APPENDIX B: C CODE FOR INTERSECTION AND SURFACE NORMAL

```
#define VEC_SUB(a,b,c)  {(c)->x = (a).x-(b).x; \
                        (c)->y = (a).y-(b).y; \
                        (c)->z = (a).z-(b).z;}
#define CROSS(a,b,c)    {(c)->x = (a.y * b.z) - (a.z * b.y); \
                        (c)->y = (a.z * b.x) - (a.x * b.z); \
                        (c)->z = (a.x * b.y) - (a.y * b.x);}
      /* assigns ray-surface intersection and surface normal */
void
Intersect_Surface( double z_near, double z_far, double epsilon, double distance,
         Vector position, Vector prev_position, Ray *ray, HitData *hit )
{
   Vector P_r,          /* point to the right, relative to ray dir and “up” */
          v_d,          /* vector from prev_position to position */
          v_l,          /* vector from prev_position to p_r */
          n;            /* surface normal */

      /* first construct three points that lie on the surface */
   prev_position.z = z_near;
   position.z = z_far;
      /* construct a point one error width to right */
   p_r.x = prev_position.x + epsilon*camera.right_dir.x;
   p_r.y = prev_position.y + epsilon*camera.right_dir.y;
   p_r.z = prev_position.z + epsilon*camera.right_dir.z;
   p_r.z = Displacement( p_r, distance );

      /* get two vectors in the surface plane; cross for surface normal */
   VEC_SUB( position, prev_position, &v_d );
   Normalize( &v_d );
   VEC_SUB( p_r, prev_position, &v_l );
   Normalize( &v_l );
   CROSS( v_l, v_d, &n );
   Normalize( &n );

      /* assign the various hit data */
   hit->distance = distance;
   hit->intersect = prev_position;
   hit->normal = n;

} /* Intersect_Surface() */
```