Tutorial 6
Introduction to Assignment 3

Zhanrui Zhang

Ray tracing basics

- Step.1 generate ray from camera
- Step.2 intersect with geometry in the scene.
- Step.3 Visibility test to generate shadow.
- Step.4 Compute radiance at interaction point.

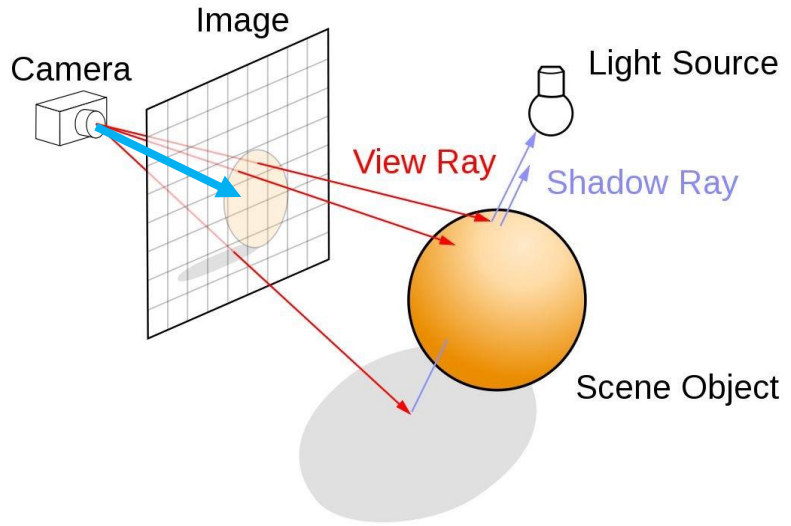
Generate ray from camera

- Definition of camera: position, forward, up, right, FoV, focal length.
- In this assignment, we assume focal length is 1.
- We define a camera by set its position, fov, and then call lookAt().
- Vector look_at is the coordinate of a point in world space.
- You need to update the camera whenever lookAt is called.

```
void lookAt(const Vec3f &look_at, const Vec3f &ref_up = {0, 0, 1});
void setPosition(const Vec3f &pos);
void setFov(float new_fov);
```

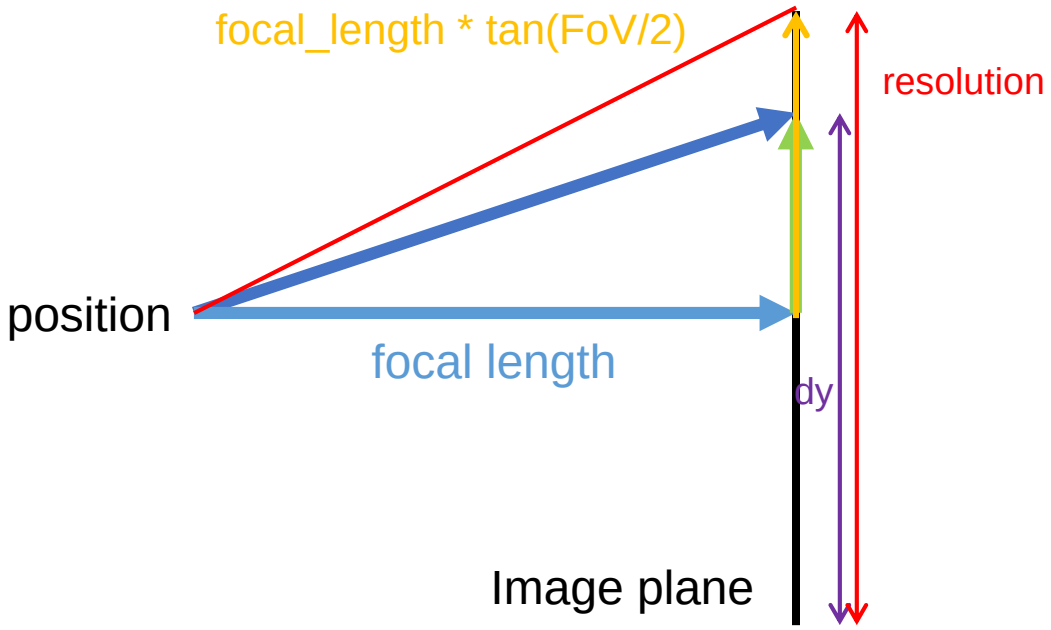
Generate ray from camera

- The ray shooting from the camera is called camera/view ray.
- Ray: $\vec{o} + t \cdot \vec{d}$, where $\vec{o}$ is the origin point, $\vec{d}$ is the direction, t is the distance.
- Origin of the ray is the position of the camera.
- Direction of the ray is vector (position of camera -> pixel coordinate).



Generate ray from camera

- Compute the pixel coordinate in world space.



Ray Geometry Intersection – Rectangle

- Consider the plane equation: $(P - P_0) \cdot \vec{n} = 0$
- Ray: $\vec{o} + t \cdot \vec{d}$
- $\left(\vec{o} + t \cdot \vec{d} - P_0\right) \cdot \vec{n} = 0$, so we can compute t.
- Then intersection point P is known.
- Next, judge whether intersection point is in the rectangle.
- Compute dot product between $P - P_0$ and tangent / cotangent.
- Compare the dot product with size.x and size.y of rectangle.

Ray Geometry Intersection – Ellipsoid

- Intersection test for sphere is easy.
- Idea: transform an ellipsoid to a sphere.

Ray: $P = P_0 + tV$
Sphere: $|P - O|^2 - r^2 = 0$

$L = O - P_0$

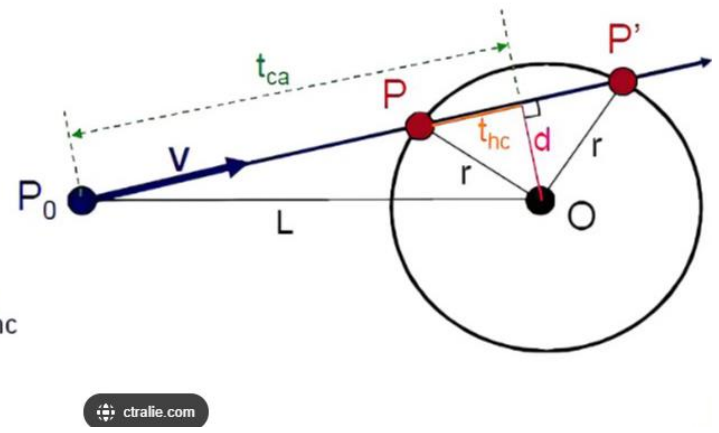
$t_{ca} = L \bullet V$
if $(t_{ca} < 0)$ return 0

$d^2 = L \bullet L - t_{ca}^2$
if $(d^2 > r^2)$ return 0

$t_{hc} = \text{sqrt}(r^2 - d^2)$
$t = t_{ca} - t_{hc}$ and $t_{ca} + t_{hc}$

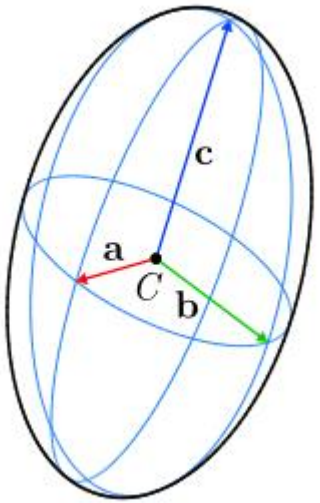
$P = P_0 + tV$

Geometric Method



Ray Geometry Intersection – Ellipsoid

- Intersection test for sphere is easy.
- Idea: transform an ellipsoid to a sphere.



$$\mathbf{T} = \begin{bmatrix} 1 & 0 & 0 & C_x \\ 0 & 1 & 0 & C_y \\ 0 & 0 & 1 & C_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad \mathbf{R} = \begin{bmatrix} \hat{a}_x & \hat{b}_x & \hat{c}_x & 0 \\ \hat{a}_y & \hat{b}_y & \hat{c}_y & 0 \\ \hat{a}_z & \hat{b}_z & \hat{c}_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad \mathbf{S} = \begin{bmatrix} \|\mathbf{a}\| & 0 & 0 & 0 \\ 0 & \|\mathbf{b}\| & 0 & 0 \\ 0 & 0 & \|\mathbf{c}\| & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

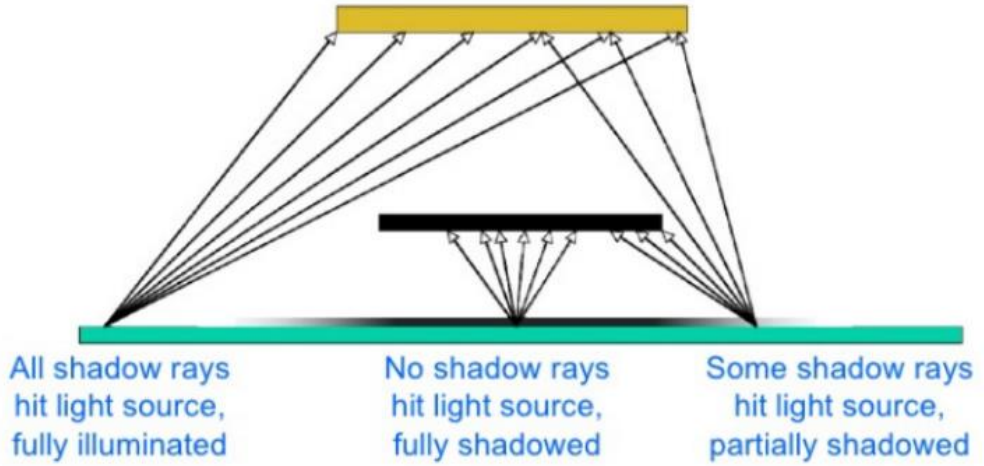
$$\mathbf{M} = \mathbf{TRS}$$

M: transform from unit sphere to ellipsoid.

So we need to apply M^{-1} to the ray (direction and origin).

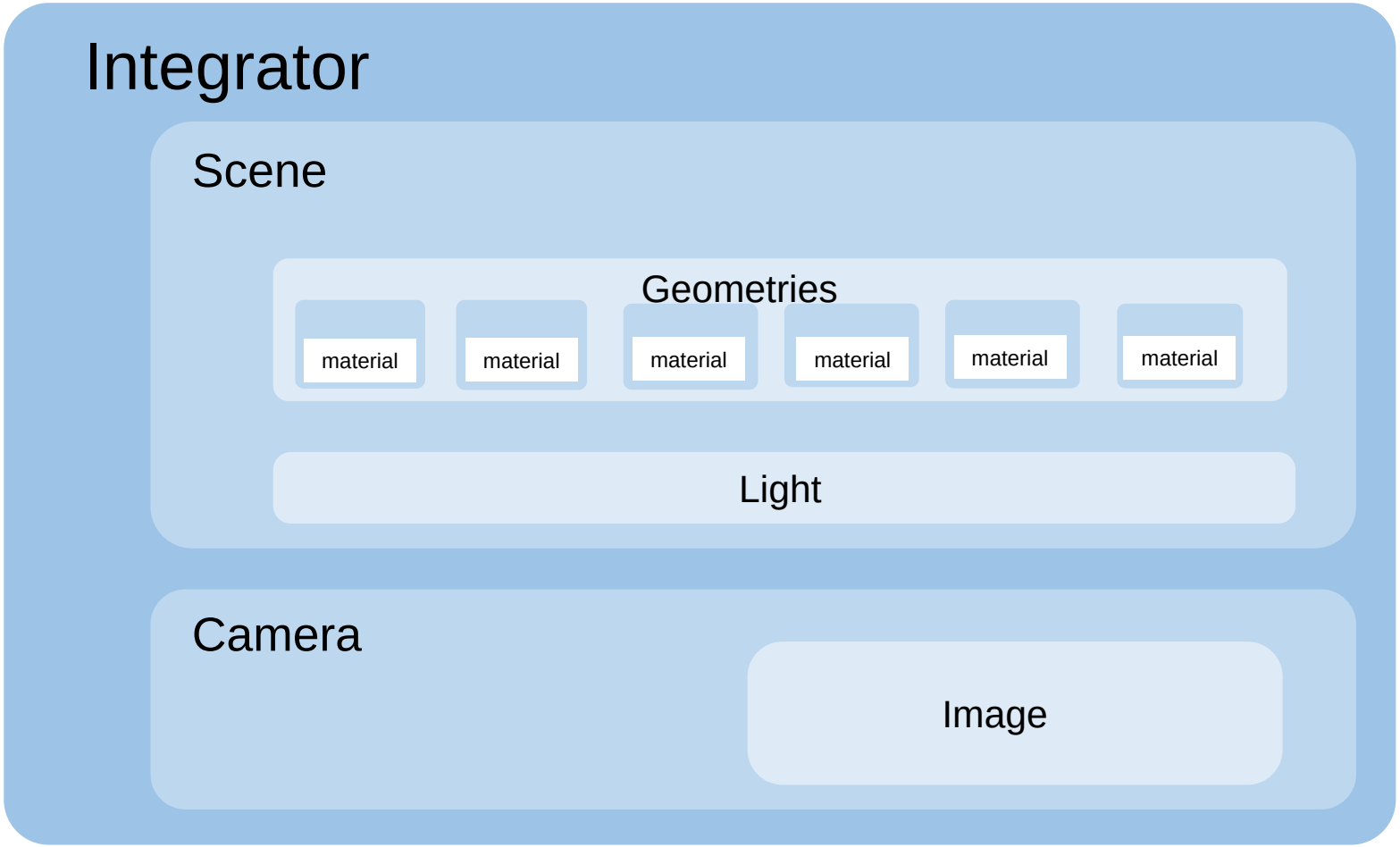
Visibility (to light source) Test

- Sample the light
- Shadow ray from object surface to light sample.
- Visibility test for each shadow ray.
- Pay attention to shadow eps.



Introduction to skeleton code

- Files you need to modify to finish must part:
  - camera.cpp
  - geometry.cpp
  - integrator.cpp
  - light.cpp
  - material.cpp
  - scene.cpp



Eigen

- Use (row, col) as matrix index.

```
float dot_product = vec1.dot(vec2);
Vec3f cross_product = vec1.cross(vec2);
Vec3f new_vec = vec1.normalized();
vec1.normalize();
Vec3f column_wise = vec1.cwiseProduct(vec2);
float length_of_vec = vec1.norm();
```

About code skeleton…

About Bonus

You need to demonstrate the effectiveness of your implementation of texture filtering/anti-aliasing.

You can add a new geometry called ground and apply a grid texture on it.

For texture filtering, you need to compute ray differentials

Check this paper from SIGGRAPH 1999: https://graphics.stanford.edu/papers/trd/trd_jpg.pdf

About autograder…

Some tips

- Rendering images in assignment 3 and assignment 4 may require a decent CPU.
- Try to use lower resolution and turn off super resolution when debugging if your computer is slow.
- START EARLY!