constituents of 3d scene

3d graphics
- Modeling
- Rendering

modeling

rendering

3d scene

background: spaces
- linear space
  - scalar
  - vectors
- affine space
  - scalar
  - vector
  - points
**Scalars:** Real numbers

- 3.8
- 2.7
- 4.1
- -1000.2
- 5

**Point:** Position in (3d) space

- (4.5, -1)

**Vector:** Distance & direction in (3d) space

- \( v = \langle q_x - p_x, q_y - p_y, q_z - p_z \rangle \)

**Vector:** Magnitude & direction in (3d) space

- A vector does not have a position in space!

**Vector:** Notation

- Abusive but useful notation: \( v = q - p \)
vector: distance & direction in (3d) space

abusive but useful notation \( v = q - p \)

... abusive because addition of points is not defined and, even if it was, we'd expect the result to be a point, not a vector.

\[ \begin{align*}
  p &= (p_x, p_y, p_z) \\
  v &= (v_x, v_y, v_z) \\
  q &= (p_x + v_x, p_y + v_y, p_z + v_z)
\end{align*} \]

points vs. vectors

• \((x, y, z)\) is a point in space
• \((x, y, z)\) is the distance/direction you have to travel from the origin to get to the point \((x, y, z)\)

sometimes it is convenient to blur the distinction

vector operating on a point

abusive but useful notation \( q = p + v \)

this notation makes perfect sense if we think of \( q \) and \( p \) as vectors.

notation makes sense if \( p \) and \( q \) are vectors.
points vs. vectors
- \((x,y,z)\) is a point in space
- \(\langle x,y,z \rangle\) is the distance/direction you have to travel from the origin to get to the point \((x,y,z)\)

sometimes it is convenient to blur the distinction ... sometimes not...

vector operations
- vector norm
- scalar multiplication
- vector addition
- dot product
- cross product (for 3d)
- linear transforms

magnitude, length, euclidean norm
- \(v = \langle vx, vy, vz \rangle\)
- \(|v| = (v_x^2 + v_y^2 + v_z^2)^{\frac{1}{2}}\)

\(v\) is a unit vector if \(|v| = 1\)

scalar multiplication
- \(v = \langle vx, vy, vz \rangle\)
- \(\alpha v = \langle \alpha vx, \alpha vy, \alpha vz \rangle\)

vector addition
- \(u = \langle ux, uy, uz \rangle\)
- \(v = \langle vx, vy, vz \rangle\)
- \(u + v = \langle ux + vx, uy + vy, uz + vz \rangle\)

dot (inner) product
- \(\mathbf{u} = \langle u_x, u_y, u_z \rangle\)
- \(\mathbf{v} = \langle v_x, v_y, v_z \rangle\)
- \(\mathbf{u} \cdot \mathbf{v} = u_x v_x + u_y v_y + u_z v_z = |\mathbf{u}| |\mathbf{v}| \cos \phi\)

note: \(\mathbf{u} \cdot \mathbf{v} = |\mathbf{v}|^2\)
orthogonal vectors

- non-zero vectors $\mathbf{u}$ and $\mathbf{v}$ are orthogonal if $\mathbf{u} \cdot \mathbf{v} = 0$

cross product (3d)

$\mathbf{u} \times \mathbf{v} = \mathbf{w}$

- magnitude: $||\mathbf{w}|| = ||\mathbf{u}|| \cdot ||\mathbf{v}|| \sin \phi$
- direction: $\mathbf{w}$ is orthogonal to both $\mathbf{u}$ and $\mathbf{v}$ in direction defined by right hand rule

cross product (3d)

- $\mathbf{u} = \langle u_x, u_y, u_z \rangle$
- $\mathbf{v} = \langle v_x, v_y, v_z \rangle$

$\mathbf{w} = \begin{vmatrix} i & j & k \\ u_x & u_y & u_z \\ v_x & v_y & v_z \end{vmatrix}$

linear transform

$f : \mathbb{R}^n \rightarrow \mathbb{R}^n$ is a linear transform if for any scalars $\alpha, \beta$ and any vectors $\mathbf{u}, \mathbf{v}$, $f$ satisfies the following:

$$f(\alpha \mathbf{u} + \beta \mathbf{v}) = \alpha f(\mathbf{u}) + \beta f(\mathbf{v})$$

Linear transforms are the functions that can be expressed as matrix multiplication!!

Linear transform

$$\begin{bmatrix} s & 0 & 0 \\ 0 & t & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ w \end{bmatrix} = \begin{bmatrix} sx \\ ty \\ w \end{bmatrix}$$

3d scene

- lights
- graphics primitives
- "world" coordinate system
- view (eye/camera)
overview

• models
  - simple shapes (for now)
  - transforms
• lights and material properties
• view
  - viewpoint
  - projection type/view volume
• illumination models
  - local
  - global

primitives

simple shapes
  - triangle
  - sphere
  later in the course we'll cover modeling in greater depth

triangle

• defined by three vertices
• vertex order (counterclockwise) defines "front"
• necessarily planar

triangle mesh

sphere

• specified by center and radius

overview

• models
  - simple primitives (for now)
  - transforms
• lights and material properties
• frustum
  - Projection
• illumination models
  - local
  - global
transforms

• scale
• rotate
• translate

let's start with 2D

scale

rotate

translate

composite transforms
computing scaled coordinates

\[
\begin{pmatrix}
  s & 0 \\
  0 & t \\
\end{pmatrix}
\begin{pmatrix}
  x \\
  y \\
\end{pmatrix}
= 
\begin{pmatrix}
  sx \\
  ty \\
\end{pmatrix}
\]

scale

again we are blurring the distinction of a point and vector

computing rotated coordinates

\[
\begin{pmatrix}
  \cos \phi & -\sin \phi \\
  \sin \phi & \cos \phi \\
\end{pmatrix}
\begin{pmatrix}
  x \\
  y \\
\end{pmatrix}
= 
\begin{pmatrix}
  x \cos \phi - y \sin \phi \\
  x \sin \phi + y \cos \phi \\
\end{pmatrix}
\]

computing translated coordinates

\[
\begin{pmatrix}
  x \\
  y \\
\end{pmatrix}
= 
\begin{pmatrix}
  x + x_0 \\
  y + y_0 \\
\end{pmatrix}
\]

linear transformation

- \( f(\mathbf{v}) \) is linear if \( f(\alpha \mathbf{u} + \beta \mathbf{v}) = \alpha f(\mathbf{u}) + \beta f(\mathbf{v}) \)
- translation is not a linear transform: define \( T(\mathbf{v}) = \mathbf{v} + \mathbf{w}_0 \) where \( \mathbf{w}_0 \) is a non-zero vector
  \[
  T(\mathbf{u} + \mathbf{v}) = \mathbf{u} + \mathbf{v} + \mathbf{w}_0 \\
  T(\mathbf{u}) + T(\mathbf{v}) = \mathbf{u} + \mathbf{v} + 2\mathbf{w}_0
  \]

transform composition

\[
\begin{pmatrix}
  M_T & M_R & M_S \\
\end{pmatrix}
\begin{pmatrix}
  \mathbf{p} \\
\end{pmatrix}
= 
\begin{pmatrix}
  \mathbf{p}' \\
\end{pmatrix}
\]

it would be incredibly convenient if we could compose transforms using matrix multiplication
transform polygon mesh

1,000,000 vertices
10 transforms
10,000,000 computations

transform polygon mesh

1,000,000 vertices
10 transforms
10,000,000 computations
1 composite transform

transform composition

\[ M_T M_R M_S p = p' \]

it would be incredibly convenient if we could compose transforms using matrix multiplication

a little trick ...

\[
\begin{pmatrix}
1 & 0 & x_0 \\
0 & 1 & y_0 \\
0 & 0 & 1
\end{pmatrix}
\begin{pmatrix}
x \\
y \\
1
\end{pmatrix}
= \begin{pmatrix}
x + x_0 \\
y + y_0 \\
1
\end{pmatrix}
\]

(x,y) \rightarrow (x+x_0,y+y_0,1)

homogenous coordinates

\[
(x,y) \leftrightarrow (x,y,1)
\]
scale

\[
\begin{bmatrix}
  s & 0 & 0 \\
  0 & t & 0 \\
  0 & 0 & 1
\end{bmatrix}
\begin{bmatrix}
  x \\
  y \\
  1
\end{bmatrix} =
\begin{bmatrix}
  sx \\
  ty \\
  1
\end{bmatrix}
\]

rotate

\[
\begin{bmatrix}
  \cos \phi & -\sin \phi & 0 \\
  \sin \phi & \cos \phi & 0 \\
  0 & 0 & 1
\end{bmatrix}
\begin{bmatrix}
  x \\
  y \\
  1
\end{bmatrix} =
\begin{bmatrix}
  x \cos \phi - y \sin \phi \\
  x \sin \phi + y \cos \phi \\
  1
\end{bmatrix}
\]

translate

\[
\begin{bmatrix}
  s & 0 & 0 \\
  0 & t & 0 \\
  0 & 0 & 1
\end{bmatrix}
\begin{bmatrix}
  x \\
  y \\
  1
\end{bmatrix} =
\begin{bmatrix}
  sx \\
  ty \\
  1
\end{bmatrix}
\]

transform form

\[
\begin{bmatrix}
  ? & ? & ? \\
  ? & ? & ? \\
  0 & 0 & 1
\end{bmatrix}
\begin{bmatrix}
  x \\
  y \\
  1
\end{bmatrix} =
\begin{bmatrix}
  x' \\
  y' \\
  1
\end{bmatrix}
\]

we are not alone...

the parallel universe view of homogenous coordinates

we live in this universe

we are not alone...

the parallel universe view of homogenous coordinates

we live in this universe

it's not the only one, but it is the only one we can experience!
and its better not to think about it …

the parallel universe view of homogenous coordinates

and its better not to think about it …

the parallel universe view of homogenous coordinates

our universe has center (0,0)

2d and 2d homogenous

our universe

our universe when it comes to computing modeling transforms

3d and 3d homogenous

our universe

our universe when it comes to computing modeling transforms

rotate about z axis

rotate about x & y axes are similar

scale

translate
transform form

\[
\begin{pmatrix}
? & ? & ? & \mathbf{x} \\
? & ? & ? & \mathbf{y} \\
2 & 2 & 2 & \mathbf{z} \\
0 & 0 & 1 & 1
\end{pmatrix}
= 
\begin{pmatrix}
\mathbf{x}' \\
\mathbf{y}' \\
\mathbf{z}' \\
1
\end{pmatrix}
\]

3d and 3d homogenous

our universe

our universe when it comes to computing modeling transforms

overview

- models
  - simple primitives (for now)
  - transforms
- lights and material properties
- camera
  - frustum
  - projection
- illumination models
  - local
  - global

3d scene

models

lights

view (eye/camera)

objective

approximate the effects of light/materials as we perceive them in a computationally efficient way

light sources (in cs155)

- ambient light
- point light
- spot light
- directional light
ambient light
- "source-less" light that arrives uniformly from all directions at all points in the scene
- specification
  - red, green, and blue intensity

point light
- light emanates uniformly in all directions
- specification
  - location
  - red, green, and blue intensity
  - how the light drops off with distance

spot light
- light emanates in a cone
- specifications
  - location in world coordinates
  - red, green, and blue intensity
  - how the light drops off with distance
  - center axis how light drops off with angle from center

directional light
- light positioned at "infinity"; intensity and incident angle are constant for all points in scene
- specification
  - direction
  - red, green, and blue intensity

material properties
how does the surface material
- emit light
- reflect light
- transmit light

surface emitter
- Light emanates from (front/outward) surface of object in scene
diffuse reflection
rough/matte surface: light reflects uniformly in all directions

specular reflection
smooth (mirror) surfaces: light reflects in one (primary) direction

transparency
can light be transmitted through surface? is transmitted light refracted?

material property specification
1. red, green, and blue emission
2. ambient reflectivity for red, green, and blue light
3. diffuse reflectivity for red, green, and blue light
4. specular reflectivity for red, green, and blue light
5. transparency and refractive index
6. cheap tricks

overview
• models
  - simple shapes (for now)
  - transforms
• lights and material properties
• camera
  - frustum
  - projection
• illumination models
  - local
  - global

3d scene
lights
"world" coordinate system
models
view (eye/camera)
pinhole camera model

how is eye situated?

standard configuration

how much of the world is seen?

projection
view plane

almost always the view plane is:
- orthogonal to the toward vector \( \mathbf{t} \)
- some specified distance \( d \) from the viewpoint \( P_0 \)

overview

- models
  - simple shapes (for now)
  - transforms
- lights and material properties
- camera
  - frustum
  - Projection
- illumination models
  - local
  - global

lighting-global

what do i see?

lighting-local

what do i see?

direct illumination

maybe reflections
lighting-local

maybe multiple reflections

rendering

• image based algorithms
  – ray casting/tracing
• object-based algorithms
  – vertex pipeline

ray casting

• cast ray through pixel into scene
• find closest intersection (if any)
• compute luminance at intersection
**ray tracing**
- cast ray through pixel into scene
- find closest intersection (if any)
- compute luminance at intersection
  - direct illumination
  - reflections
  - transmission

**object-based**
- project each object
- hidden surface removal

**vertex pipeline**
- project vertices of each polygon
- turn on "inside" pixels
- use hidden surface removal to resolve conflicts