Advanced Graphics 2004 p9q6

(a) a plane can be defined by a point on the plane, Q, and a normal vector, N

a ray can be defined by an eye point, E, and a direction vector, D

The plane equation is: $(P-Q)\cdot N = 0$

The ray equation is: $P(t) = E + tD$, $t \in \mathbb{R}$, $t \geq 0$

The value of t at the intersection point can be found by substituting one equation in the other:

$$(E + tD - Q)\cdot N = 0$$

$$\Rightarrow \qquad tD\cdot N = (Q-E)\cdot N$$

$$\Rightarrow \qquad t = \frac{(Q-E)\cdot N}{D\cdot N}$$

If the numerator is zero, the eye lies in the plane.

If the denominator is zero, the ray is parallel to the plane and there is no intersection.

If $t < 0$ then the intersection point with the <u>line</u> is behind the eye and there is no intersection with the <u>ray</u>

One way to implement this is:

```
float rayplane ( E: point, D: vector, Q: point, N: vector)
{
   float denom = D • N;
   if (denom == Ø) raise ("No intersection")
   else {
      float t = (Q - E) • N / denom;
      if ( t < Ø ) raise ("No intersection")
      else return (t);
   }
}
```

raise(), raises an exception as a way of flagging that there is no intersection point

Once you have t, the intersection point itself can be found by substituting the value of t into the ray equation.

(b) You must define one side of the plane to be INSIDE while the other side is OUTSIDE. You can do this, for example, by saying that the normal vector points toward OUTSIDE. An inside/outside test is thus:

if $((P-Q)\cdot N) > \emptyset$ then OUTSIDE
else INSIDE

The combination of the OUTSIDE/INSIDE flag for the ray's eyepoint, along with a list of the zero or one intersection points between ray and plane, is what is required by the CSG algorithm.

(c) Three operators: UNION ($\cup$), INTERSECTION ($\cap$), DIFFERENCE ($\setminus$)
Binary tree: primitive objects at the leaves, operators at the internal nodes [primitive objects are those for which we have a ray-object intersection algorithm]

Assume: each ray-primitive intersection algorithm returns a boolean, INOUT, which is TRUE if the eye point is inside the primitive, FALSE if not; and a list of intersection points between the ray and the primitive sorted in ascending order.

Given this, we can take any two objects and combine them to produce a single object with its own INOUT and list of intersection points.

The algorithm takes as input:

$(\text{INOUT}_A, (t_{A1}, t_{A2}, t_{A3}, \ldots))$

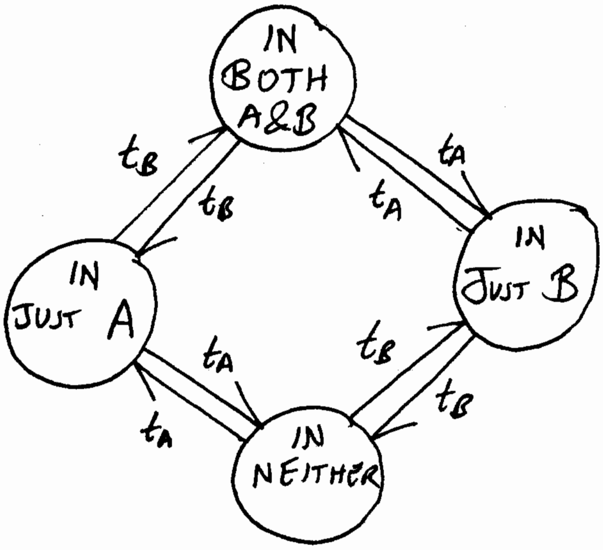
$(\text{INOUT}_B, (t_{B1}, t_{B2}, t_{B3}, \ldots))$

and produces as output:

$(\text{INOUT}_C, (t_{C1}, t_{C2}, t_{C2}, \ldots))$

where $C = A \cup B$ or $A \cap B$ or $A \setminus B$ or $B \setminus A$ depending on the operator.

We use a simple state machine:



Algorithm:

start state is determined by INOUT_A and INOUT_B in the obvious way (e.g. $\text{INOUT}_A = \text{INOUT}_B = \text{TRUE} \Rightarrow$ start in 'IN BOTH A&B')

list of t_{Ci} is generated by:

(*) compare heads of t_A and t_B lists, pick the smallest and remove it from its list

change state, as shown in the diagram, depending on whether the value was taken from t_A or t_B

if the state change took you into or out of state X (see below) then add the t value to the tail of the t_C list, else discard it

repeat from (*) until both t_A and t_B lists are exhausted

State X and the value of $INOUT_C$ are the only things which depend on the operator

operator	State X	$INOUT_C$
union	IN NEITHER	$INOUT_A$ <u>OR</u> $INOUT_B$
intersection	IN BOTH A & B	$INOUT_A$ <u>AND</u> $INOUT_B$
A-B	IN JUST A	$INOUT_A$ <u>AND</u> <u>NOT</u> $INOUT_B$
B-A	IN JUST B	<u>NOT</u> $INOUT_A$ <u>AND</u> $INOUT_B$

To find the <u>first</u> intersection point with the final object, take the first t-value from the list of intersection points and substitute it into the ray equation.

(d) $N_{4,1}(t) = \begin{cases} 1, & 4 \le t < 5 \\ 0, & \text{otherwise} \end{cases}$

$N_{5,1}(t) = 0$

$N_{6,1}(t) = 0$

$N_{7,1}(t) = \begin{cases} 1, & 5 \le t < 6 \\ 0, & \text{otherwise} \end{cases}$

$\cancel{N_{8,1}(t) = \begin{cases} 1, & 6 \le t < 7 \\ 0, & \text{otherwise} \end{cases}}$

$$N_{i,k}(t) = \frac{t - t_i}{t_{i+k-1} - t_i} N_{i,k-1}(t) + \frac{t_{i+k} - t}{t_{i+k} - t_{i+1}} N_{i+1,k-1}(t)$$

$N_{4,2}(t) = \begin{cases} (t-4), & 4 \le t < 5 \\ 0, & \text{o/w} \end{cases}$

$N_{5,2}(t) = 0$

$N_{6,2}(t) = \begin{cases} (6-t), & 5 \le t < 6 \\ 0, & \text{o/w} \end{cases}$

$$N_{4,3}(t) = \begin{cases} (t-4)^2, & 4 \le t < 5 \\ 0, & o/w \end{cases}$$

$$N_{5,3}(t) = \begin{cases} (6-t)^2, & 5 \le t < 6 \\ 0, & o/w \end{cases}$$

$$N_{4,4}(t) = \begin{cases} (t-4)^3, & 4 \le t < 5 \\ (6-t)^3, & 5 \le t < 6 \\ 0, & o/w \end{cases}$$

2004 p9q6 MARKING SCHEME

(a) Correct plane equation 1
Subtitute ray equation into plane equation 1
Correct equation for t 1
Check for $D \cdot N = 0$ 1
Check for $t < 0$ 1
5

(b) The key point is that you need to define one side of the plane to be IN and one OUT because there is no natural definition of IN OUT
One mark for this 1
One mark for overall sense & clarity 1
2

(c) The three operators 1
The binary tree structure 1
Six marks for the algorithm
Correct information from ray/primitive algorithms 1
State diagram 1
How to get $INOUT_c$ 1
How to get t_c list 2
How to find first intersection point 1
8

(d) Correct method:
- for Base cases $(N_{i,1})$ 1
- for Recursive cases $(N_{i,k},\ k \geq 2)$ 2

Correct answer. 2
5

20