APPENDIX A: C CODE IMPLEMENTING QAEB TRACING

```
    /* determines 3D position along ray at distance t */
#define RAY_POS(ray,t,pos) \
            {(pos)->x = (ray)->origin.x + t*(ray)->dir.x; \
             (pos)->y = (ray)->origin.y + t*(ray)->dir.y; \
             (pos)->z = (ray)->origin.z + t*(ray)->dir.z; }

typedef struct { /* catch-all type for 3D vectors and positions */
    double x;
    double y;
    double z; } Vector;
    /* returns TRUE if HF intersected, FALSE otherwise */
Boolean
Intersect_Terrain(int row, int column, double epsilon, Ray *ray, Hit *hit )
{
double d, /* ray parameter, equal to distance travelled */
       alt,                              /* alt at current step */
       prev_d,                           /* d at last step */
       prev_alt;                         /* alt at last step */
Vector position,                         /* current position along ray */
       prev_position;                    /* previous position along ray */

if ( row == 0 ) { /* if at bottom of bottom-to-top rendering */
    d = near_clip_dist;                  /* init march stride */
    RAY_POS( ray, d, &prev_position );   /* init previous 3D position */
    prev_alt = Displacement( prev_position, d ); /* evaluate the HF function */
} else { /* (this scheme is valid only for vertical columns) */
    d = prev_d = prev_dist[column]; /* start at final d of prev. ray in column */
    prev_position = prev_pos[column];
    prev_alt = prev_alts[column];
}

while ( d < far_clip_dist ) {            /* the QAEB raymarch loop */
    d += d * epsilon;                    /* update the marching stride */
    RAY_POS( ray, d, Sposition );        /* get current 3D position */
    alt = Displacement( position, d );   /* evaluate the HF function */
    if ( position.z < alt ) {            /* surface penetrated */
```

```
            Intersect_Surface( prev_alt, alt, d*epsilon, d,
            position, prev_position, ray, hit );
            prev_dist[column] = prev_d; /* update prev. distance data */
            prev_alts[column] = prev_alt;
            prev_pos[column] = prev_position;
            return( TRUE );
        }
        prev_d = d;
        prev_alt = alt;
        prev_position = position;
    }

    /* exceeded far clip distance; update "prev_dist" appropriately & exit. */
    prev_dist[column] = d;
    return( FALSE );

    } /* Intersect_Terrain() */
```