Previously
Shadows
Shadows

• Three distinct parts of a shadow:
Shadows

• Three distinct parts of a shadow:
  • **Umbra** – full shadow
Shadows

• Three distinct parts of a shadow:
  • Umbra – full shadow
  • Penumbra – half shadow
Shadows

- Three distinct parts of a shadow:
  - **Umbra** – full shadow
  - **Penumbra** – half shadow
  - **Antumbra** – after shadow
Three distinct parts of a shadow:

- **Umbra** – full shadow
- **Penumbra** – half shadow
- **Antumbra** – after shadow

What happens with point or directional light sources?
Ray Traced Shadows

- With path or ray tracing, shoot rays (shadow rays) to points on the light source.
Ray Traced Shadows

- With path or ray tracing, shoot rays (shadow rays) to points on the light source.

- When the rays do not hit the light source, the point is in a shadow (rays hit the occluder).
Shadow Mapping

• Goal is the same: find an occluder between a surface point and the light source.
Shadow Mapping

- Goal is the same: find an occluder between a surface point and the light source.

- **Render** the depth buffer **from the light source**.

Store it in a texture.
Shadow Mapping

• Goal is the same: find an occluder between a surface point and the light source.

• Render the depth buffer from the light source.

• In the main rendering: for each fragment consider the depth to the light source.
Shadow Mapping

- Goal is the same: find an occluder between a surface point and the light source.
- Render the depth buffer from the light source.
- For each fragment in the main rendering consider the depth to the light source.

How to find the depth to the light source?
- We need the actual depth (not the distance).
• What happens if it is **larger then the closest depth** seen from the light source?

The one we rendered from the light source and stored in a texture.
Shadow Mapping

Directional Light Source

Near plane

Camera

Far plane

F_2

1.98

F_1

3.9

F_0

3.77

F_3

Directional Light
Shadow Mapping

What problems do you see here?
Shadow Mapping

- See the example in CGLearn
Shadow Volume

- We create a volume (mesh) around the shadow of each object.
Shadow Volume

- We create a volume (mesh) around the shadow of each object.

- When rendering a fragment, we count the times we enter a volume, minus the times we exit a volume.
Shadow Volume

- We create a volume (mesh) around the shadow of each object.
- When rendering a fragment, we count the times we enter a volume, minus the times we exit a volume.
- Multiple ways to get this info:
  - Depth pass – counts from front
    Shadow volume faces between the object and the camera
Shadow Volume

• We create a volume (mesh) around the shadow of each object.
• When rendering a fragment, we count the times we enter a volume, minus the times we exit a volume.
• Multiple ways to get this info:
  • Depth pass – counts from front
    Shadow volume faces between the object and the camera
  • Depth fail – counts from back (Carmack's reverse)
    Shadow volume faces after the object to infinity
Shadow Volume

- We create a volume (mesh) around the shadow of each object.
- When rendering a fragment, we count the times we enter a volume, minus the times we exit a volume.
- Multiple ways to get this info:
  - Depth pass – counts from front
    Shadow volume faces between the object and the camera
  - Depth fail – counts from back (Carmack's reverse)
    Shadow volume faces after the object to infinity
- When the count (sum) is 0, the object is lit.
1. Shadow Volume

- Find the silhouette of the object.
1. Shadow Volume

- Find the silhouette of the object.
- How to find those edges?
2. Shadow Volume

- Copy the vertices and extrude the copy to infinity from the light source.
2. Shadow Volume

- Copy the vertices and extrude the copy to infinity from the light source.

- Assigning 0 as the fourth coordinate, results in a point projected to infinity.
3. Shadow Volume

- Create **caps and sides** of the shadow volume.
3. **Shadow Volume**

- Create caps and sides of the shadow volume.
- In order for the far plane not to clip the volume, send the far plane to infinity.

\[
\lim_{\text{far} \to \infty} P = \begin{pmatrix}
\frac{1}{ar \cdot \tan(\alpha)} & 0 & 0 & 0 \\
0 & \frac{1}{\tan(\alpha)} & 0 & 0 \\
0 & 0 & -1 & -2 \cdot \text{near} \\
0 & 0 & -1 & 0
\end{pmatrix}
\]
4. Shadow Volume

- **Depth-pass** – count the front facing shadow planes minus the back facing shadow planes *in front* of the fragment.

One of those has a problem.
Shadow Volume

- **Depth-fail** – count the front facing shadow planes minus the back facing shadow planes **behind** the fragment.
Shadow Algorithms

- **Projection Shadow** – In the Basic I bonus task. One of the early shadow algorithms.
- **Shadow Volume** – In Doom 3.
- **Shadow Mapping** – Used today in game engines. Many advancements (*eg* cascaded shadow mapping).
- **Ray traced shadows** – When you include a ray trace based rendering solution, you can cast shadow rays.
Conclusion
Conclusion

- Coordinate Systems
  - Left / Right Handed
- Triangles
  - Planar
- Polygons
  - Convex / Concave
  - Simple
Conclusion

- Convex Combination
- Barycentric Coordinates
- Points and Vectors
Conclusion

- Standard Graphics Pipeline
- Application code
- GPU steps
Conclusion

- Linear Transformations
  - Rotation
  - Scale
  - Shear
- Affine Transformations
  - Translation
- Homogeneous Coordinates

\[
\begin{pmatrix}
\cos(\alpha) & -\sin(\alpha) \\
\sin(\alpha) & \cos(\alpha)
\end{pmatrix}
\]

\[
\begin{pmatrix}
a_x & 0 \\
0 & a_y
\end{pmatrix}
\]

\[
\begin{pmatrix}
1 & 0 \\
\tan(\varphi) & 1
\end{pmatrix}
\]

\[
\begin{pmatrix}
1 & 0 & x_t \\
0 & 1 & y_t
\end{pmatrix}
\]

\[
\begin{pmatrix}
1 & 0 & x_t \\
0 & 1 & y_t
\end{pmatrix}
\]

\[
\begin{pmatrix}
1 & 0 & x_t \\
0 & 1 & y_t
\end{pmatrix}
\]

\[
\begin{pmatrix}
w & w & w & w
\end{pmatrix}
\]
Conclusion

- Augmented Transformation Matrix

\[
\begin{pmatrix}
a & b & c & x_t \\
d & e & f & y_t \\
g & h & i & z_t \\
0 & 0 & 0 & 1
\end{pmatrix}
\begin{pmatrix}
x \\
y \\
z \\
1
\end{pmatrix} =
\begin{pmatrix}
ax + by + cz + x_t \\
dx + ey + fz + y_t \\
gx + hy + iz + z_t \\
1
\end{pmatrix}
\]

Used for perspective projection...
Conclusion

- Projection
  - Orthographic
  - Oblique
  - Perspective
Conclusion

- Frames of Reference
  - Object Space
  - World Space
  - View Space
  - Clip Space
  - Normalized Device Space
  - Screen Space
Conclusion

- Shading Models
  - Flat
  - Gouraud
  - Phong

- Lighting Models
  - Lambert
  - Phong
  - Blinn-Phong

- sRGB Color Space and Gamma
Conclusion

- Textures
- UV Mapping
- Mipmap
- Interpolation
  - Upscale
  - Downscale
- Anisotropy
Conclusion

• Blending
  • Alpha
    – Conventional
    – Premultiplied
  • General Blending Function
    – Alpha Blending
    – Additive
    – Multiplicative

blend \((src, dst) = (src \cdot srcFactor) \text{ function} (dst \cdot dstFactor)\)
Conclusion

- Asset Pipeline
- Code vs Design vs Art
- Vertical Slice
- Game Engine
  - Runtime
  - Tools
  - Layered Architecture
- Drivers, OS, Resource Manager, Middleware, SDK

Image by Ats Kurvet and Timo Kallaste
Conclusion

- Curves
  - Interpolating vs Approximating
  - Smoothness $C^n$ and $G^n$
  - Construction with constraints
- Hermite
- Catmull-Rom
- Bezier
Conclusion

- Procedural Generation
  - Noise
    - Value noise
    - Gradient noise (Perlin)
- Lindenmayer Systems
- Particle Systems
  - Boids
Conclusion

- Ray Casting
  - Ray-Triangle Intersection
- Ray Tracing
  - Ray Trace Rendering
- Data Structures
  - Octree
  - K-D Tree
  - Binary Space Partitioning
  - Bounding Volume Hierarchy
Conclusion

- Global Illumination
- Path Tracing
- Photon Mapping
- Radiosity

- The Rendering Equation

\[ L_{\text{emit}}(x, \omega_o) + \int_{\Omega} f_{\text{brdf}}(x, \omega_i, \omega_o) \cdot L_i(x, \omega_i) \cdot (\omega_i \cdot n) \, d\omega_i \]
Conclusion

- Shadows
  - Shadow Rays
  - Shadow Mapping
  - Shadow Volume
- Umbra, penumbra, antumbra
Conclusion

- Conclusion
  - Coordinate System Handedness
  - Polygons
    - Convex and Concave
    - Simple
  - Triangles
    - Planar
  - Barycentric coordinates
-...

...
Computer Graphics

Thanks for the ride!

The End
What knowledge did you gain today?

What more would you like to know?

Next time: Your open mic lecture!

If you can not present your project in the projects presentation. Bonus +2 points.