Ray Casting

MIT EECS 6.837
Frédo Durand and Barb Cutler
Some slides courtesy of Leonard McMillan

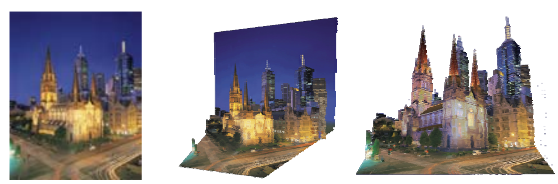
Luxo Jr

- Pixar Animation Studios, 1986
- Director: John Lasseter

Games

Image-based Rendering

- Use images as inputs and representation
  - E.g. Image-based modeling and photo editing
    Boh, Chen, Dorsey and Durand 2001



Input image New viewpoint Relighting

Administrative

- Assignment 1
  - Due Wednesday September 17

Calendar

- ***1st quiz – Tuesday October 07th***
- ***2nd quiz --Thursday Nov 20th***
- ***Week Dec 1-5 project presentation***
- ***Last day of class: December 9: best projects & final report due***

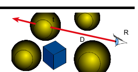
Questions?

Overview of the semester

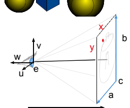
- Ray Tracing
  - Quiz 1
- Animation, modeling, IBMR
  - Choice of final project
- Rendering pipeline
  - Quiz 2
- Advanced topics

Overview of today

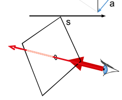
- Introduction



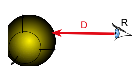
- Camera and ray generation



- Ray-plane intersection

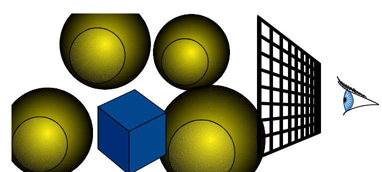


- Ray-sphere intersection



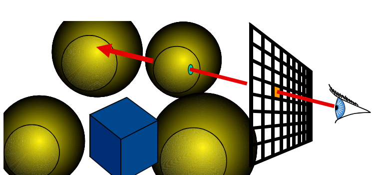
Ray Casting

```
For every pixel
  Construct a ray from the eye
  For every object in the scene
     Find intersection with the ray
     Keep if closest
```



Ray Casting

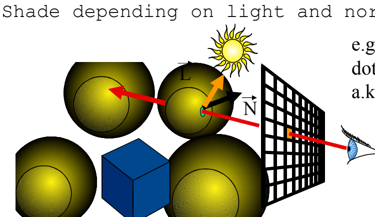
```
For every pixel
  Construct a ray from the eye
  For every object in the scene
     Find intersection with the ray
     Keep if closest
```



Shading

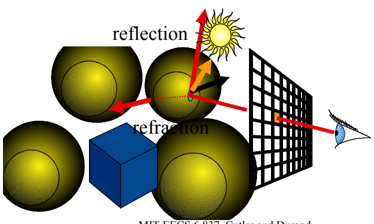
```
For every pixel
  Construct a ray from the eye
  For every object in the scene
     Find intersection with the ray
     Keep if closest
   Shade depending on light and normal vector
```

e.g. diffuse shading: dot product $\vec{N}.\vec{L}$
a.k.a. Lambertian

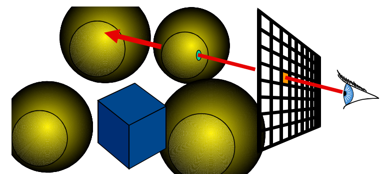


Ray Tracing

- Secondary rays (shadows, reflection, refraction)
- In a couple of weeks

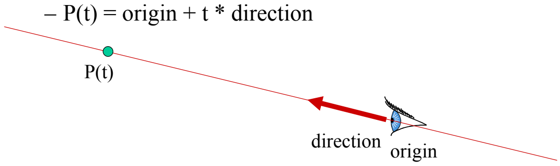


Ray representation?



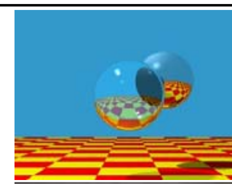
Ray representation

- Two vertors:
  - Origin
  - Direction (normalized is better)
- Parametric line
  - P(t) = origin + t * direction



Ray Tracing

- Original Ray-traced image by Whitted

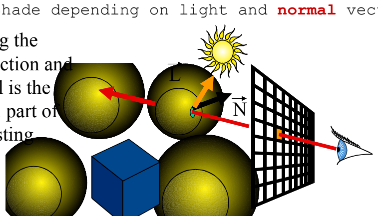


- Image computed using the Dali ray tracer by Henrik Wann Jensen
- Environment map by Paul Debevec

Ray casting

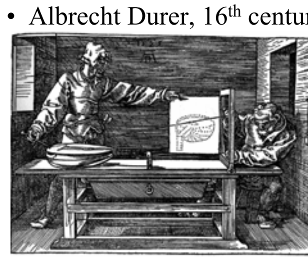
```
For every pixel
  Construct a ray from the eye
  For every object in the scene
      Find intersection with the ray
      Keep if closest
   Shade depending on light and normal vector
```

Finding the intersection and normal is the central part of ray casting



Durer's Ray casting machine

- Albrecht Durer, 16th century



Durer's Ray casting machine

- Albrecht Durer, 16th century

Durer's Ray casting machine

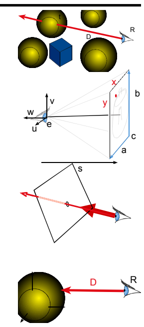
- Albrecht Durer, 16th century

Questions?

- Image computed using the Dali ray tracer from Henrik Wann jensen
- Model Stephen Duck

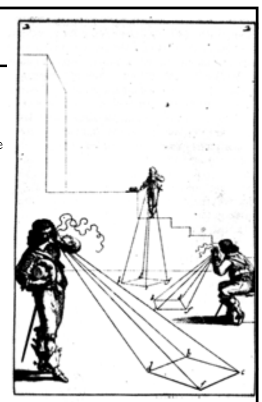
Overview of today

- Introduction
- Camera and ray generation
- Ray-plane intersection
- Ray-sphere intersection



Cameras

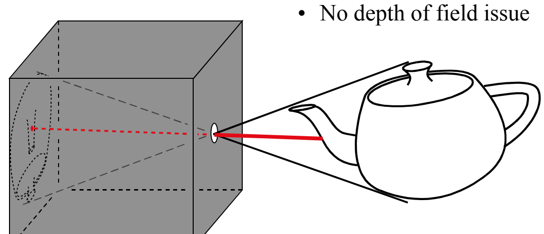
```
For every pixel
  Construct a ray from the eye
  For every object in the scene
     Find intersection with the
       ray
     Keep if closest
```



braham Bosse, *Les Perspecteurs*. Gravure extraite de la

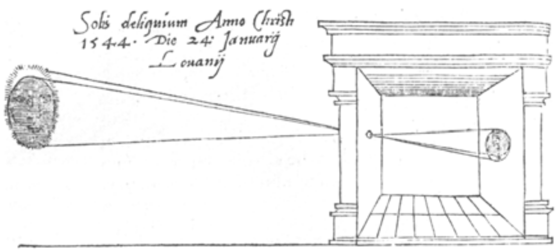
Pinhole camera

- Box with a tiny hole
- Inverted image
- Similar triangles
- Perfect image if hole infinitely small
- Pure geometric optics
- No depth of field issue



Oldest illustration

- From. R. Gemma Frisius, 1545



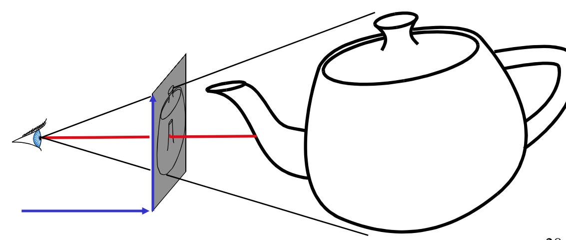
Camera Obscura

Abelardo Morell

- Photographer who turns hotel room into a camera obscura (pinhole optics)

Simplified pinhole camera

- Eye-image pyramid (frustum)
- Note that the distance/size of image are arbitrary

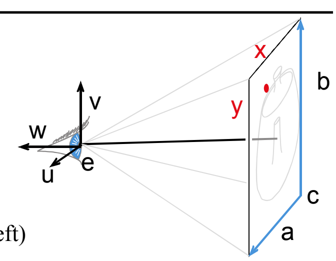


Durer's Ray casting machine

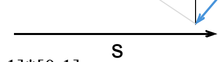
- Albrecht Durer, 16th century

Camera description

- Eye point e
- Orthobasis u, v, w
- Image distance s
- Image rectangle (u0, v0, u1, v1)



- Deduce c (lower left)
- Deduce a and b
- Screen coordinates in [0,1]*[0,1]
- A point is then c + x a +y b

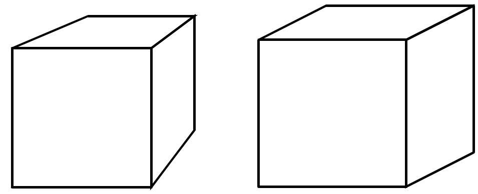


Alternative perspective encoding

- 4x4 matrix & viewing frustum
- More about that next week

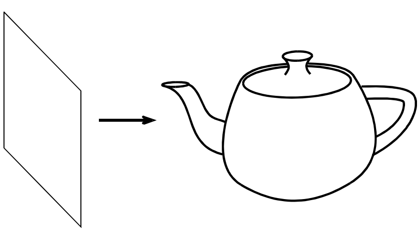
Orthographic camera

- Parallel projection
- No foreshortening
- No vanishing point



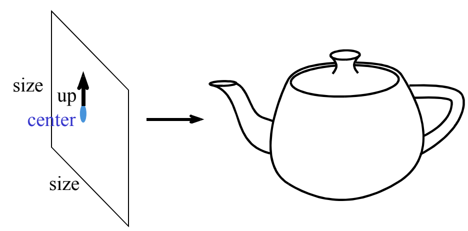
perspective orthographic

Orthographic camera description



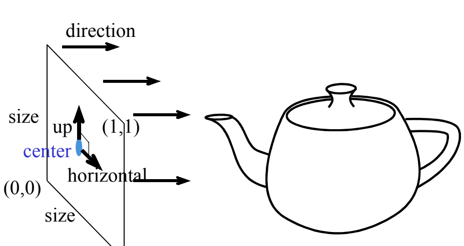
Orthographic camera description

- Direction
- Image center
- Image size
- Up vector



Orthographic ray generation

- Direction is constant
- Origin = center + (x-0.5)*size*up + (y-0.5)*size*horizontal



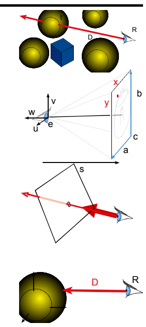
Other weird cameras

- E.g. fish eye, omnimax, panorama

Questions?

Overview of today

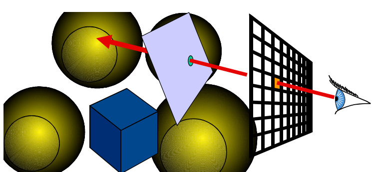
- Introduction
- Camera and ray generation
- Ray-plane intersection
- Ray-sphere intersection



Ray Casting

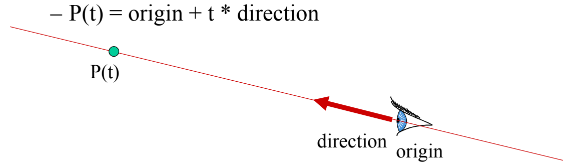
```
For every pixel
  Construct a ray from the eye
  For every object in the scene
      Find intersection with the ray
      Keep if closest
```

First we will study ray-plane intersection



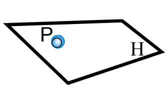
Recall: Ray representation

- Two vectors:
  - Origin
  - Direction (normalized)
- Parametric line
  - P(t) = origin + t * direction



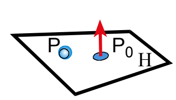
3D plane equation

- Implicit plane equation
  H(p) = Ax+By+Cz+D = 0
- Gradient of H?



3D plane equation

- Implicit plane equation
  H(p) = Ax+By+Cz+D = 0
- Gradient of H?
- Plane defined by
  - P0(x,y,z,1)
  - n(A, B, C, 1)

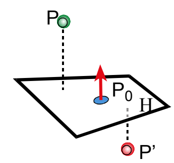


Explicit vs. implicit?

- Plane equation is implicit
  - Solution of an equation
  - Does not tell us how to generate a point on the plane
  - Tells us how to check that a point is on the plane
- Ray equation is explicit
  - Parametric
  - How to generate points
  - Harder to verify that a point is on the ray

Plane-point distance

- Plane Hp=0
- If n is normalized
  d=HP
- Signed distance!

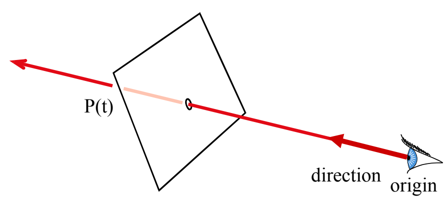


Explicit vs. implicit?

- Plane equation is implicit
  - Solution of an equation
  - Does not tell us how to generate a point on the plane
  - Tells us how to check that a point is on the plane
- Ray equation is explicit
  - Parametric
  - How to generate points
  - Harder to verify that a point is on the ray
- Exercise: explicit plane and implicit ray

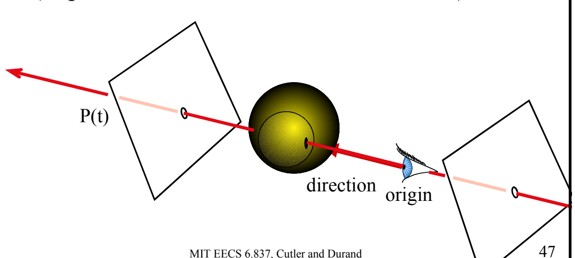
Line-plane intersection

- Insert explicit equation of line into implicit equation of plane



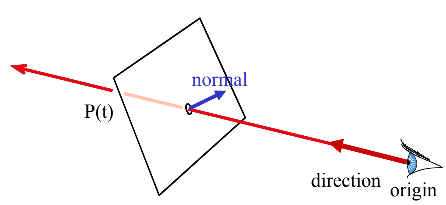
Additional house keeping

- Verify that intersection is closer than previous
- Verify that it is in the allowed range
  (in particular not behind the camera, t<0)



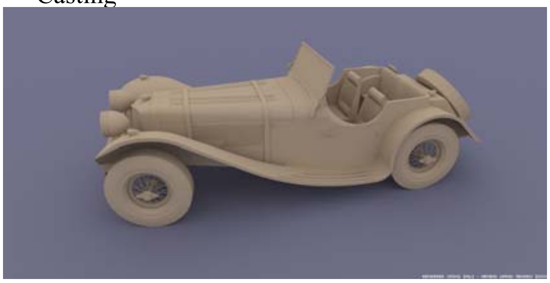
Normal

- For shading (recall, diffuse: dot product between light and normal)
- Simply the normal to the plane



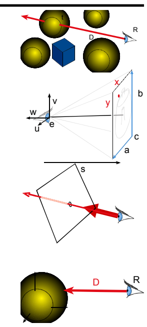
Questions?

- Image by Henrik Wann Jensen using Ray Casting



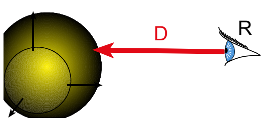
Overview of today

- Introduction
- Camera and ray generation
- Ray-plane intersection
- Ray-sphere intersection



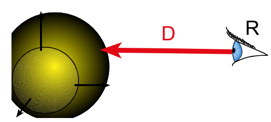
Sphere equation

- Sphere equation (implicit): $||P||^2 = r^2$
- (assume centered at origin, easy to translate)



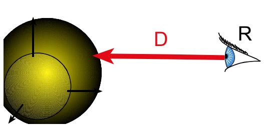
Ray-Sphere Intersection

- Sphere equation (implicit): $||P||^2 = r^2$
- Ray equation (explicit): $P(t) = R+tD$ with $||D|| = 1$
- Intersection means both are satisfied



Ray-Sphere Intersection

$$\begin{aligned} 0 &= \mathbf{P}\cdot\mathbf{P} - r^2 \\ &= (\mathbf{R}+t\mathbf{D})\cdot(\mathbf{R}+t\mathbf{D}) - r^2 \\ &= \mathbf{R}\cdot\mathbf{R} + 2t\mathbf{D}\cdot\mathbf{R} + t^2\mathbf{D}^2 - r^2 \\ &= t^2 + 2t\mathbf{D}\cdot\mathbf{R} + \mathbf{R}\cdot\mathbf{R} - r^2 \end{aligned}$$



Ray-Sphere Intersection

- This is just a quadratic $at^2 + bt + c = 0$, where
  - $a = 1$
  - $b = 2D.R$
  - $c = R.R - r^2$
- With discriminant

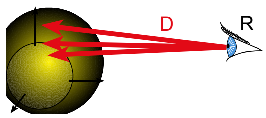
$$d = \sqrt{b^2 - 4ac}$$

- and solutions

$$t_{\pm} = \frac{-b \pm d}{2a}$$

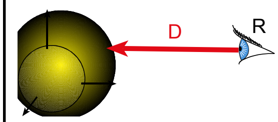
Ray-Sphere Intersection

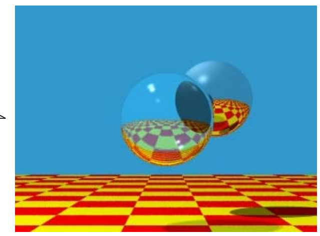
- Discriminant $d = \sqrt{b^2 - 4ac}$
- Solutions $t_{\pm} = \frac{-b \pm d}{2a}$
- Three cases, depending on sign of $b^2 - 4ac$
- Which root (t+ or t-) should you choose?
  - Closest positive! (usually t-)



Ray-Sphere Intersection

- So easy that all ray-tracing images have spheres!

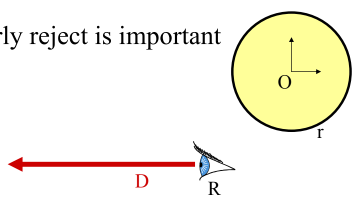




Geometric ray-sphere intersection

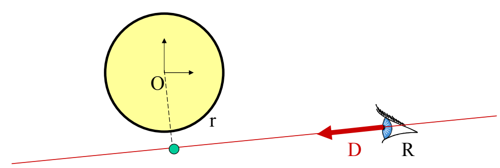
- Try to shortcut (easy reject)
- e.g.: if the ray is facing away from the sphere
- Geometric considerations can help

- In general, early reject is important



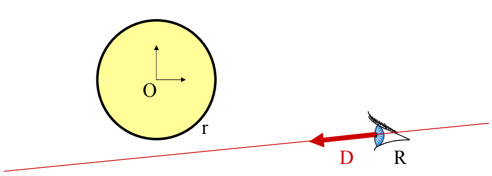
Geometric ray-sphere intersection

- What geometric information is important?
  - Inside/outside
  - Closest point
  - Direction



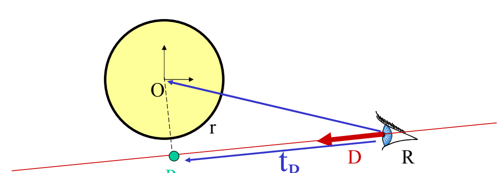
Geometric ray-sphere intersection

- Find if the ray's origin is outside the sphere
  - $R^2 > r^2$
  - If inside, it intersects
  - If on the sphere, it does not intersect (avoid degeneracy)



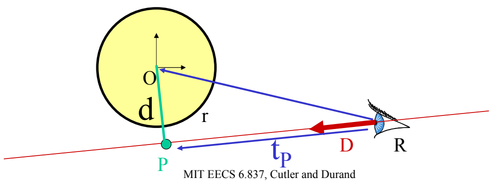
Geometric ray-sphere intersection

- Find if the ray's origin is outside the sphere
- Find the closest point to the sphere center
  - $t_P = RO.D$
  - If $t_P < 0$, no hit



Geometric ray-sphere intersection

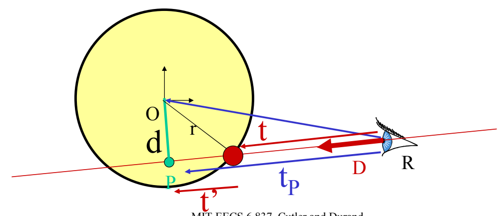
- Find if the ray's origin is outside the sphere
- Find the closest point to the sphere center
  - If $t_P<0$, no hit
- Else find squared distance d^2
  - Pythagoras: $d^2=R^2-t_P^2$
  - … if $d^2> r^2$ no hit



Geometric ray-sphere intersection

- Find if the ray's origin is outside the sphere
- Find the closest point to the sphere center
  - If $t_P<0$, no hit
- Else find squared distance d^2
  - if $d^2 > r^2$ no hit

- If outside $t = t_P-t'$
  - $t'^2+d^2=r^2$
- If inside $t = t_P+t'$

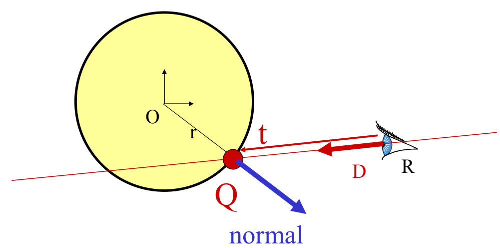


Geometric vs. algebraic

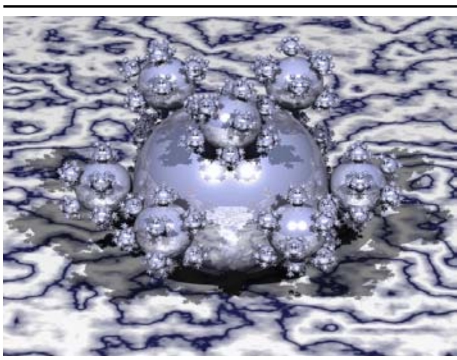
- Algebraic was more simple
  (and more generic)
- Geometric is more efficient
  - Timely tests
  - In particular for outside and pointing away

Normal

- Simply Q/||Q||

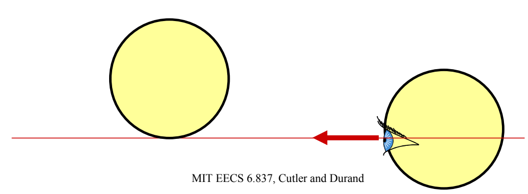


Questions?



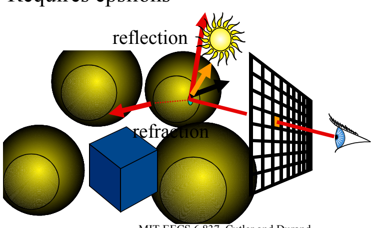
Precision

- What happens when
  - Origin is on an object?
  - Grazing rays?
- Problem with floating-point approximation



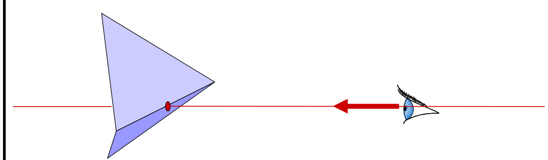
The evil ε

- In ray tracing, do NOT report intersection for rays starting at the surface (no false positive)
  - Because secondary rays
  - Requires epsilons



The evil ε: a hint of nightmare

- Edges in triangle meshes
  - Must report intersection (otherwise not watertight)
  - No false negative

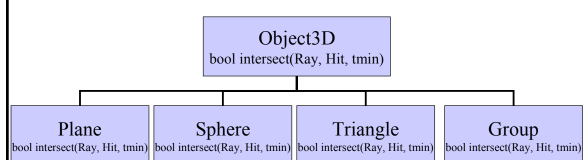


Assignment 1

- Write a basic ray caster
  - Orthographic camera
  - Spheres
  - Display: constant color and distance
- We provide
  - Ray
  - Hit
  - Parsing
  - And linear algebra, image

Object-oriented design

- We want to be able to add primitives easily
  - Inheritance and virtual methods
- Even the scene is derived from Object3D!



Ray

```
//////////
class Ray {
//////////

  Ray () {}
  Ray (const Vec3f &dir, const Vec3f &orig)
  {_dir =dir; _orig=orig;}
  Ray (const Ray& r) {*this=r;}

  const Vec3f &origin() {return _orig;}
  const Vec3f &direction() {return &dir;}
  Vec3f pointAtParameter (float t) {return _orig+t*_dir;}

private:
  Vec3f _dir;
  Vec3f _orig;

};
```


Hit

- Store intersection point & various information

```
/////////
class Hit {
/////////

  float _t;
  Vec3f _color;
  //Material *_material;
  //Vec3f _normal;
};
```

Tasks

- Abstract Object3D
- Sphere and intersection
- Scene class
- Abstract camera and derive Orthographic
- Main function

Thursday: More Ray Casting

- Other primitives
  - Boxes
  - Triangles
  - IFS?
- Antialiasing