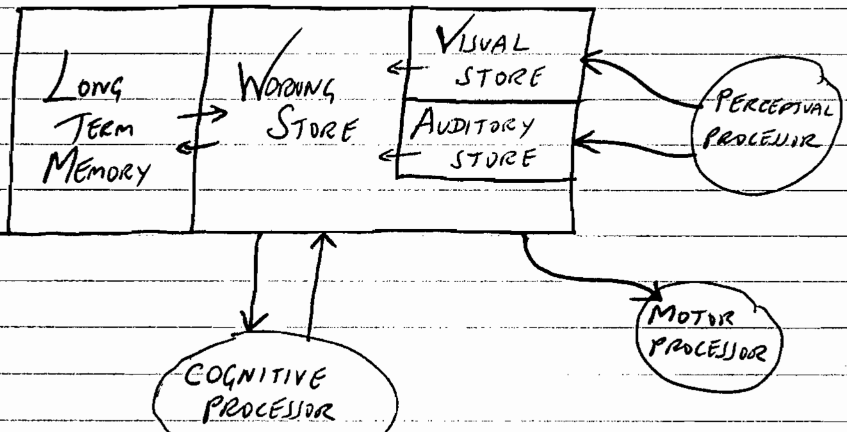
1999

Adv-Graph
1999/p9/1 24
MOD ANSWER

(a) The Model Human Processor (MHP) treats the human as some NAD sort of idealised processing device. There are ~~thought to~~ three processors in the model: perceptual, cognitive & motor. Each has an experimentally measurable cycle time which can be thought of as the time required by that processor to undertake one basic operation. The times for different people vary quite widely.

The three processors are attached to memory, roughly as shown below.



Each of the stores is thought to have a limited capacity & a half-life for decay. Long-term memory has a large capacity & a very long storage half-life; it seems to be different type of thing to the short term stores.

When considering an action we expect it to require a number of cycles of each processor and can estimate the total time as:

$$t = n_p \cdot t_p + n_c \cdot t_c + n_m \cdot t_m$$

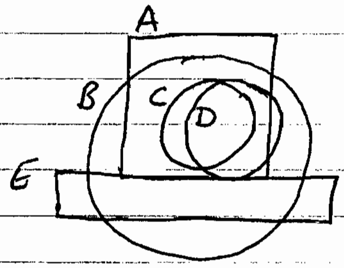
(b) The MHP is only a partial model of human processing. It works well in explaining the handling of small/simple tasks but is not as useful in modelling large complex tasks.

Other limitations would be valid here, provided they are reasonable.

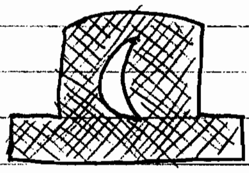
(c) CSG has three operations : UNION (∪)
INTERSECTION (∩)
DIFFERENCE (−)

A CSG object can be represented as a binary tree by insisting that each operation has ~~two~~ exactly two operands. This does not restrict the range of possible CSG objects in any way.

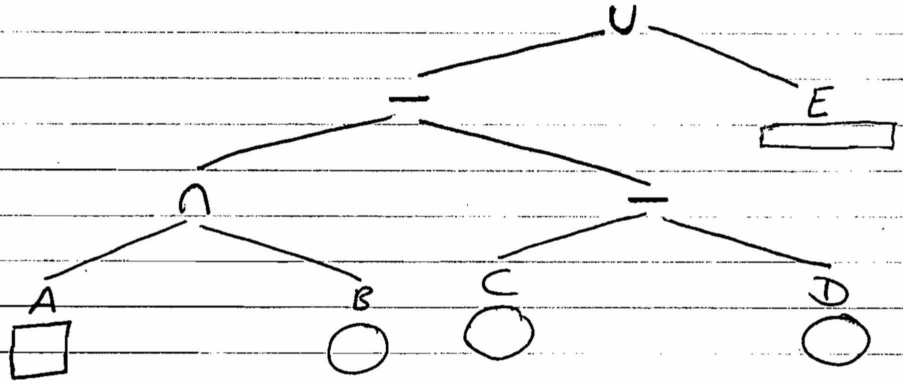
e.g.



$((A \cap B) - (C - D)) \cup E$



can be represented as



Given a ray the intersection points of that ray with every primitive (leaf) are calculated. These lists of points are then passed up the tree to find the intersection points of the entire CSG object with the ray, and hence the closest intersection point.

The list of intersection points consists of a (potentially empty) set of pairs of ~~poits represent~~ values. Each pair represents the intersection points as the ray enters and then leaves the object. The values are values of the parameter t, representing distance along the ray $\underset{\sim}{R}(t) = \underset{\sim}{O} + t\underset{\sim}{P}$.

Given two lists of pairs (from its two children) a node combines them into a single list of pairs in the following ways:

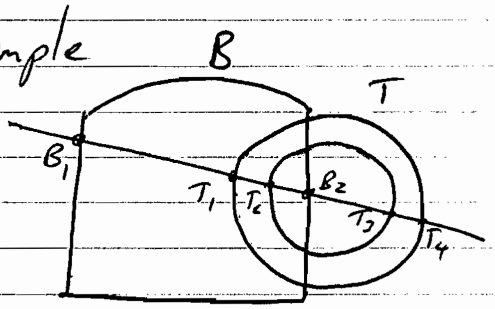
UNION: while ~~the~~ ray is in at least one of the two objects it is in the UNION

INTERSECTION: while the ~~ob~~ ray is in both of the objects it is in the INTERSECTION

DIFFERENCE: while the ray is in one (the left child's) object and not the other (the right child's) object it is in the difference.

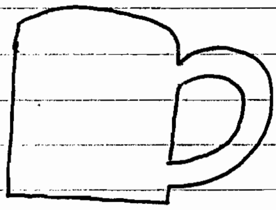
These processes can be achieved by parsing the two lists in order of t value and keeping track of inside/outside for the two objects

An example

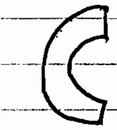


LISTS $((B_1, B_2))$
$((T_1, T_2), (T_3, T_4))$

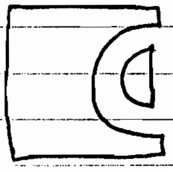
$B \cup T \longrightarrow ((B_1, B_2), (T_3, T_4))$



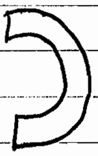
$B \cap T \longrightarrow ((T_1, T_2))$



$B - T \longrightarrow ((B_1, T_1), (T_2, B_2))$



$T - B \longrightarrow ((T_3, T_4))$



~~may be used. At the end of scan-conversion all of the sub-pixel grids are combined, in depth order, to generate the appropriate colour for the pixel.~~

~~This results in an image in which the effects of aliasing are greatly ameliorated. The extra overhead involved in using an A-buffer is not high because the sub-pixel mask manipulation can be done mainly by simple boolean algebra.~~ 4/5

~~Key points:~~

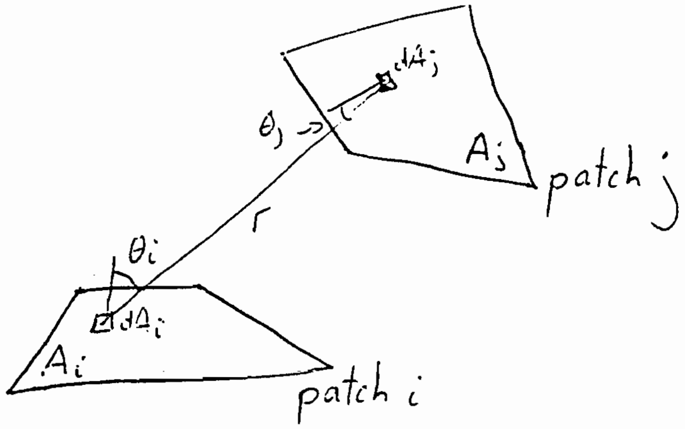
~~(1) Z-buffer algorithm produces images with artefacts~~
~~(2) A-buffer algorithm greatly reduces (but does not remove) these artefacts~~
~~(3) some explanation of how the A-buffer works:~~
- ~~sub-pixel sampling grid at polygon edges~~
- ~~grids combined at end to get colour~~
- ~~use of boolean algebra makes A-buffer fairly cheap extension to z-buffer~~

In radiosity, a form factor is the proportion of one patch's ~~energy that is~~ transmitted energy that arrives at another patch. Thus a form factor must be calculated for every pair of patches in the scene.

Form factors can, theoretically, be calculated by performing an ~~(quadruple)~~ integration across both source and destination patches' surfaces

[redacted]

Adv Graph 1999/p9/5
q4
NAD



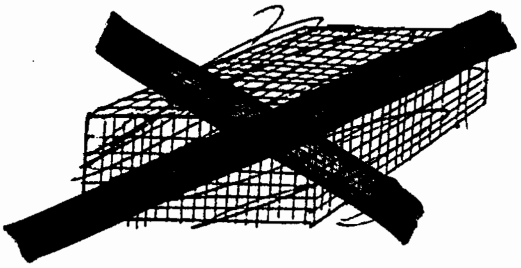
$$F_{i\to j} = \int_{A_i}\int_{A_j} \frac{\cos\theta_i \cos\theta_j}{\pi r^2}\, dA_j\, dA_i$$

This is made even more difficult by the fact that the above equation assumes that there is no occluding object between any dA_i and any dA_j.

~~An approximation to the form factor can be found by raytracing a number of rays from random positions on one patch to random~~

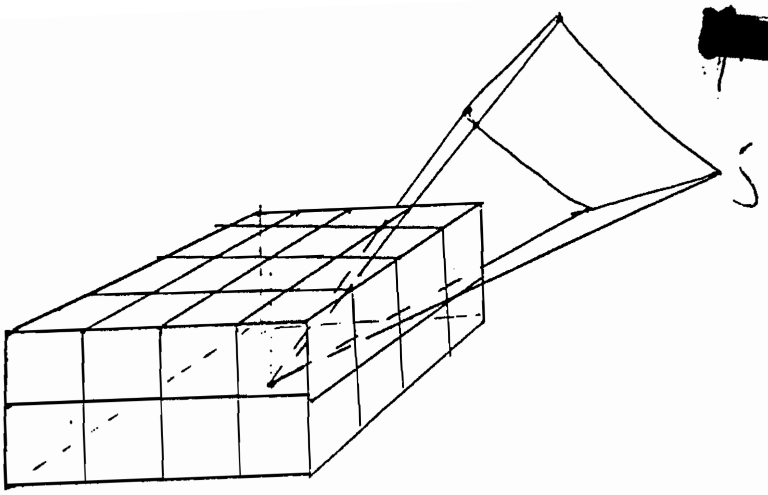
In practice the integral cannot be solved.

An approximation can be found by surrounding the centre of patch i with a hemicube. Each face of the hemicube can be considered an image & ray-tracing or z-buffer techniques used to calculate which objects are visible at each pixel. This (with the pre-calculation of the form-factor contribution, each pixel) allows the calculation of $F_{i\to j}$ all j.



[redacted]

Adv. Graph
1999/p9/6
q4
NAD



Key points:

(1) Form factors are the proportion of a patch's transmitted energy which is received by another patch

(2) A reasonable method of calculating formfactors must be outlined, it must be able to cope with occlusion.

7