Imatge Sintètica

Ray Tracing for Realistic Image Synthesis

Ricardo Marques
(ricardo.marques@upf.edu)

Group de Tecnologies Interactives (GTI)
Departament de Tecnologies de la Informació i les Comunicacions (DTIC)
Universitat Pompeu Fabra (UPF)

Edifici Tànger - Office 55.106

Lecture 2 - Camera Rays and Ray-Sphere Intersection

2017/2018

Class Outline

Lecture 2 - Camera Rays and Ray-Sphere Intersection

Last Class Summary

Simple Ray Tracing

- Coordinate Spaces
- Camera Models
- Camera Rays
- Ray-Sphere Intersection

Next Classes

Section 1

Last Class Summary

Last Class Summary

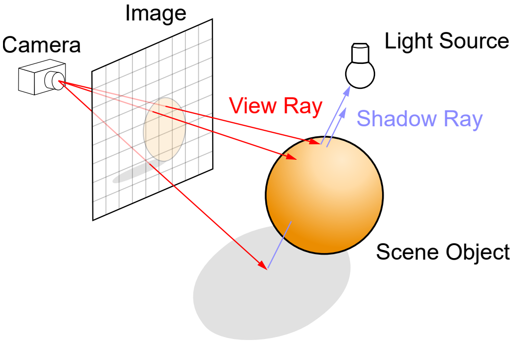
- We have seen:
  - 3D points, 3D vectors and normals and other useful 3D entities
  - Simple vector and matrix operations
  - Common 3D transformations
  - How to transform points, vectors and normals
  - The notion of image
- This gives us the basic tools to learn ray tracing!

Section 2

Simple Ray Tracing

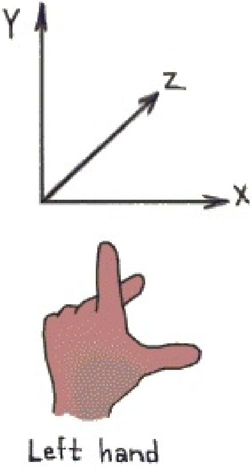
Recall: The Ray Tracing Principle

- Use rays to compute the light that arrives at each image pixel
  - Simulate the light propagation from light sources
  - Model the light interaction with the scene materials



Recall: 3D Frame

- To represent points and vectors in a 3D space we need a 3D frame
- A 3D frame is a coordinate system represented by an origin and three base vectors
- World frame
  - Specifies the world space coordinates
  - Origin $(x, y, z) = (0, 0, 0)$
  - Base vectors $(1, 0, 0)$, $(0, 1, 0)$ and $(0, 0, 1)$
- In this course we will use a *left-handed* coordinate system
- All other frames (e.g., object's or camera's local frames) must be defined with respect to the world frame



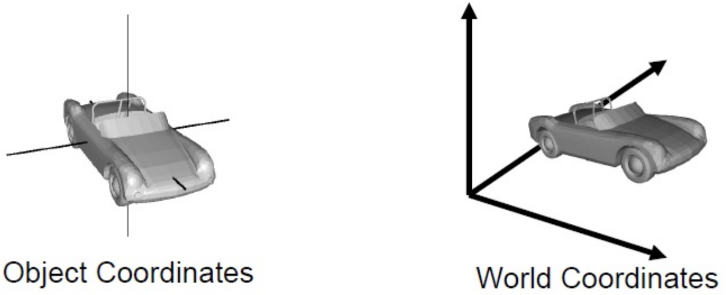
[Image from George H. Otto]

Subsection 1

Coordinate Spaces

World Space and Object Space

- A stand alone object is expressed in object space
  - The origin of the object frame is usually the object center
- Placing that object in a scene consists of specifying how to pass from object space to world space

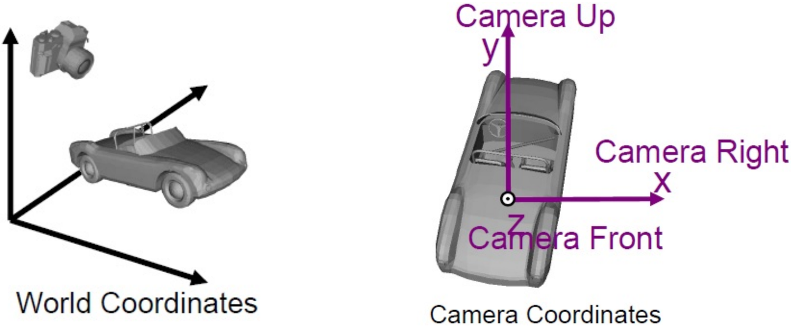


[Image from Misha Kazhdan]

- All objects have *objectToWorld* and *worldToObject* matrices

World Space and Camera Space

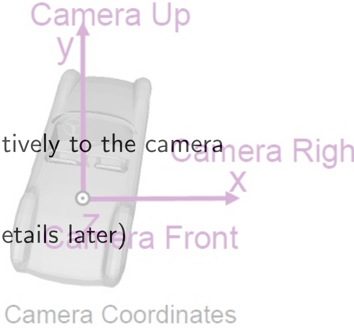
- Camera space
  - Coordinate system based upon the viewpoint of the observer



[Image from Misha Kazhdan]

- All cameras have *cameraToWorld* and *worldToCamera* matrices

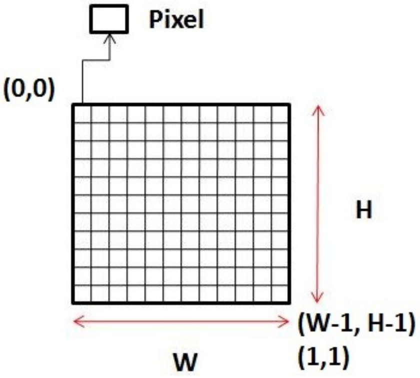
Camera Space



- Camera space $((x, y, z) \in \mathbb{R}^3)$
  - Expresses directions and locations relatively to the camera position
  - Handy space to generate rays (more details later)
- Assumptions of the **camera space**
  - The camera is looking toward the z-axis
  - The x-axis gives the camera right direction
  - The y-axis gives the camera up direction
  - The camera position in camera coordinates is $(0, 0, 0)$

Image Space and NDC

- Image space $((x, y) \in [0, W-1] \times [0, H-1])$
  - (x, y) are pixel coordinates
- NDC space $\left((x, y) \in [0, 1]^2\right)$
  - Similar to Image Space
  - Also defined in the image plan
  - $(0, 0)$ is the upper-left corner



Summary Coordinate Spaces

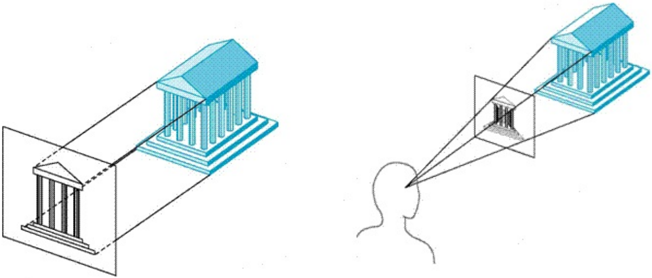
- World space: main frame
- Object space: expresses object components with respect to the object (local) frame
- Camera space: expresses directions and locations with respect to the camera position and orientation
- Image space and NDC space

Subsection 2

Camera Models

Cameras

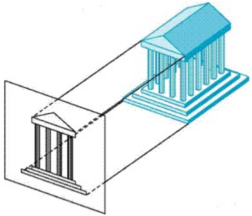
- Camera: specifies how the scene is "seen"
- Project a 3D region of the scene onto a 2D image plane
- Examples: orthographic and perspective cameras
  - Two different ways of projecting



[Image from Loren K. Rhodes, http://jcsites.juniata.edu/faculty/rhodes/]

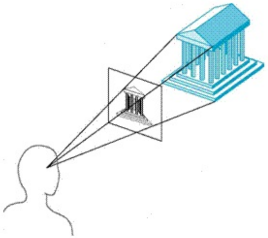
Orthographic Camera

- Extremely simple camera model
  - Handy for **early development stages and debugging**
- Mapping from *camera space* to *image space* amounts to projecting along an axis (usually the **z**-axis)
  - Parallel lines in the 3D scene remain parallel in the 2D image
  - Preserves the relative distance between objects
  - No *foreshortening* effect (distant objects do not look smaller)



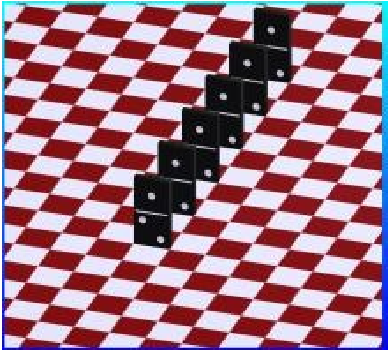
Perspective Camera

- Slightly more complex than the orthographic camera
  - Includes the *foreshortening* effect (distant objects are seen smaller)
  - Does not preserve distances between objects
  - Parallel lines do not remain parallel in the image
  - More realistic camera model

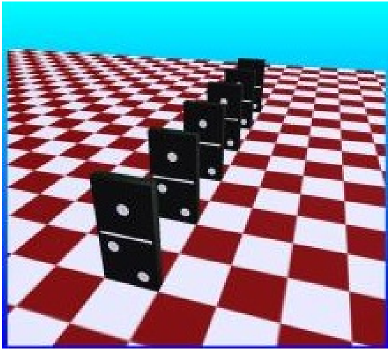


Orthografic VS Perspective Camera

[Images from Ian Griffiths In Weblog Form]



Orthographic Camera



Perspective Camera

Subsection 3

Camera Rays

3D Ray

- A ray is a semi-infinite line
  - A point specifies the origin ($\mathbf{o}$)
  - A vector specifies its direction ($\mathbf{d}$)
- The parametric representation of a ray is given by:

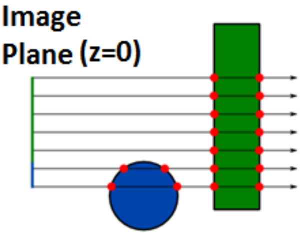
$$\mathbf{r}(t) = \mathbf{o} + t\,\mathbf{d}\,, \quad \text{with} \quad 0 \leq t < \infty$$

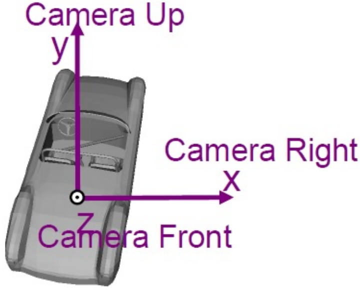
Camera Rays

- Camera rays are generated by the camera
  - Determine the visible objects from the viewer's position
- The camera ray generation depends on the used camera model
- For simplicity, rays are **generated in the camera space** and then transformed to world space

Camera Rays from Orthographic Camera

$$\mathbf{r}(t) = \mathbf{o} + t\,\mathbf{d}\,, 0 \leq t < \infty$$



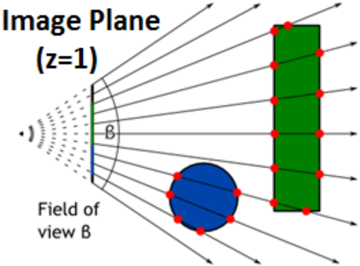


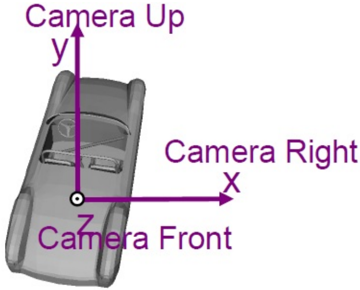
Camera Coordinates

- All rays have the same direction $\mathbf{d} = (0, 0, 1)$
- All ray origins lie on the image plane $(z = 0)$
  - $\mathbf{o} = (x, y, 0)$
  - x and y depend on the pixel coordinates
  - If $xRes = yRes$, then $x, y \in [-1, 1] \times [-1, 1]$

Camera Rays from Perspective Camera

$$\mathbf{r}(t) = \mathbf{o} + t\,\mathbf{d}\,, 0 \leq t < \infty$$



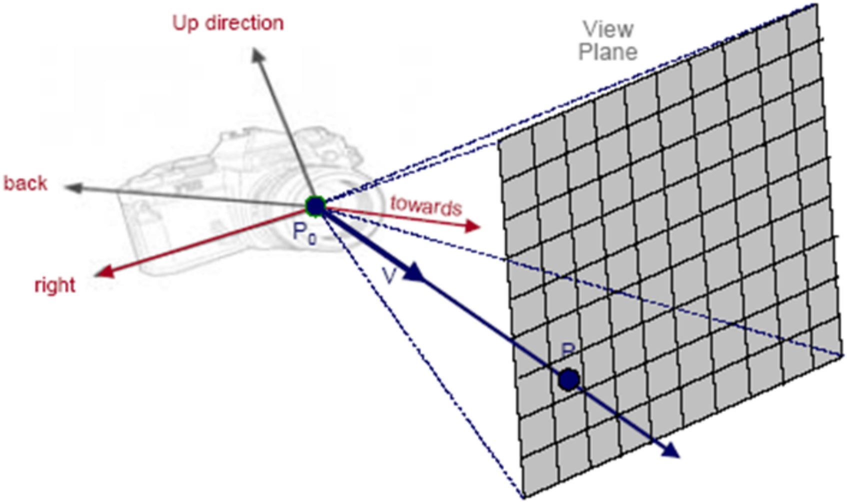


Camera Coordinates

- All rays have origin in the same point ($\mathbf{o} = (0, 0, 0)$)
- Given a point $\mathbf{p}$ on the image plane (in camera coordinates), the direction $\mathbf{d}$ of the camera ray $\mathbf{r}$ passing through $\mathbf{p}$ is

$$\mathbf{d} = \mathbf{p} - \mathbf{o}$$

Camera Rays from Perspective Camera

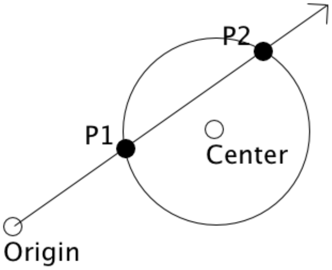


[Image from Kadi Bouatouch]

Subsection 4

Ray-Sphere Intersection

Intersecting Rays



- In the following examples, the intersection is computed in object coordinates
  - In object coordinates, the sphere center is always $(0,0,0)$
  - Simplifies computations
  - Rays must thus be transformed from world space to object space before computing the intersection

Ray-Sphere Intersection

- Let us consider a sphere of radius r centered at point $\mathbf{p}_c = (0, 0, 0)$
- Recall the expression of a 3D sphere centered at zero

$$x^2 + y^2 + z^2 = r^2$$

- Recall the expression of a ray

$$\begin{aligned}\mathbf{r}(t) &= \mathbf{o} + t\,\mathbf{d}\,, \quad 0 \leq t < \infty \\ &= (o_x, o_y, o_z) + t\,(d_x, d_y, d_z) \\ &= (o_x + t\,d_x, o_y + t\,d_y, o_z + t\,d_z)\end{aligned}$$

- Intersecting a ray with a sphere amounts to finding the points along the ray $\mathbf{r}$ which belong to the sphere
  - **Objective:** find the values of t, which satisfy both equations

Ray-Sphere Intersection

- Substituting the ray expression on the sphere expression, yields

$$(o_x + t\, d_x)^2 + (o_y + t\, d_y)^2 + (o_z + t\, d_z)^2 = r^2$$

- It can be shown that developing and rearranging the terms, the above Eq. is written as:

$$a\, t^2 + b\, t + c = 0$$

- The solutions t_{hit} (if any) give us the intersection points

Ray-Sphere Intersection

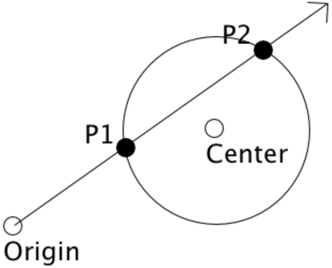
- The discriminant $\Delta = b^2 - 4ac$ of the ray-sphere intersection equation tells us the number of solutions
  - If $\Delta < 0$ then there are no real solutions (no intersection!)
  - If $\Delta > 0$ then there are two intersection points (we are interested on the closest one!)

$$t_{hit} = \frac{-b \pm \sqrt{\Delta}}{2a}$$

  - If $\Delta = 0$ then there is a single intersection point (rare case)

$$t_{hit} = -\frac{b}{2a}$$

Ray-Sphere Intersection



- If the ray direction **d** is normalized (i.e., $\|\mathbf{d}\| = 1$) then t_{hit} gives the intersection distance along the ray

Simple Ray Tracing Algorithm

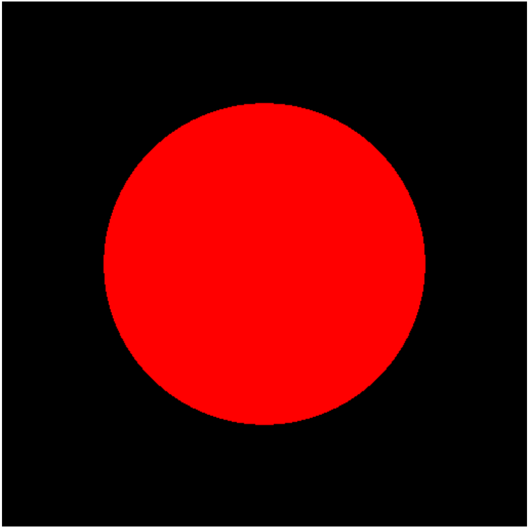
Exercice: What Does It Do?

```
for (line = 0; line ≤ height − 1; line++)
{
    for (col = 0; col ≤ width − 1; col++)
    {
        ray ← camera.generateRay(line, col);

        if(object.intersect(ray))
            image(col, line) = red;
        else
            image(col, line) = black;
    }

}
```

Simple Ray Tracing Algorithm



Lecture Summary

- We have seen basic ray tracing concepts such as:
  - Commonly used coordinate spaces
    - World space, object space, camera space
    - Image space, NDC
  - How to define a ray
  - The orthographic and perspective camera models
  - How to generate camera rays using these camera models
  - How to compute ray-sphere intersections
- Now we can launch rays!

Next Classes at a Glance

- Next Seminar and Practical Class
  - Consolidate notions learned in this Lecture (Assignment 2)
- Next Lecture: direct illumination without shadows
  - The concept of *diffuse point light source*
  - Diffuse and specular reflections
  - Materials