◊ V.5

A Minimal Ray Tracer

Paul S. Heckbert
Computer Science Department
Carnegie Mellon University
5000 Forbes Ave.
Pittsburgh, PA 15213-3891
ph@cs.cmu.edu

In the Spring of 1987 I ran a "minimal ray tracer programming contest" over the Internet on the USENET group *comp.graphics*, challenging people to write the shortest Whitted-style ray tracing program in C. This Gem summarizes the contest and lists an ultra-minimal hybrid of the winning entries. In 21 lines of C code, this program ray traces a scene consisting of spheres, recursing for specular reflected and transmitted rays, and simulating shadows and colored lights.

First, the contest rules are briefly summarized, then the announcement of winners is given, and finally, C code is listed. The text here is a condensed version of the e-mail messages from *comp.graphics*. For full listings, see the Gems IV distribution on floppy or by FTP.

◊ Contest Rules ◊

The goal: write the shortest Whitted-style ray tracing program in C (Whitted 1980). Countless people have written basic sphere ray tracers and made pictures of glass and chrome balls, so isn't it time we found out just how short such a program can be?

An entry in the contest was considered valid if it met the following conditions:

- it consisted of a single C source file, which compiled into an executable with no warnings,
- it ray traced spheres using Whitted's recursive shading model, simulating shadows and colored lights (more precisely, the pixel values output had to agree within ± 10 with a solution computed using a specified shading formula),
- the scene description was compiled into the program from a header file,
- it used a specified perspective camera model,
- a color picture was output in a specified ascii format.

Not required were: antialiasing, any geometric primitives besides spheres, CSG, Jell-O, any distribution ray tracing effects, lint-free code, or speed.

IBM ISBN 0-12-336155-9
Macintosh ISBN 0-12-336156-7

The winner was the valid entry with the minimum number of tokens after running the source through the C preprocessor. (This is a better measure of program length than number of lines or object code size, since it is machine-independent and more hacker-resistant.) Code to count tokens was provided to the entrants.

The contest was announced on 4 May 1987 and the winners were announced on 20 June 1987.

◇ Contest Winners ◇

I received entries from five people. The entries I received all passed my validation tests and they were all quite clever. Joe Cychosz of Purdue won first place with an incredibly short program of just 916 tokens.

PLACE	#TOKENS	AUTHOR	NOTES
		GENUINE ENTRIES	
1	916	Joe Cychosz, Purdue	compiler-dependent
1a	932	Joe Cychosz, Purdue	portable
2	956	Darwyn Peachey, Saskatchewan	portable
3	981	Michel Burgess, Montreal	portable
4	1003	Greg Ward, Berkeley	portable
		HONORABLE MENTIONS	
c1	10	Tony Apodaca, Pixar	cheater
c2	66	Greg Ward, Berkeley	cheater

Interestingly, the entries had nearly identical modularization into six subroutines: main, trace, intersect-sphere, vector-normalize, vector-add, and dot-product. One person, Greg Ward, cleverly exploited a loophole in my rules which allowed arbitrarily long character strings to count as a single token, and submitted a program which writes a source file, compiles it, and runs it! Tony Apodaca used a different cheat: hiding the real program in an unused part of the source file and compiling and running that program with a single call to `system()`. His program is about as minimal as you can get: 10 tokens.

What did we learn from all this? The first thing I learned is that there aren't many real hackers in *comp.graphics*. Various people offered explanations for the sparse turnout:

> "The unwashed might think that the task is too difficult and the cognoscenti might think 'I know I could do it, but it's too much trouble just for a contest'. "
>
> "We're all busy with previous hacking commitments."
>
> "If you had a 'Get a job at Pixar Ray Tracer Contest', I might be motivated to enter."[1]

[1] Actually, Darwyn Peachey did get a job at Pixar shortly after this contest, but it's not clear if there was a connection.

Those who entered the contest also learned some repulsive C coding tricks:

- color = point: `struct {float r, g, b;}` ⇒ `struct {float x, y, z;}`
- pass structures (not just pointers to structs) to allow vector expressions
- use global variables for return values
- no optimizations: trace specular and transmitted rays even if coeff=0!
- reuse variables!
- `for (i=0; i<n; i++)` ⇒ `i = n; while (i--)`
- merge x and y loops into one
- assume statics are initialized to 0
- `&sph[i]` ⇒ `sph+i`
- choose just the right set of utility routines
- creative use of commas, e.g.: `if (a) {b;c;}` ⇒ `if (a) b,c;`
- move assignments into expressions: `b = vdot(D, U = vcomb(-1., P, s->cen))`

Cheats (non-portable C):

- eliminate semicolons in struct def: `struct {int a;}` ⇒ `struct {int a}`
- assume right-to-left argument evaluation order

Winner of the most shocking cheat award, a little gem by Joe Cychosz:

- `printf("%f %f %f\n", pt.x, pt.y, pt.z);` ⇒ `printf("%f %f %f\n", pt);`

Since Joe Cychosz said this was only his second C program (!), I asked him how he managed to do so well. He responded:

> "My primary program language is FORTRAN and COMPASS (CDC assembly) for the CYBER 205 and the CYBER 800s. ... This is an interesting way to learn C. I spent the weekend looking through Kernighan & Ritchie trying to determine if what I wanted to do was legal C. Finally, Kirk Smith found 12 tokens while we were sitting in a bar."

But if you were expecting a useful ray tracing program out of this, a warning: as one would expect of any "minimal" program, the winning program is cryptic, inefficient, unmaintainable, and nearly useless except as a source of C coding tricks. If you want a good ray tracer, look elsewhere!

◇ C Code ◇

After the contest, I took some of the tricks from Darwyn Peachey's and Joe Cychosz's programs and combined them with some of my own to create an ultra-minimal hybrid program. Two versions are reproduced below: a compacted version, `minray.card.c` that fits nicely on a business card or on an old-fashioned 24×80 terminal screen, and a "readable" version, `minray.c`.

minray.card.c

```
typedef struct{double x,y,z}vec;vec U,black,amb={.02,.02,.02};struct sphere{
vec cen,color;double rad,kd,ks,kt,kl,ir}*s,*best,sph[]={0.,6.,.5,1.,1.,1.,.9,
.05,.2,.85,0.,1.7,-1.,8.,-.5,1.,.5,.2,1.,.7,.3,0.,.05,1.2,1.,8.,-.5,.1,.8,.8,
1.,.3,.7,0.,0.,1.2,3.,-6.,15.,1.,.8,1.,7.,0.,0.,0.,.6,1.5,-3.,-3.,12.,.8,1.,
1.,5.,0.,0.,0.,.5,1.5,};yx;double u,b,tmin,sqrt(),tan();double vdot(A,B)vec A
,B;{return A.x*B.x+A.y*B.y+A.z*B.z;}vec vcomb(a,A,B)double a;vec A,B;{B.x+=a*
A.x;B.y+=a*A.y;B.z+=a*A.z;return B;}vec vunit(A)vec A;{return vcomb(1./sqrt(
vdot(A,A)),A,black);}struct sphere*intersect(P,D)vec P,D;{best=0;tmin=1e30;s=
sph+5;while(s-->sph)b=vdot(D,U=vcomb(-1.,P,s->cen)),u=b*b-vdot(U,U)+s->rad*s
->rad,u=u>0?sqrt(u):1e31,u=b-u>1e-7?b-u:b+u,tmin=u>=1e-7&&u<tmin?best=s,u:
tmin;return best;}vec trace(level,P,D)vec P,D;{double d,eta,e;vec N,color;
struct sphere*s,*l;if(!level--)return black;if(s=intersect(P,D));else return
amb;color=amb;eta=s->ir;d= -vdot(D,N=vunit(vcomb(-1.,P=vcomb(tmin,D,P),s->cen
)));if(d<0)N=vcomb(-1.,N,black),eta=1/eta,d= -d;l=sph+5;while(l-->sph)if((e=l
->kl*vdot(N,U=vunit(vcomb(-1.,P,l->cen))))>0&&intersect(P,U)==l)color=vcomb(e
,l->color,color);U=s->color;color.x*=U.x;color.y*=U.y;color.z*=U.z;e=1-eta*
eta*(1-d*d);return vcomb(s->kt,e>0?trace(level,P,vcomb(eta,D,vcomb(eta*d-sqrt
(e),N,black))):black,vcomb(s->ks,trace(level,P,vcomb(2*d,N,D)),vcomb(s->kd,
color,vcomb(s->kl,U,black))));}main(){printf("%d %d\n",32,32);while(yx<32*32)
U.x=yx%32-32/2,U.z=32/2-yx++/32,U.y=32/2/tan(25/114.5915590261),U=vcomb(255.,
trace(3,black,vunit(U)),black),printf("%.0f %.0f %.0f\n",U);}/*minray!*/
```

ray.h

```
/* ray.h for test1, first test scene */
#define DEPTH 3          /* max ray tree depth */
#define SIZE 32          /* resolution of picture in x and y */
#define AOV 25           /* total angle of view in degrees */
#define NSPHERE 5        /* number of spheres */

AMBIENT = {.02, .02, .02};      /* ambient light color */

/* sphere: x y z  r g b  rad  kd ks kt kl  ir */
SPHERE = {
     0., 6., .5,    1., 1., 1.,   .9,   .05, .2, .85, 0.,  1.7,
    -1., 8., -.5,   1., .5, .2,   1.,   .7, .3, 0., .05,   1.2,
     1., 8., -.5,   .1, .8, .8,   1.,   .3, .7, 0., 0.,    1.2,
     3., -6., 15.,  1., .8, 1.,   7.,   0., 0., 0., .6,    1.5,
    -3., -3., 12.,  .8, 1., 1.,   5.,   0., 0., 0., .5,    1.5,
};
```

minray.c

```
/* minimal ray tracer, hybrid version - 888 tokens
 * Paul Heckbert, Darwyn Peachey, and Joe Cychosz  13 Jun 87 */

#define TOL 1e-7
#define AMBIENT vec U, black, amb
#define SPHERE struct sphere {vec cen, color; double rad, kd, ks, kt, kl, ir} \
    *s, *best, sph[]
typedef struct {double x, y, z} vec;
#include "ray.h"
yx;
double u, b, tmin, sqrt(), tan();

double vdot(A, B)
vec A, B;
{ return A.x*B.x + A.y*B.y + A.z*B.z; }

vec vcomb(a, A, B)      /* aA+B */
double a;
vec A, B;
{
    B.x += a*A.x;
    B.y += a*A.y;
    B.z += a*A.z;
    return B;
}

vec vunit(A)
vec A;
{ return vcomb(1./sqrt(vdot(A, A)), A, black); }

struct sphere *intersect(P, D)
vec P, D;
{
    best = 0;
    tmin = 1e30;
    s = sph+NSPHERE;
    while (s-->sph)
        b = vdot(D, U = vcomb(-1., P, s->cen)),
        u = b*b-vdot(U, U)+s->rad*s->rad,
        u = u>0 ? sqrt(u) : 1e31,
        u = b-u>TOL ? b-u : b+u,
        tmin = u>=TOL && u<tmin ?
            best = s, u : tmin;
    return best;
}

vec trace(level, P, D)
vec P, D;
{
    double d, eta, e;
    vec N, color;
    struct sphere *s, *l;
```

```
    if (!level--) return black;
    if (s = intersect(P, D));
    else return amb;

    color = amb;
    eta = s->ir;
    d = -vdot(D, N = vunit(vcomb(-1., P = vcomb(tmin, D, P), s->cen)));
    if (d<0)
        N = vcomb(-1., N, black),
        eta = 1/eta,
        d = -d;
    l = sph+NSPHERE;
    while (l-->sph)
        if ((e = l->kl*vdot(N, U = vunit(vcomb(-1., P, l->cen)))) > 0 &&
            intersect(P, U)==l)
                color = vcomb(e, l->color, color);
    U = s->color;
    color.x *= U.x;
    color.y *= U.y;
    color.z *= U.z;
    e = 1-eta*eta*(1-d*d);
    /* the following is non-portable: we assume right to left arg evaluation.
     * (use U before call to trace, which modifies U) */
    return vcomb(s->kt,
        e>0 ? trace(level, P, vcomb(eta, D, vcomb(eta*d-sqrt(e), N, black)))
            : black,
        vcomb(s->ks, trace(level, P, vcomb(2*d, N, D)),
            vcomb(s->kd, color, vcomb(s->kl, U, black))));
}

main()
{
    printf("%d %d\n", SIZE, SIZE);
    while (yx<SIZE*SIZE)
        U.x = yx%SIZE-SIZE/2,
        U.z = SIZE/2-yx++/SIZE,
        U.y = SIZE/2/tan(AOV/114.5915590261),   /* 360/PI~=114 */
        U = vcomb(255., trace(DEPTH, black, vunit(U)), black),
        printf("%.0f %.0f %.0f\n", U);          /* yowsa! non-portable! */
}
```

◇ Acknowledgments ◇

I'd like to thank everyone who participated in the contest, especially Darwyn Peachey, who helped me debug the rules, and Paul Haeberli, who hatched the idea for the contest with me. A Postscript version of the minimal ray tracer was written by John Hartman of U.C. Berkeley. It runs on most printers.

◇ Bibliography ◇

(Whitted 1980) Turner Whitted. An improved illumination model for shaded display. *CACM*, 23(6):343–349, June 1980.