COS 344: L3 Chapter 4

Cobus Redelinghuys

University of Pretoria

26/02/2024

Practical Schedule

Assessment	Start By Date	Due	Marking 1	Marking 2
Practical 1	26/02/2024	11/03/2024	N/A	N/A
Practical 2	22/03/2024	04/04/2024	08/04/2024	15/04/2024
Practical 3	08/04/2024	22/04/2024	22/04/2024	29/04/2024
Practical 4	29/04/2024	20/05/2024	20/05/2024	27/05/2024
Homework Assignment	22/03/2024	03/06/2024	TBD	TBD

Remember the math recap session this afternoon online.

Introduction

Rendering

Process of taking in a set of objects and producing an array of pixels as output.

- Two types of rendering approaches:

Object-order rendering

Each **object** is considered in turn, and for each **object**, all the **pixels** that it influences are found and updated.

Image-order rendering

Each **pixel** is considered in turn, and for each **pixel**, all the **objects** that influence it are found and the pixel value is computed.

Section 4.1: Basic Ray-Tracing Algorithm

- A ray tracer works by:
  - Determining for each pixel which objects are "seen" by said pixel.
  - Thus, the object must intersect the *viewing ray*.

Viewing ray

A line that emanates from the viewpoint, in the direction the pixel is looking in.

- Does the viewing ray stop at the first object it intersects?

Section 4.1: Basic Ray-Tracing Algorithm

- A ray tracer works by:
  - Determining for each pixel which objects are "seen" by said pixel.
  - Thus, the object must intersect the *viewing ray*.

Viewing ray

A line that emanates from the viewpoint, in the direction the pixel is looking in.

- Does the viewing ray stop at the first object it intersects?
  - What if the object is translucent?

Section 4.1: Basic Ray-Tracing Algorithm

- A ray tracer works by:
  - Determining for each pixel which objects are "seen" by said pixel.
  - Thus, the object must intersect the *viewing ray*.

Viewing ray

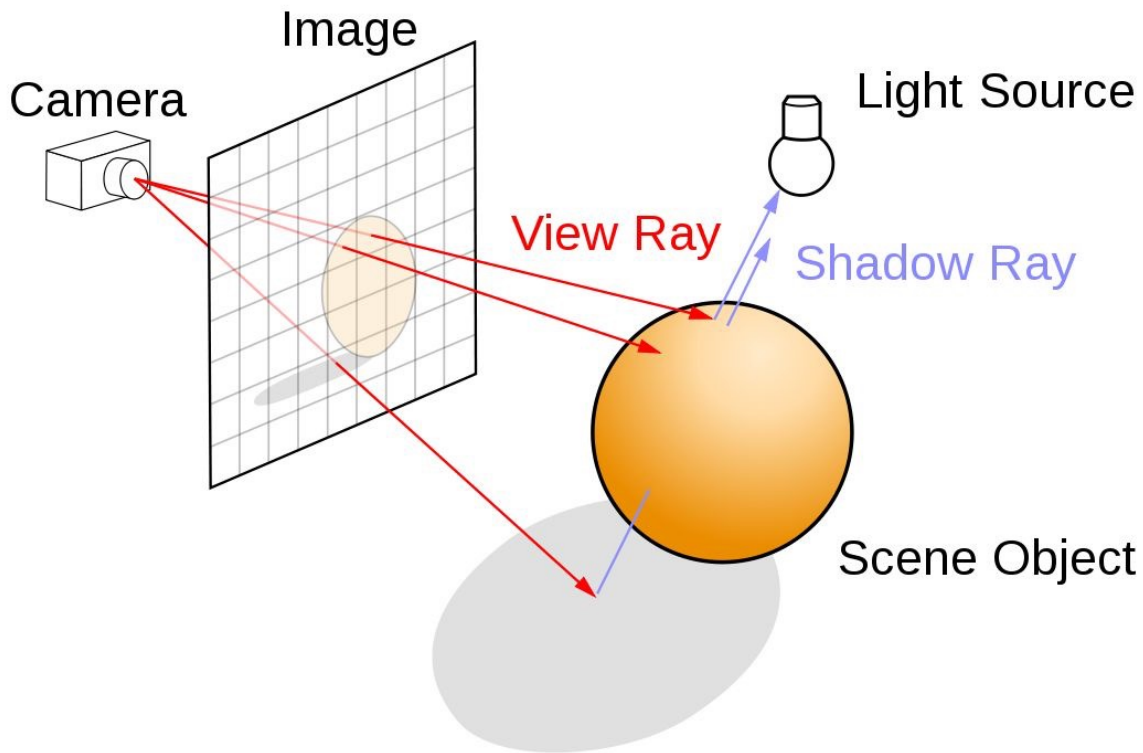
A line that emanates from the viewpoint, in the direction the pixel is looking in.

- Does the viewing ray stop at the first object it intersects?
  - What if the object is translucent?
- Once the object is found, shading is applied to the surface of the object.

- Basic components of ray tracer:
  1. Ray generation
     - Computes the origin, and direction of each pixel's viewing ray, based on camera geometry.
  2. Ray intersection
     - Finds the closest object intersecting the viewing ray. (*)
  3. Shading
     - Computes the pixel color based on the results of the intersection.

Algorithm 1 Basic Ray Tracing Algorithm

for each pixel **do**
  Compute viewing ray
  Find the first object that it intersects with
  Compute the normal **n** of the surface of the object the ray hit
  Set pixel color to the value computed using **n**, light, and hit point
end for



https://d29g4g2dyqv443.cloudfront.net/sites/default/files/pictures/2018/RayTracing/ray-tracing-image-1.jpg

Section 4.2: Perspective

- Many "approaches" exist for creating 3D shapes on a 2D shape.
  - Cubist Painting
  - Fish eye lenses
  - Peripheral cameras
  - Linear Perspective
    - Most common approach and used in computer graphics.

Section 4.2: Perspective

- Many "approaches" exist for creating 3D shapes on a 2D shape.
  - Cubist Painting
  - Fish eye lenses
  - Peripheral cameras
  - Linear Perspective
    - Most common approach and used in computer graphics.

Linear perspective

3D objects are projected onto an image plane, in such a way that straight lines in the scene become straight lines in the image.

- Types of projection:
  - Parallel projection
  - Perspective projection

Cubist Painting

Fisheye lens

https://www.canon.com.hk/public/article/Skill_Sharing_Photography/10351/images/02.jpg

Parallel Projection

Parallel projection

3D points are mapped to 2D points by moving them parallel along a projection direction, until they hit the image plane.

- Produced view is constructed by choosing the:
  - Projection direction
  - Image plane

Parallel Projection

Parallel projection

3D points are mapped to 2D points by moving them parallel along a projection direction, until they hit the image plane.

- Produced view is constructed by choosing the:
  - Projection direction
  - Image plane

Orthographic

The image plane is perpendicular to the view direction.

Obligue

The image plane is **not** perpendicular to the view direction.

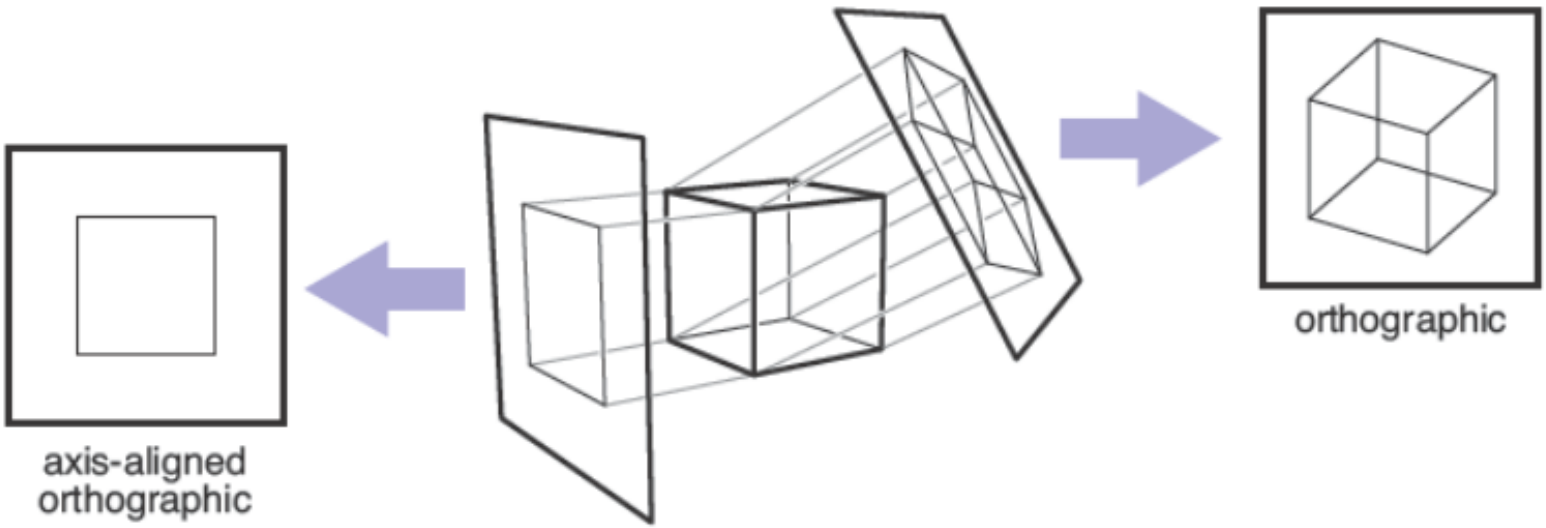


Figure 4.3. When projection lines are parallel and perpendicular to the image plane, the resulting views are called orthographic.

- Where can parallel projection be used?

- Where can parallel projection be used?
  - Mechanical and architectural drawings.
- Why?

- Where can parallel projection be used?
  - Mechanical and architectural drawings.
- Why?
  - Parallel lines stay parallel.
  - Size and shapes are preserved.

Perspective Projection

Perspective projection

3D points are projected to 2D points along lines that pass through a single point, known as the viewpoint, rather than along parallel lines.

- Allows objects to appear smaller the further they are from the viewpoint when projected.
- Produced view is constructed by choosing the:
  - Viewpoint
  - Image plane

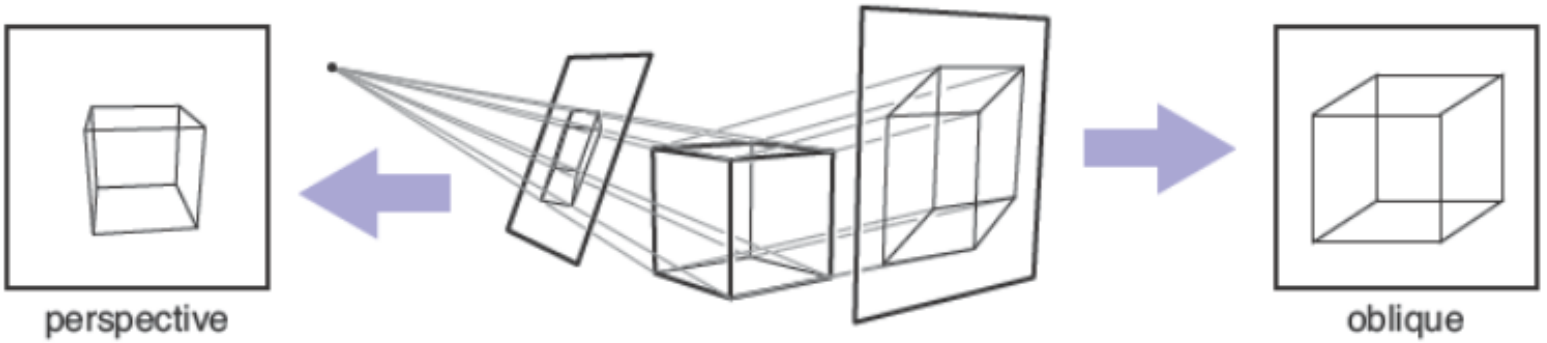


Figure 4.4. A parallel projection that has the image plane at an angle to the projection direction is called oblique (right). In perspective projection, the projection lines all pass through the viewpoint, rather than being parallel (left). The illustrated perspective view is non-oblique because a projection line drawn through the center of the image would be perpendicular to the image plane.

Section 4.3: Computing Viewing Rays

- Mathematical definition of a ray:

$$\mathbf{p}(t) = \mathbf{e} + t(\mathbf{s} - \mathbf{e})$$

$$\mathbf{p}(t) = (1 - t)\mathbf{e} + t(\mathbf{s})$$

  where:
  - $\mathbf{p}$ is a point
  - $\mathbf{e}$ is the start point (eye point)
  - $\mathbf{s}$ is the end point (surface intersect)
  - t is a real value bounded between 0 and 1
  - $\mathbf{s} - \mathbf{e}$ is the direction of the ray.
- Properties:
  - $\mathbf{p}(0) = \mathbf{e}$
  - $\mathbf{p}(1) = \mathbf{s}$
  - If $0 < t_1 < t_2$ then $\mathbf{p}(t_1)$ is closer to $\mathbf{e}$ than $\mathbf{p}(t_2)$.

Camera Frame

- Consider an orthonormal (camera frame) defined by:
  - **e** or the viewpoint.
  - **u** pointing rightward from e.
  - **v** pointing upward from e.
  - **w** pointing backward.
- $\{\mathbf{u}, \mathbf{v}, \mathbf{w}\}$ form a right-handed coordinate system.
- Common camera frame construction:
  - viewpoint (**e**)
  - view direction (-**w**)
  - up (**v**).
- This is used to construct a basis such that **v** and **w** are in the plane defined by the view direction and the up direction.
  - See Section 2.4.7

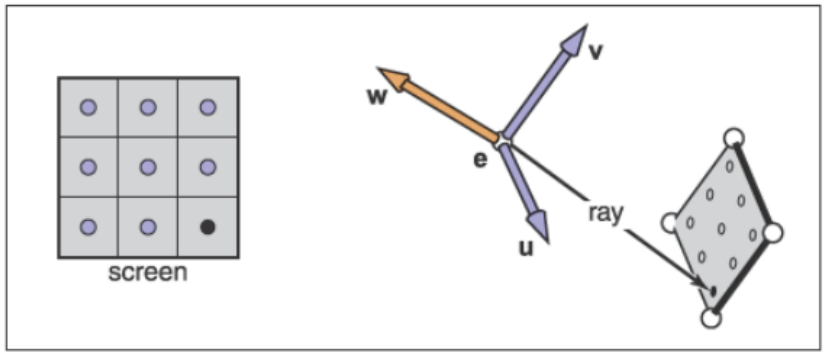


Figure 4.7. The sample points on the screen are mapped to a similar array on the 3D window. A viewing ray is sent to each of these locations.

This will be an example in this afternoon's session.

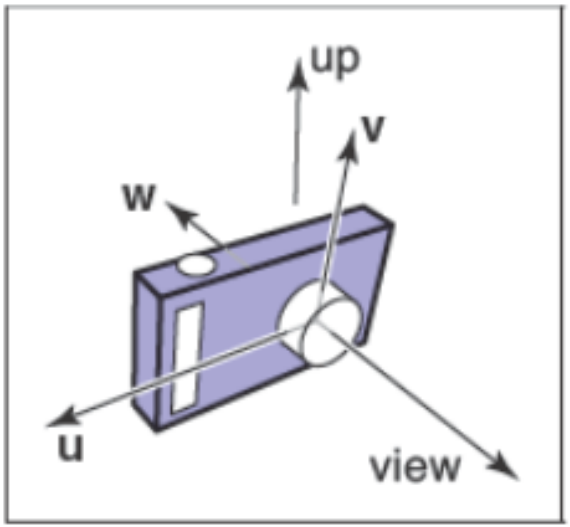


Figure 4.8. The vectors of the camera frame, together with the view direction and up direction. The w vector is opposite the view direction, and the v vector is coplanar with w and the up vector.

Section 4.3.1: Orthographic views

- Constructing an image plane:
  - l and r are the left and right edges of the plane, as measured from **e** along the **u** direction.
  - Usually: $l < 0 < r$
  - b and t are the bottom and top edges of the image as measured from **e** along the **v** direction.
  - Usually: $b < 0 < t$
- To translate a pixel at position (i, j) in pixel coordinates (Chapter 3) to (u, v) in the image plane use:
  - $u = l + \frac{(r-l)(i+0.5)}{n_x}$
  - $v = b + \frac{(t-b)(j+0.5)}{n_y}$
- (u, v) are coordinates on the image plane.
- Note the textbook overloads notation!

- Orthographic view ray:
  - Direction: -**w**.
  - Starting position: plane defined by **e**, **u**, and **v**.
  - **s** is on the image plane.
- Construct the orthographic viewing ray using:

Algorithm 2 Generating orthographic viewing rays

---

compute u and v using formulae on previous slide.
Ray origin := $\mathbf{e} + u\mathbf{u} + v\mathbf{v}$
Ray direction := -**w**

- To create oblique parallel view:
  - Allow the image plane's normal to be specified separately from the view direction.
  - Use the same approach as Algorithm 2, but with d instead of w.

Section 4.3.2: Perspective view

- In perspective view rays, all rays have the same origin (viewpoint), but different directions.
- Image plane is positioned some distance, d, away from e.
- d is referred to as the plane distance or focal length
- Ray direction is defined by the viewpoint and position on the image plane.

Algorithm 3 Generating perspective viewing rays

compute u and v using formulae on previous slides.
Ray origin := $\mathbf{e}$
Ray direction := $-d\mathbf{w} + u\mathbf{u} + v\mathbf{v}$

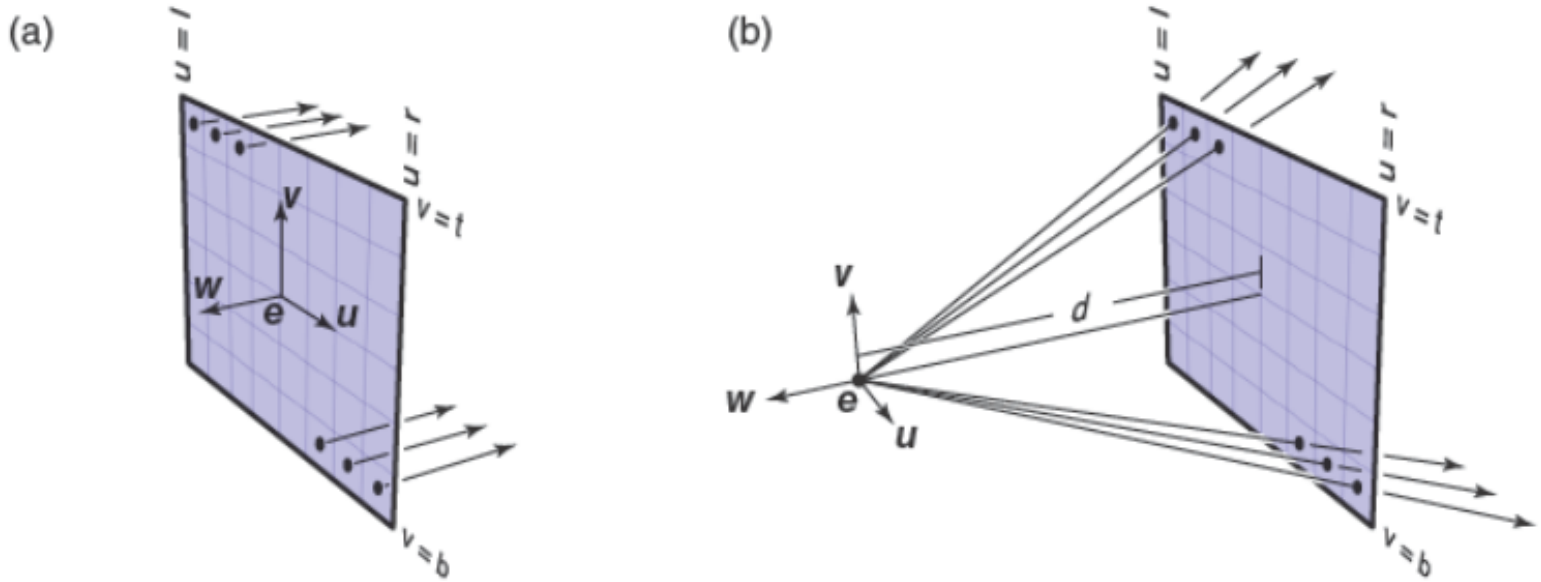


Figure 4.9. Ray generation using the camera frame. (a) In an orthographic view, the rays start at the pixels' locations on the image plane, and all share the same direction, which is equal to the view direction. (b) In a perspective view, the rays start at the viewpoint, and each ray's direction is defined by the line through the viewpoint, **e**, and the pixel's location on the image plane.

Section 4.4.1: Ray-Sphere Intersection

- To find where a ray ($\mathbf{p}(t) = \mathbf{e} + t\mathbf{d}$) intersects a sphere centered around $\mathbf{c}$ and has a radius of R:
  - Find the value of t for which the ray intersects the sphere.
  - Use the formula:

$$(\mathbf{d} \cdot \mathbf{d})t^2 + 2\mathbf{d} \cdot (\mathbf{e} - \mathbf{c})t + (\mathbf{e} - \mathbf{c}) \cdot (\mathbf{e} - \mathbf{c}) - R^2 = 0$$

  - Only unknown is thus t.
  - Solve t using the standard quadratic equation.
  - The derivation can be found in the textbook for the curious.

Section 4.4.2: Ray-Triangle Intersection

- To find where a ray ($\mathbf{p}(t) = \mathbf{e} + t\mathbf{d}$) intersects a triangle with vertices a, b, c solve:

$$e + td = a + \beta(b - a) + \gamma(c - a)$$

- Solve for: t, β, and γ.
  - Set up and solve a linear set of equations:

$$x_e + tx_d = x_a + \beta(x_b - x_a) + \gamma(x_c - x_a)$$
$$y_e + ty_d = y_a + \beta(y_b - y_a) + \gamma(y_c - y_a)$$
$$z_e + tz_d = z_a + \beta(z_b - z_a) + \gamma(z_c - z_a)$$

- Textbook shows how to use Cramer's rule to solve this system of linear equations.
- If $\beta > 0$, $\gamma > 0$ and $\beta + \gamma < 1$ then the intersect is inside of the triangle.

More efficient algorithm to calculate if the ray hits the triangle:

Algorithm 4 Early termination ray-triangle intersection algorithm

Require: Ray r, Vector a, Vector b, Vector c, t_0, t_1
Ensure: If the ray hits the triangle

```
Compute t
if (t < t_0) or (t > t_1) then
    return false
end if
Compute γ
if (γ < 0) or (γ > 1) then
    return false
end if
Compute β
if (β < 0) or (β + γ > 1) then
    return false
end if
return true
```

Remainder of Chapter 4.

- Sections 4.4.3 and 4.4.4 are left out, but can be useful in practical 4.
- Section 4.5 will be covered in the next lecture as part of Chapter 5's discussion.

Joke of the day - By ChatGPT

Why did the ray tracing algorithm always avoid blind dates?

Joke of the day - By ChatGPT

Why did the ray tracing algorithm always avoid blind dates?

Because it was tired of intersecting with disappointment!