CSC418: Computer Graphics
DAVID LEVIN
Today’s Topics

1. Texture mapping
2. More Ray Tracing

Some slides and figures courtesy of Wolfgang Hürst, Patricio Simari
But First ... Logistical Things

• Assignment 3 available on BBS (coming soon to website)
Topic 1:

Texture Mapping

- Motivation
- Sources of texture
- Texture coordinates
- \{Bump, MIP, displacement, environmental\} mapping
Motivation

• Adding lots of detail to our models to realistically depict skin, grass, bark, stone, etc., would increase rendering times dramatically, even for hardware-supported projective methods.
Motivation

- Adding lots of detail to our models to realistically depict skin, grass, bark, stone, etc., would increase rendering times dramatically, even for hardware-supported projective methods.
Motivation

Basic idea of **texture mapping**:

Instead of calculating color, shade, light, etc. for each pixel we just paste images to our objects in order to create the illusion of realism.

Different approaches exist (e.g. tiling; cf. previous slide)
Motivation

In general, we distinguish between 2D and 3D texture mapping:

2D mapping (aka *image textures*): paste an image onto the object

3D mapping (aka *solid* or *volume textures*): create a 3D texture and "carve" the object

2D texture $\leftrightarrow$ 3D texture
Topic 1:

Texture Mapping

- Motivation
- Sources of texture
- Texture coordinates
- \{Bump, MIP, displacement, environmental\} mapping
Texture sources: Photographs
Texture sources: Solid textures
Texture sources: Procedural
Texture sources: Synthesized

Kwatra et al., SIGGRAPH '05
Topic 1:

Texture Mapping

• Motivation
• Sources of texture
• Texture coordinates
• {Bump, MIP, displacement, environmental} mapping
Texture coordinates

How does one establish correspondence? (UV mapping)

For each triangle in the model, establish a corresponding region in the phototexture.

During rasterization, interpolate the coordinate indices into the texture map.
Texture coordinates

Example: use world map and sphere to create a globe

Per conventions we usually assume $u, v \in [0, 1]$. 
Texture coordinates

\[
\begin{align*}
  x &= x_c + r \cos \phi \sin \theta \\
  y &= y_c + r \sin \phi \sin \theta \\
  z &= z_c + r \cos \theta
\end{align*}
\]

Given a point \((x, y, z)\) on the surface of the sphere, we can find \(\theta\) and \(\phi\) by

\[
\begin{align*}
  \theta &= \arccos \frac{z-z_c}{r} \quad (\text{cf. longitude}) \\
  \phi &= \arctan \frac{y-y_c}{x-x_c} \quad (\text{cf. latitude})
\end{align*}
\]

(Note: \(\arccos\) is the inverse of \(\cos\), \(\arctan\) is the inverse of \(\tan = \frac{\sin}{\cos}\))
Texture coordinates

For a point \((x, y, z)\) we have

\[
\theta = \arccos \frac{z - z_c}{r} \\
\phi = \arctan \frac{y - y_c}{x - x_c}
\]

\((\theta, \phi) \in [0, \pi] \times [-\pi, \pi]\), and \(u, v\) must range from \([0, 1]\).

Hence, we get:

\[
\begin{align*}
u &= \frac{\phi \mod 2\pi}{2\pi} \\
v &= \frac{\pi - \theta}{\pi}
\end{align*}
\]

(Note that this is a simple scaling transformation in 2D)
Texture coordinates

Example: “Tiling” of 2D textures into a $UV$-object space

We’ll call the two dimensions to be mapped $u$ and $v$, and assume an $n_x \times n_y$ image as texture.

Then every $(u, v)$ needs to be mapped to a color in the image, i.e. we need a mapping from pixels to texels.
A standard way is to first remove the integer portion of $u$ and $v$, so that $(u, v)$ lies in the unit square.
Texture coordinates

This results in a simple mapping from $0 \leq u, v \leq 1$ to the size of the texture array, i.e. $n_x \times n_y$.

$$i = un_x \text{ and } j = vn_y$$

Yet, for the array lookup, we need integer values.
The texel \((i, j)\) in the \(n_x \times n_y\) image for \((u, v)\) can be determined using the floor function \([x]\) which returns the highest integer value \(\leq x\).

\[
i = \lfloor un_x \rfloor \quad \text{and} \quad j = \lfloor vn_y \rfloor
\]
Texture coordinates

\[ c(u, v) = c_{i,j} \text{ with } i = \lfloor un_x \rfloor \text{ and } j = \lfloor vn_y \rfloor \]

This is a version of nearest-neighbor interpolation, where we take the color of the nearest neighbor.
For smoother effects we may use **bilinear interpolation**:

\[
c(u, v) = (1-u')(1-v')c_{ij} + u'(1-v')c_{(i+1)j} + (1-u')v'c_{i(j+1)} + u'v'c_{(i+1)(j+1)}
\]

with

\[
u' = un_x - \lfloor un_x \rfloor \quad \text{and} \quad v' = vn_y - \lfloor vn_y \rfloor
\]

Notice that all weights are between 0 and 1 and add up to 1:

\[
(1 - u')(1 - v') + u'(1 - v') + (1 - u')v' + u'v' = 1
\]
Topic 1:

Texture Mapping

- Motivation
- Sources of texture
- Texture coordinates
  - {Bump, MIP, displacement, environmental} mapping
Mipmapping

aliasing
MIP-Mapping: Basic Idea

Given a polygon, use the texture image, where the projected polygon best matches the size of the polygon on screen.
Mipmapping

Solutions: MIP maps

- Pre-calculated, optimized collections of images based on the original texture
- Dynamically chosen based on depth of object (relative to viewer)
- Supported by today's hardware and APIs
Mipmapping
Environment mapping

... why not use this to make objects appear to reflect their surroundings specularly?

Idea: place a cube around the object, and project the environment of the object onto the planes of the cube in a preprocessing stage; this is our texture map.

During rendering, we compute a reflection vector, and use that to look-up texture values from the cubic texture map.
Environment mapping
Environment mapping

Remember Phong shading: “perfect” reflection if

angle between eye vector $\vec{e}$ and $\vec{n} = $ angle between $\vec{n}$ and reflection vector $\vec{r}$
Environment mapping
One of the reasons why we apply texture mapping:

Real surfaces are hardly flat but often rough and bumpy. These bumps cause (slightly) different reflections of the light.
Bump mapping

Instead of mapping an image or noise onto an object, we can also apply a bump map, which is a 2D or 3D array of vectors. These vectors are added to the normals at the points for which we do shading calculations.

The effect of bump mapping is an apparent change of the geometry of the object.
Bump mapping

Major problems with bump mapping: silhouettes and shadows
2D Image Bump Mapping Using a 24-bit Bitmap
To overcome this shortcoming, we can use a displacement map. This is also a 2D or 3D array of vectors, but here the points to be shaded are actually displaced.

Normally, the objects are refined using the displacement map, giving an increase in storage requirements.
Displacement mapping
Bounce Maps
Topic 2:

Basic Ray Tracing

• Introduction to ray tracing
• Computing rays
• Computing intersections
  • ray-triangle
  • ray-polygon
  • ray-quadric
  • the scene signature
• Computing normals
• Evaluating shading model
• Spawning rays
• Incorporating transmission
  • refraction
  • ray-spawning & refraction
A basic ray tracing algorithm

FOR each pixel DO
  • compute viewing ray
  • find the 1st object hit by the ray and its surface normal \( \vec{n} \)
  • set pixel color to value computed from hit point, light, and \( \vec{n} \)
Shading model

Remember our shading model:

\[ c = c_r (c_a + c_l \max(0, n \cdot l)) + c_l (\hat{h} \cdot \hat{n})^p \]

with

- Ambient shading
- Lambertian shading
- Phong shading

and Gouraud interpolation.
Topic 3:

Less Basic Ray Tracing

- Introduction to ray tracing
- Computing rays
- Computing intersections
  - ray-triangle
  - ray-polygon
  - ray-quadric
  - the scene signature
- Computing normals
- Evaluating shading model
- Spawning rays
- Incorporating transmission
  - refraction
  - ray-spawning & refraction
local illumination  reflection  refraction
Modeling Reflection: Transmission

Transmission:
- Caused by materials that are not perfectly opaque
- Examples include glass, water and translucent materials such as skin
Physics of Refraction

Physics: the speed of light depends on the material through which it travels (and the wavelength of light, but we will ignore that)

![Diagram of light refraction](image)

Refraction (bending of rays) occurs when light crosses an interface between two media with different speeds of light.
Physics of Refraction

Snell's law
\[
\frac{\sin \theta_1}{\sin \theta_2} = \frac{C_1}{C_2}
\]

Ratio called the relative index of refraction

Refraction (bending of rays) occurs when light crosses an interface between two media with different speeds of light.
Geometry of Refraction

Snell's law
\[
\frac{\sin \theta_1}{\sin \theta_2} = \frac{c_1}{c_2}
\]

3. If \( c_2 < c_1 \), light bends toward the normal.
If \( c_2 > c_1 \), light bends away from normal.

(interface)

(light source)
Geometry of Refraction: Transmission Vector

Snell's law
\[ \frac{\sin \theta_1}{\sin \theta_2} = \frac{c_1}{c_2} \]

1. Incident ray, outgoing ray & normal always lie on the same plane \( \Rightarrow \)
   \[ \vec{d} \text{ along } -\frac{c_2}{c_1} \vec{s} + \left[ \frac{c_2}{c_1} \cos \theta_1 - \cos \theta_2 \right] \vec{n} \]
Assumption: Refracted ray lies in the **same plane** as the incident ray

\[ \vec{S} = \vec{S}_\perp + \vec{S}_\parallel \]

- \( \vec{S}_\perp \): Perpendicular component
- \( \vec{S}_\parallel \): Parallel component
\[
\vec{s}_\perp = (\vec{s} \cdot \vec{n}) \vec{n}
\]
\[
\vec{s}_\parallel = \vec{s} - (\vec{s} \cdot \vec{n}) \vec{n}
\]
\[
\sin \theta_1 = \frac{\| \vec{s}_\parallel \|}{\| \vec{s} \|}
\]
\[
\sin \theta_2 = \frac{\| \vec{t}_\parallel \|}{\| \vec{t} \|}
\]
\[
\| \vec{s}_\parallel \| = \frac{c_1}{c_2} \| \vec{t}_\parallel \|
\]
\[
\| \vec{s}_\parallel \| = \frac{c_1}{c_2} \| \vec{t}_\parallel \|
\]

\[
\vec{s}_\parallel = \vec{s} - (\vec{s} \cdot \hat{n}) \hat{n}
\]

\[
\vec{t}_\parallel = -\frac{c_2}{c_1} (\vec{s} - \cos \theta_1) \hat{n}
\]

\[
\vec{t}_\perp = -\| \vec{t}_\perp \| \hat{n}
\]

\[
\vec{t}_\perp = -\sqrt{1 - \left( \frac{c_2}{c_1} \right)^2 (1 - \cos^2 \theta_1)} \cdot \hat{n}
\]

\[
\vec{t} = \vec{t}_\perp + \vec{t}_\parallel
\]
\[ \vec{t} = \vec{t}_\perp + \vec{t}_\parallel \]

\[ \vec{t} = -\frac{c_2}{c_1} \vec{s} + \left( \frac{c_2}{c_1} \cos \theta_1 - \cos \theta_2 \right) \vec{n} \]

Q1: Define all the terms in this equation?
Q2: Which terms are known and unknown during ray tracing?
Q3: How do you compute the unknown terms?
Geometry of Refraction: The Critical Angle

Snell's law

\[
\frac{\sin \theta_1}{\sin \theta_2} = \frac{c_1}{c_2}
\]

If \( c_2 > c_1 \), there is a critical angle above which no transmission occurs (\( \Rightarrow \) have total internal reflection)
Total Internal Reflection

Snell's law
\[
\frac{\sin \theta_1}{\sin \theta_2} = \frac{C_1}{C_2}
\]

4. Deriving the critical angle: From Snell's law,
\[
\cos \theta_2 = \sqrt{1 - \left(\frac{C_2}{C_1}\right)^2 \sin^2 \theta_1}
\]
At critical angle, \( \theta_2 = \frac{\pi}{2} \) \( \Rightarrow \) \( \sin \theta_1^* = \frac{C_1}{C_2} \)
Total Internal Reflection

Snell's law
\[ \frac{\sin \theta_1}{\sin \theta_2} = \frac{c_1}{c_2} \]

\[ 0^\circ \text{ for } \theta_1 > \theta_1^* \text{, } \theta_2 \text{ is undefined} \]
Geometry of Refraction: Normal Incidence

Snell's law
\[ \frac{\sin \theta_1}{\sin \theta_2} = \frac{c_1}{c_2} \]

If \( \theta_1 = 0 \), no bending occurs

\[ \vec{d} \text{ along } -\frac{c_2}{c_1} \vec{s} + \left[ \frac{c_2}{c_1} \cos \theta_1 - \cos \theta_2 \right] \vec{n} \]
Topic 12:

Less Basic Ray Tracing

- Introduction to ray tracing
- Computing rays
- Computing intersections
  - ray-triangle
  - ray-polygon
  - ray-quadric
  - the scene signature
- Computing normals
- Evaluating shading model
- Spawning rays
- Incorporating transmission
  - refraction
  - ray-spawning & refraction
Ray Spawning

Ray Spawning: The Ray Tree

Primary ray

Depth 0

Depth 1

Depth 2

Depth 3

© www.scratchapixel.com
No reflection
Single reflection
Double reflection
Topic 12: Less Basic Ray Tracing

- Introduction to ray tracing
- Computing rays
- Computing intersections
  - ray-triangle
  - ray-polygon
  - ray-quadric
  - the scene signature
- Computing normals
- Evaluating shading model
- Spawning rays
- Incorporating transmission
  - refraction
  - ray-spawning & refraction
  - Improvements
Ray Tracing Improvements: Caustics
Ray Tracing Improvements: Caustics

Reverse Direction Ray Tracing

• Trace from the light to the surfaces and then from the eye to the surfaces
• “shower” scene with light and then collect it
• “Where does light go?” vs “Where does light come from?”
• Good for caustics
• Transport  $E \rightarrow S \rightarrow S \rightarrow D \rightarrow S \rightarrow S \rightarrow L$
Ray Tracing Improvements: Image Quality

Cone tracing
  • Models some dispersion effects

Distributed Ray Tracing
  • Super sample each ray
  • Blurred reflections, refractions
  • Soft shadows
  • Depth of field
  • Motion blur

Stochastic Ray Tracing
How many rays do you need?

1 ray/light  
10 ray/light  
20 ray/light  
50 ray/light

Antialiasing – Supersampling

Point light:
- Jaggies
- W/ antialiasing

Area light:
- Jaggies
- W/ antialiasing
Radiosity

- Diffuse interaction within a closed environment
- Theoretically sound
- View independent
- No specular interactions
- Color bleeding visual effects
- Transport $E - D - D - D - L$
Topic 13:

Instancing
**Copying and transforming objects**

**Instancing** is an elegant technique to place various transformed copies of an object in a scene.

*Expl.: circle → ellipse*

1. **scale**
2. **rotate**
3. **move**
**Copying and transforming objects**

**Instancing** is an elegant technique to place various transformed copies of an object in a scene.

Expl.: circle $\rightarrow$ ellipse

1. **scale**
2. **rotate**
3. **move**
Copying and transforming objects

Instead of making actual copies, we simply store a reference to a base object, together with a transformation matrix.

That can save us lots of storage.

Hmm, but how do we compute the intersection of a ray with a randomly rotated ellipse?
Ray-instance intersection

Assume an object \( O \) that is used to create an object \( MO \) via instancing.
Ray-instance intersection

Now, we want to create the intersection of $MO$ with the ray $\mathbf{r}(t)$, which in turn is defined by the line

$$\mathbf{l}(t) = \mathbf{e} + t \mathbf{d}.$$
Fortunately, such complicated intersection tests (e.g. ray/ellipsoid) can often be replaced by much simpler tests (e.g. ray/sphere).

\[ \vec{l}(t) = \vec{e} + t \vec{d} \]
To determine the intersections $p_i$ of a ray $\vec{r}$ with the instance $MO$, we first compute the intersections $p_i'$ of the inverse transformed ray $M^{-1}\vec{r}$ and the original object $O$.

$$l(t) = \vec{e} + t \vec{d}$$
The points $\vec{p}_i$ are then simply

$$M\vec{p}_i'$$

or $\vec{l}(t_i')$ because the linear transformation preserves relative distances along the line.

$$\vec{l}(t) = \vec{e} + t \vec{d}$$
Ray-instance intersection

Two pitfalls:
- The direction vector of the ray should *not* be normalized
- Surface normals transform differently!
  → use \((M^{-1})^T\) instead of \(M\) for normals

\[ t(t) = \vec{e} + t \vec{d} \]
Unfortunately, normal vectors are not always transformed properly.

E.g. look at shearing, where tangent vectors are correctly transformed but normal vectors not.

To transform a normal vector $\vec{n}$ correctly under a given linear transformation $A$, we have to apply the matrix $(A^{-1})^T$. Why?
Transforming normal vectors

We know that tangent vectors are transformed correctly: $A\vec{t} = \vec{t}_A$.
But this is not necessarily true for normal vectors: $A\vec{n} \neq \vec{n}_A$.

Goal: find matrix $N_A$ that transforms $\vec{n}$ correctly, i.e. $N_A\vec{n} = \vec{n}_N$ where $\vec{n}_N$ is the correct normal vector of the transformed surface.

Because our original normal vector $\vec{n}^T$ is perpendicular to the original tangent vector $\vec{t}$, we know that:

$\vec{n}^T \vec{t} = 0$.

This is the same as

$\vec{n}^T I \vec{t} = 0$

which is is the same as

$\vec{n}^T A^{-1} A\vec{t} = 0$
Transforming normal vectors

Because $A\vec{t} = \vec{t}_A$ is our correctly transformed tangent vector, we have

$$\vec{n}^T A^{-1} \vec{t}_A = 0$$

Because their scalar product is 0, $\vec{n}^T A^{-1}$ must be orthogonal to it. So, the vector we are looking for must be

$$\vec{n}_N^T = \vec{n}^T A^{-1}.$$ 

Because of how matrix multiplication is defined, this is a transposed vector. But we can rewrite this to

$$\vec{n}_N = (\vec{n}^T A^{-1})^T.$$ 

And if you remember that $(AB)^T = B^T A^T$, we get

$$\vec{n}_N = (A^{-1})^T \vec{n}.$$