrix} = \frac{1}{4} \begin{bmatrix} \ddots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \iddots \\ \cdots & 0 & 3 & 1 & 0 & 0 & 0 & \cdots \\ \cdots & 0 & 1 & 3 & 0 & 0 & 0 & \cdots \\ \cdots & 0 & 0 & 3 & 1 & 0 & 0 & \cdots \\ \cdots & 0 & 0 & 1 & 3 & 0 & 0 & \cdots \\ \cdots & 0 & 0 & 0 & 3 & 1 & 0 & \cdots \\ \cdots & 0 & 0 & 0 & 1 & 3 & 0 & \cdots \\ \iddots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots \end{bmatrix} \begin{bmatrix} \vdots \\ P_k^{2i-2} \\ P_k^{2i-1} \\ P_k^{2i} \\ P_k^{2i+1} \\ P_k^{2i+2} \\ P_k^{2i+3} \\ \vdots \end{bmatrix}$$

- This matrix is then encoded as a kernel $h = \frac{1}{4}[\cdots, 0, 0, 1, 3, 3, 1, 0, 0, \cdots]$

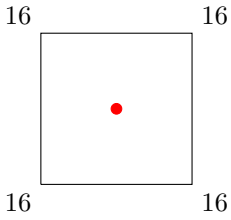
## 4.2 Doo-Sabin

- Takes Chaikin to 3D
- Replaces each point with four new points
- Has the following formula $P = \frac{9}{16}A + \frac{3}{16}B + \frac{3}{16}C + \frac{1}{16}D$, for a face

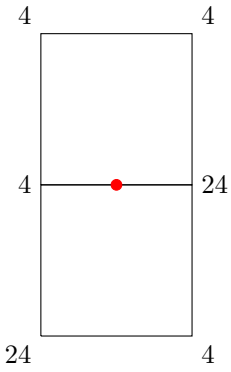


## 4.3 Catmull-Clark

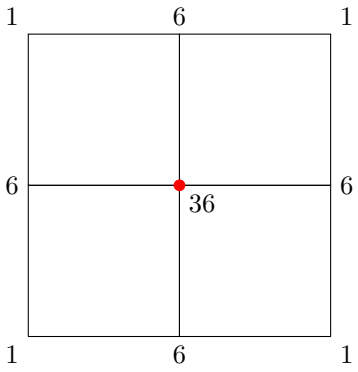
- Has kernel $h = \left(\frac{1}{8}\right)[1, 4, 6, 4, 1]$
- Consists of three rules
  1. Face rule




  2. Edge rule




  3. Vertex rule



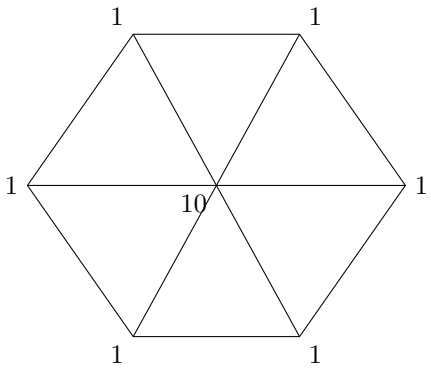

- This can be encoded as a matrix

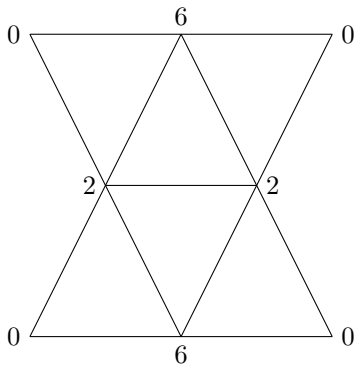
$$\frac{1}{8}\begin{bmatrix}1\\4\\6\\4\\1\end{bmatrix} \otimes \frac{1}{8}\begin{bmatrix}1\\4\\6\\4\\1\end{bmatrix} = \frac{1}{64}\begin{bmatrix}1 & 4 & 6 & 4 & 1\\4 & 16 & 24 & 16 & 4\\6 & 24 & 36 & 24 & 6\\4 & 16 & 24 & 16 & 4\\1 & 4 & 6 & 4 & 1\end{bmatrix}$$

- Both Catmull-Clark and Doo-Sabin work on quadrilateral meshes
- An extraordinary vertex is one which has a valance other than four
- An extraordinary face is one which does not have four edges
- Catmull-clark can be extended to cope with extraordinary vertices

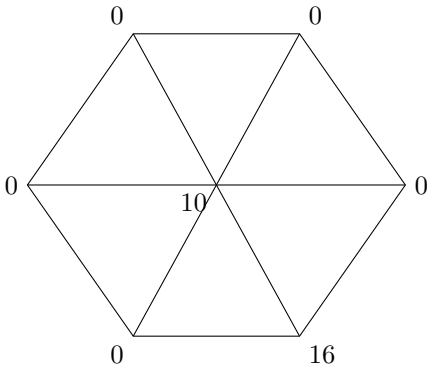
## 4.4 Triangular meshes

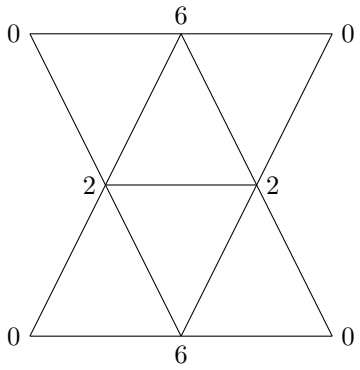
- Loop scheme




- Butterfly Scheme




## 4.5 General points

- Continuous level of detail
  - For live applications can dynamically increase the resolution
  - Subdivide a face when it is the same size as a pixel
- One can directly calculate the limit surface of Catmull-Clark
- Bounding boxes
  - The limit surface is a weighted average of the control points
  - If all weights are nonnegative and sum to one then the limit surface lies within the convex hull of the control points
  - Under these conditions the limit surface's bounding box will lie entirely within that of the control points
- Can not just split a surface as will remove some of the control points and produce a different limit surface
- For live applications can have continuous levels of detail as a function of distance

# 5 Shader techniques

## 5.1 Illumination

- The classic lighting equation

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

- Have two types of shading
  - Gouraud - Colour at the vertex
  - Phong - Colour at the polygon fragment
- Gooch Shading
  - Map illumination to hot and cold colours
  - Highlight edges in black

## 5.2 Textures

- Map coordinates on the object's surface to coordinates in the texture image
- Can procedural generate textures based on the coordinate of the pixel
- Can use a texture to map the normals of a surface, making the shape appear to have a 3D texture

## 5.3 Antialiasing

### 5.3.1 GPU Rendering

- Multiple-sample anti-aliasing
  - Render at a higher resolution and then down sample the image to blur out any aliasing
  - This has high memory requirements which can be limitting in high resolution environments
- An alternative method
  - Draw the scene normally
  - Draw wide lines around the object's silhouettes, blur accross this line
  - Compose the shapes with their silhouette lines
  - Great for polygons less so for more complex scenes (eg. video games)
- Fast approximate anti-aliasing
  - Use pixel-to-pixel contrast to determine edges
  - Use the pixel pairs to find vertical and horizontal edges
  - For each edge find the highest contrast pixel pair 90°
  - Use this to identify edges
  - Resample along these edges

#### 5.3.2 Textures

- To stop antialiasing of textures sample them dynamically
- The density of samples is a function of how the texture changes pixel to pixel
- Can represent textures as signed distance feilds
  - For monochromatic images each pixel is stored as the distance to the closest pixel of the opposite colour
  - Black pixels are stored as negatives
  - Can render smoothly by rendering an isocline
  - Borders can be added by rendering multiple isoclines

### 5.4 Tessellation shader

- Generate new vertices within patches
- Can have levels-of-detail, make render more coarse as you move from the camera
- Can use it deform geometry to apply textures
- Tell OpenGL how many vertices to generate for each patch
- New vertices generated in the *tessellation primitive generator* and are then passed into the *tessellation evaluation shader* which will calculate all per vertex data

## 6 Global illumination

### 6.1 Rendering equation

- Anisotropic shading is when light reflects off a surface differently depending on the direction of the incident light
- Bidirectional reflectance distribution function

$$\rho(\omega_i, \omega_r) = \frac{d(L_r(\omega_r))}{d(H_i(\omega_i))}$$

  - For given incident and reflectance vectors provides a ratio between the irradiance (input light) and the radiance (output light)
  - This is something which is measured experimentally, but is very costly to measure and store
- The rendering equations

$$L_r(\omega_r) = \int_\Omega \rho(\omega_i, \omega_r) L_i(\omega_i) cos\ \theta_i d\omega_i$$

  - where $\Omega$ is a unit hemisphere, $L_i$ is the magnitude of the incident light, $L_r$ is the magnitude of the reflected light
  - Loosely this is "the sum of contributions to the relfectance along $\omega_r$ from incident lights with all possible directions within the unit hemisphere $\Omega$"

## 6.2 Ambient occlusion

- Ambient occlusion is a method for removing ambient light from areas in which it would be blocked
- Treat the background (sky) as a light source
  - For each vertex calculate the amount of background light reaches the vertex
  - Very expensive
- Can render the scene from the perspective of each vertex
  - Count the amount of background visible
  - Also very expensive
- Can use the Monte-Carlo method to approximate the amount of background
- For stationary objects can precompute an occlusion map and apply this as a texture
- Screen space ambient occlusion
  - For each pixel sample it's neighbours' z-values
  - If it's neighbour is closer to the camera then there is a degree of occlusion
  - If it's neighbour is significantly closer to the camera then these may be different objects and no occlusion will occur
  - Sum the occlusions
- Ambient occlusion can be achieved trivially with signed distance fields

## 6.3 Rediosity

- Simulates the global reflection and dispersion of light
- Breaks the scene into mainly finite patches and then calculates the energy transfer between them
- The algorithm
  - Divide surfaces into patches
  - For each pair of patches $A$, $B$ compute the *view factor* (or *form factor*) the amount energy which is transferred from $A$ to $B$, this is a function of their separation and relative orientation
  - Calculate the lighting of all directly light patches
  - Bounce light from light patches to all they light, weighting the lighting by the view factor
  - Repeat
- The radiosity of a patch is the amount of light leaving the patch per discrete time interval, this is the sum of the light it emits and it reflects
- Can generate patches procedural or dynamically
  - Can subdivide surface into small patches of a similar size
  - Can subdivide when first derivative of the intensity rises above a threshold (too much change in that patch)
- The cost of calculating radiosity is proportional to the square of the number of patches
- An equation for view factor is

$$F(i \to j) = \frac{cos\ \theta_i cos\ \theta_j}{\pi r^2} V(i,j)$$

  where

  - $\theta_i$ is the angle from the normal of $i$ and the line to patch $j$
  - $\theta_j$ is the angle from the normal of $j$ and the line to patch $i$
  - $r$ is the distance between patches $i$ and $j$
  - $V(i, j)$ is the visibility from $i$ to $j$ with 0 being totally occluded and 1 being clear
- The value $V(i, j)$ can be calculated by rendering the scene from the point of view on each patch onto the walls of a hemicube (or more accurately but harder a hemisphere) and then for each other patch determining the percentage visible