3D Graphics Systems

Luiz Velho
IMPA

Outlook

- System A
  - Generative Modeling
  - Z-Buffer + Rasterization
- System B
  - CSG Modeling
  - Ray Tracing
- System C
  - Primitive Modeling
  - Painter’s Algorithm

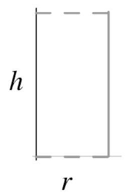
System A

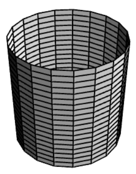
Generative Modeling

- Surface of Revolution

```
Vector3 g[MAXPTS];
main(int argc, char **argv)
{
  int nu, nv = NVPTS;
  Poly *tl;
  if (argc == 2)
    nv = atoi(argv[1]);
  nu = read_curve();
  tl = rotsurf(nu, g, nv);
  trilist_write(tl, stdout);
  exit(0);
}
```

cylinder





Making a Cylinder

- Command: `rotsurf 12 < ln.pts > cyl.scn`

```
1 1 0
1 -1 0
```

ln.pts

```
trilist {
{{0.186494, -1, 0.0722484}, {0.2, -1, 0}, {1, -1, 0}},
{{0.932472, -1, 0.361242},  {0.186494, -1, 0.0722484},  {1, -1, 0}},
{{0.932472, -1, 0.361242},  {1, -1, 0},  {1, 1, 0}},
{{0.932472, 1, 0.361242},  {0.932472, -1, 0.361242},  {1, 1, 0}},
{{0.932472, 1, 0.361242},  {1, 1, 0},  {0.2, 1, 0}},
{{0.186494, 1, 0.0722484},  {0.932472, 1, 0.361242},  {0.2, 1, 0}},
{{0.147802, -1, 0.134739},  {0.186494, -1, 0.07224},  {0.93247, -1, 0.361}},
{{0.739009, -1, 0.673696},  {0.147802, -1, 0.13473},  {0.93247, -1, 0.361}},
....
{{0.2, 1, 0},  {1, 1, 0},  {0.186494, 1, -0.0722483}}}
}
```

cyl.scn

Rasterization Rendering

- Object Centered Pipeline

```
int main(int argc, char **argv)
{
  Object *o;  Poly *p;  Color c;
  init_sdl();
  s = scene_read();
  init_render();
  for (o = s->objs; o != NULL; o = o->next) {
    for (p = o->u.pols; p != NULL; p = p->next) {
      if (is_backfacing(p, v3_sub(poly_centr(p), s->view->center)))
        continue;
      c = flat_shade(p, s->view->center, rc, o->mat);
      if (poly_clip(VIEW_ZMIN(s->view), poly_transform(p, mclip), 0))
        scan_poly(poly_homoxform(p, mdpy), pix_paint, &c);
    }
  }
  img_write(s->img, "stdout", 0);
  exit(0);
}
```

Z-Buffer Paint

- Pixel Depth

```
void pix_paint(Vector3 v,int n,int lv,Real lt,int rv,Real rt,Real st,void *c)
{
  if (zbuf_store(v))
    img_putc(s->img, v.x, v.y, col_dpymap(*((Color *)(c))));
}
```

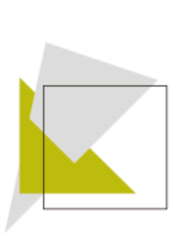
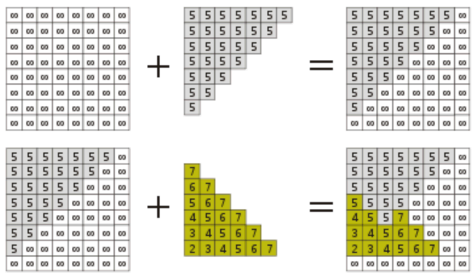


image buffer



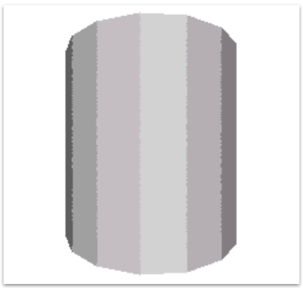
z-buffer

Rendering the Cylinder

- Command: `zbuff < cyl.scn > cyl.ras`

```
scene {
      camera = view { from = {0, 0, -2.5}, up = {0, 1, 0}},
      light = dist_light {direction = {0, 0, -1} },
      object = polyobj { shape = trilist {
                        {{0.186494, -1, 0.0722484}, {0.2, -1, 0}
                   ...
                        {{0.2, 1, 0},  {1, 1, 0},  {0.186494, 1,
       }
};
```

cyl.scn



cyl.ras

System B

CSG Modeling

- Constructive Solid Geometry Expression

```
main(int argc, char **argv)
{
  CsgNode *t;
  if((t = csg_parse()) == NULL)
    exit(-1);
  else
    csg_write(t, stdout);
  exit(0);
}
```

```
csg_obj:  '(' csg_obj bop csg_obj ')'
        | prim_obj
        ;
bop:      '|'
        | '&'
        | '\\'
        ;
prim_obj: 's' '{' NUM NUM NUM NUM '}'
```

CSG Grammar

Making a Carved Ball

- Command: `csg < s.csg > s.scn`

```
(s{ 0 0 0 1} \ s{ 1 1 -1 1}
```

s.csg

```
csgobj = csg_diff {
   csg_prim{ sphere { center = {0, 0,  0}}},
   csg_prim{ sphere { center = {1, 1, -1}}} }
```

s.scn

Ray Tracing

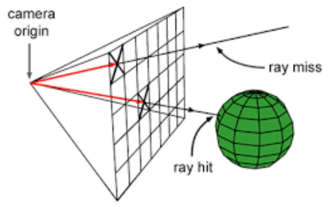
- Image Centered Pipeline

```
main(int argc, char **argv)
{
  Color c; int u, v;
  Ray r; Inode *l;
  init_sdl();
  s = scene_read();
  init_render();
  for (v = s->view->sc.ll.y; v < s->view->sc.ur.y; v += 1) {
    for (u = s->view->sc.ll.x; u < s->view->sc.ur.x; u += 1) {
      r = ray_unit(ray_transform(ray_view(u, v), mclip));
      if ((l = ray_intersect(s->objs, r)) != NULL)
        c = point_shade(ray_point(r, l->t), l->n, s->view->center, rc,
      else
        c = bgcolor;
      inode_free(l);
      img_putc(s->img, u, v, col_dpymap(c));
    }
  }
  img_write(s->img,"stdout",0);
  exit(0);
}
```

Creating a Ray

- Pixel Sampling Grid

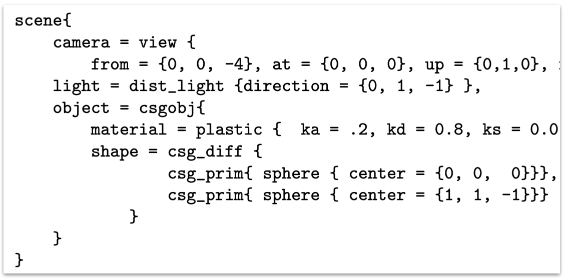
```
Ray ray_view(int u, int v)
{
  Vector4 w = v4_m4mult(v4_make(u, v, s->view->sc.ur.z, 1), mdpy);
  return ray_make(v3_v4conv(v4_m4mult(v4_make(0, 0, 1, 0), mdpy)),
                  v3_make(w.x, w.y, w.z));
}
```



rays from camera

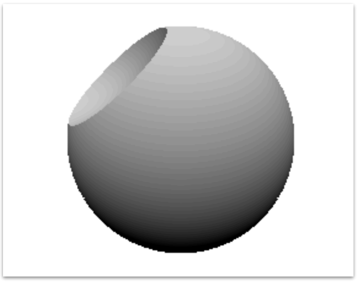
Rendering the Carved Ball

- Command: `rt < s.scn > s.ras`



```
scene{
    camera = view {
        from = {0, 0, -4}, at = {0, 0, 0}, up = {0,1,0},
    light = dist_light {direction = {0, 1, -1} },
    object = csgobj{
        material = plastic {  ka = .2, kd = 0.8, ks = 0.0
        shape = csg_diff {
                csg_prim{ sphere { center = {0, 0,  0}}},
                csg_prim{ sphere { center = {1, 1, -1}}}
            }
    }
}
```

s.scn



s.ras

System C

Modeling by Primitives

- Scene Description
  - Transformation Hierarchy
  - Primitive Objects
  - Camera, Etc

```
hier {
        transform { translate = { .5, .5, 0}},
        group {
                transform { zrotate = .4 },
                obj = sphere{ },
                transform { translate = {.2, 0, 1}},
                group {
                        transform{ scale = {2, 0.4, 1}},
                        obj = sphere{ radius = .1} } }
};
```

Example of Scene - h.scn

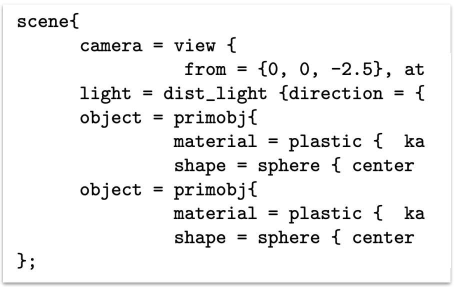
Painter’s Algorithm

- Z-Sort

```
int main(int argc, char **argv)
{
  Poly *l, *p, *c = poly_alloc(3);
  Item *i;
  init_sdl();
  s = scene_read();
  init_render();
  for (o = s->objs; o != NULL; o = o->next) {
     ....
  }
  z = z_sort(z);
  for (i = z->head; i != NULL; i = i->next) {
    gouraud_shade(c, P(i), N(i), s->view->center, rc, M(i));
    p = poly_homoxform(S(i),mdpy);
    scan_poly(p, gouraud_paint, gouraud_set(g\c c,s->img));
  }
  img_write(s->img, "stdout", 0);
  exit(0);
}
```

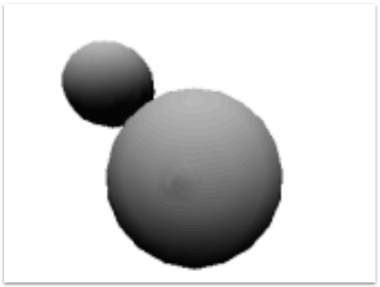
Rendering the Primitives Scene

- Command: `zsort < pr.scn > pr.ras`



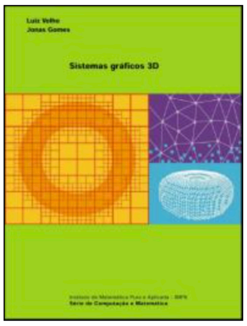
```
scene{
      camera = view {
                 from = {0, 0, -2.5}, at
      light = dist_light {direction = {
      object = primobj{
                material = plastic {  ka
                shape = sphere { center
      object = primobj{
                material = plastic {  ka
                shape = sphere { center
};
```

pr.scn



pr.ras

Follow-Up



Sistemas Gráficos 3D

Luiz Velho e Jonas Gomes

330 páginas
Série de Computação e Matemática
IMPA, 2001
ISBN 85-244-0167-2