CS 380 - GPU and GPGPU Programming
Lecture 15: GPU Texturing 2

Markus Hadwiger, KAUST

Reading Assignment #9 (until Nov 2)

Read (required):

- MIP-Map Level Selection for Texture Mapping

  `https://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=765326`

- Homogeneous Coordinates

  `https://en.wikipedia.org/wiki/Homogeneous_coordinates`

- *Don't forget:*

  Interpolation for Polygon Texture Mapping and Shading,
  Paul Heckbert and Henry Moreton

  `http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.48.7886`

Read (optional):

- Vulkan Tutorial

  `https://vulkan-tutorial.com`

This week is IEEE VIS: registration is free

ieeevis.org

virtual.ieeevis.org

Quiz #2: Oct 28

Organization

- First 30 min of lecture
- No material (book, notes, ...) allowed

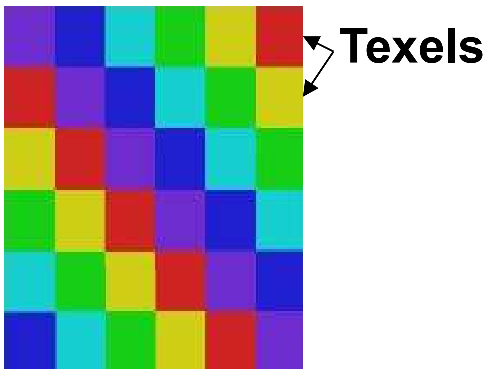
Content of questions

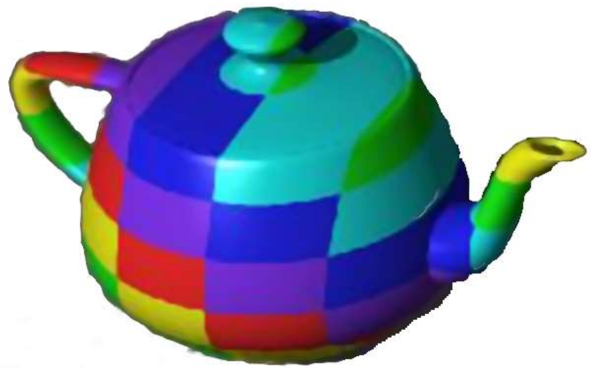
- Lectures (both actual lectures and slides)
- Reading assigments
- Programming assignments (algorithms, methods)
- Solve short practical examples

GPU Texturing

Rage / id Tech 5 (id Software)

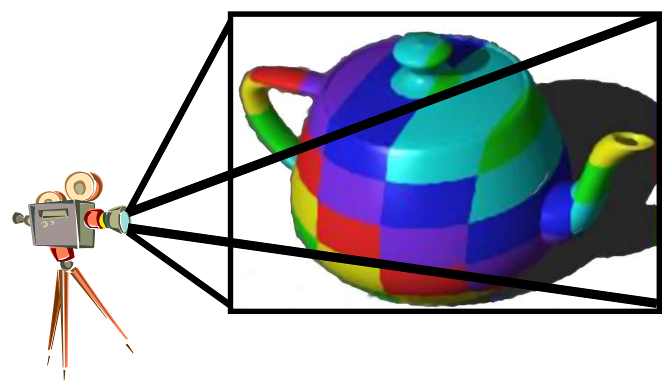
Texturing: General Approach





Texture space *(u,v)* **Object space** (x_O,y_O,z_O) **Image Space** (x_I,y_I)

Parametrization → **Rendering (Projection etc.)** →



Texture Mapping

2D (3D) Texture Space

Texture Transformation

2D Object Parameters

Parameterization

3D Object Space

Model Transformation

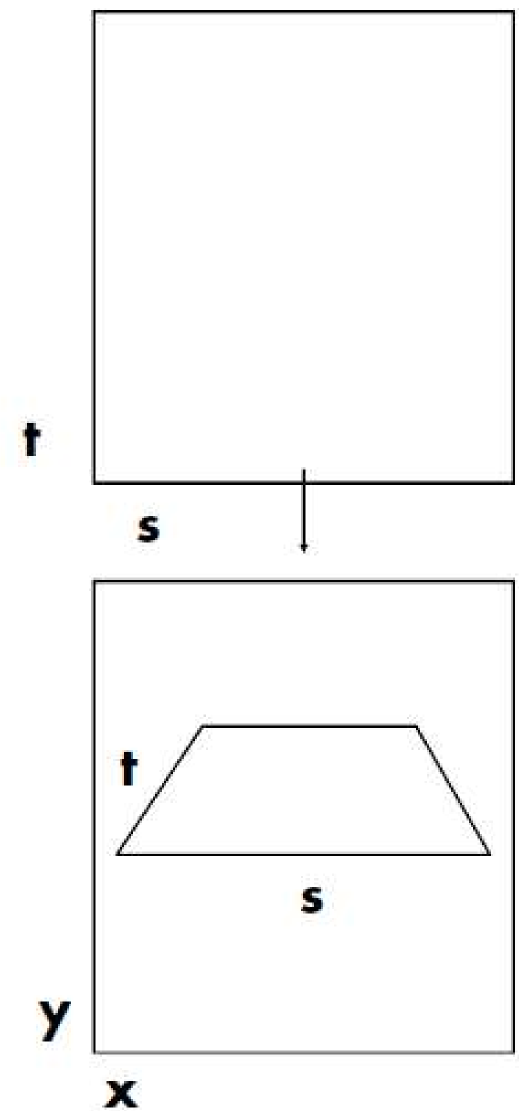
3D World Space

Viewing Transformation

3D Camera Space

Projection

2D Image Space



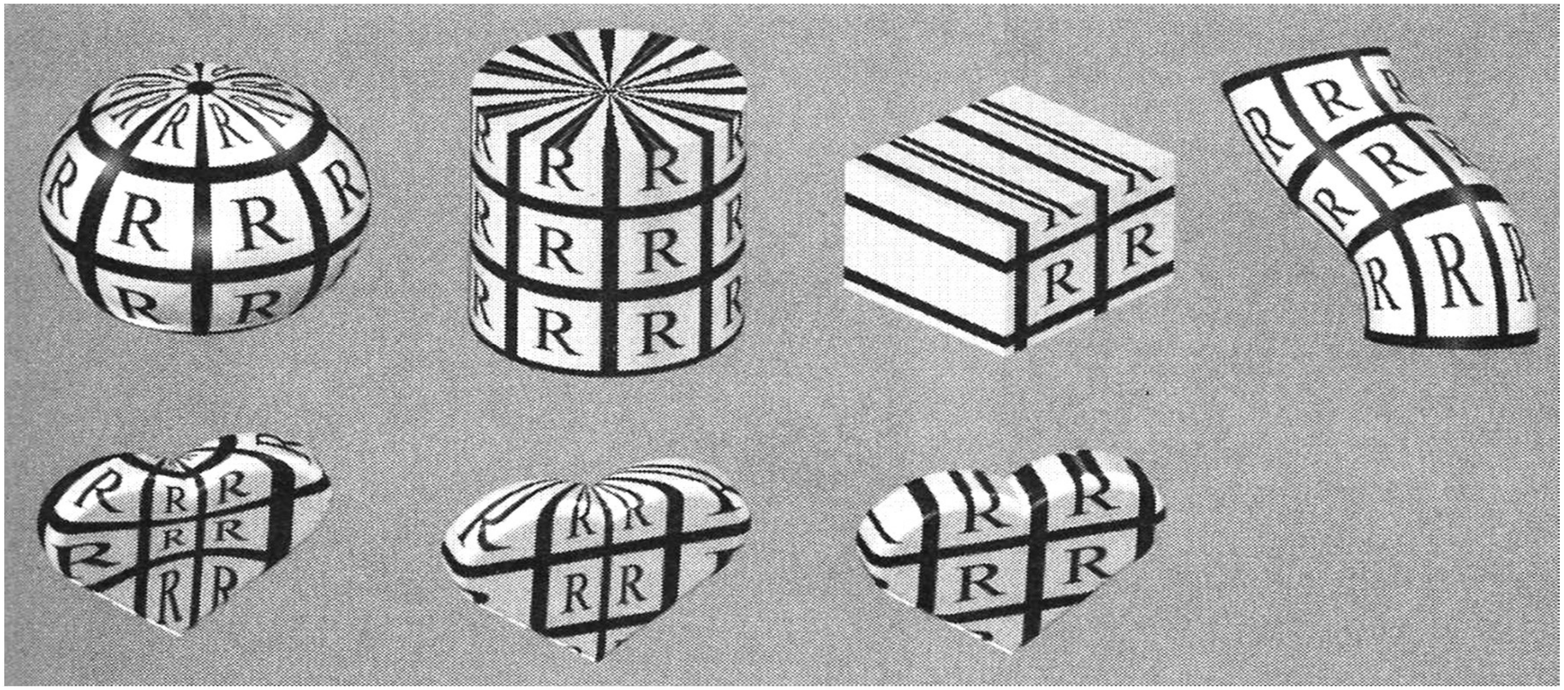
Kurt Akeley, Pat Hanrahan

Texture Projectors

Where do texture coordinates come from?

- Online: texture matrix/texcoord generation
- Offline: manually (or by modeling program)

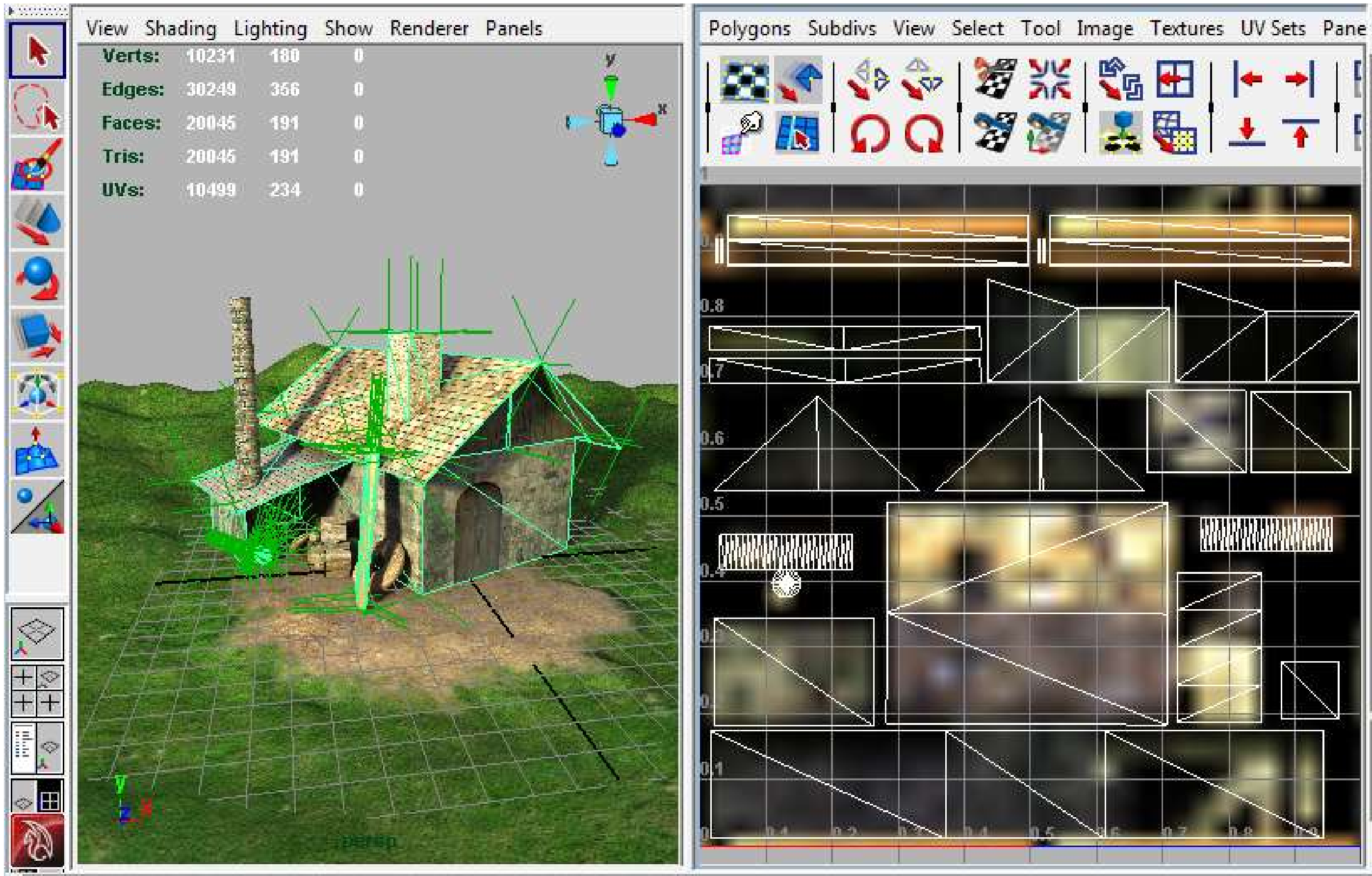
spherical *cylindrical* *planar* *natural*



Texture Projectors

Where do texture coordinates come from?

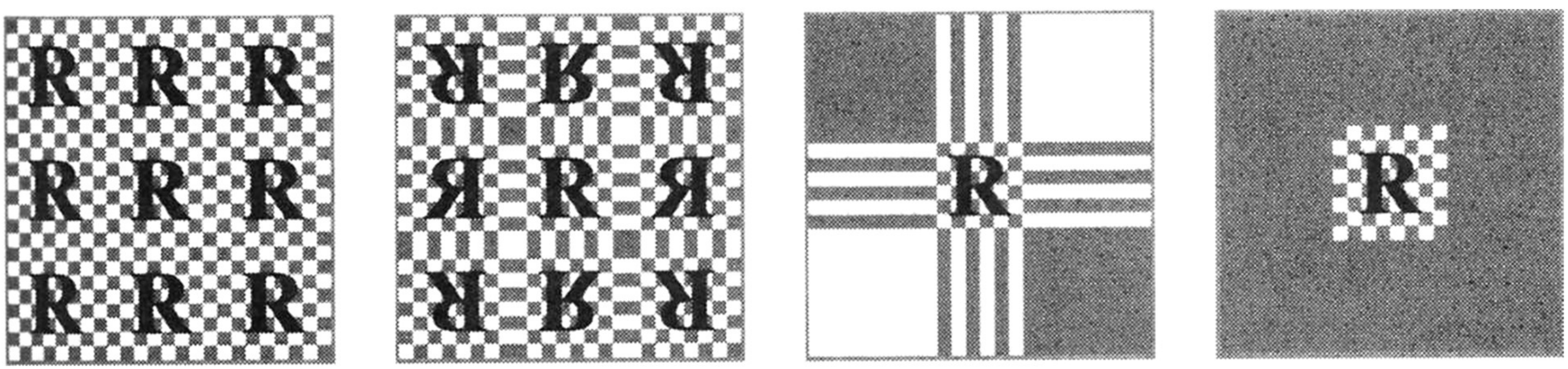
- Offline: manual UV coordinates by DCC program
- Note: a modeling problem!



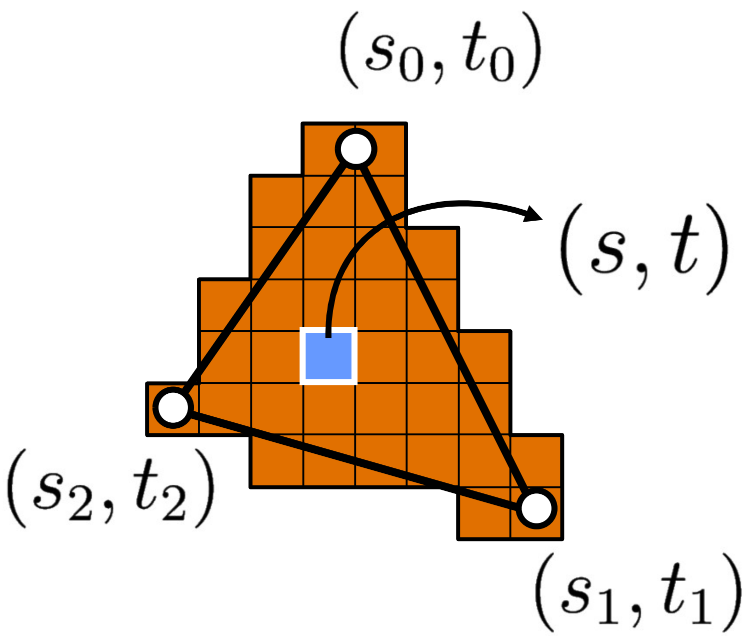
Texture Wrap Mode

- How to extend texture beyond the border?
- Border and repeat/clamp modes
- Arbitrary (s,t,…) → [0,1] x [0,1] → [0,255] x [0,255]

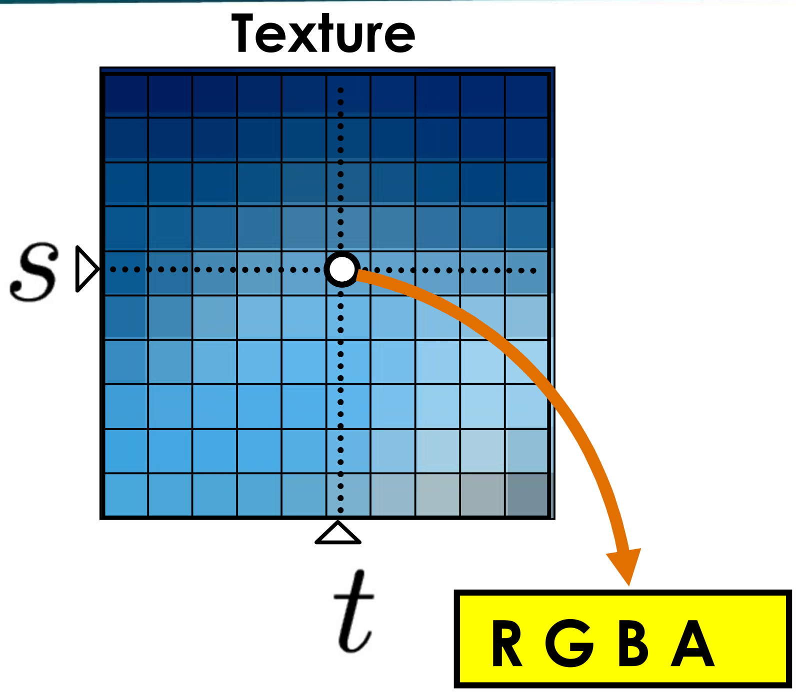
repeat mirror/repeat clamp border



2D Texture Mapping

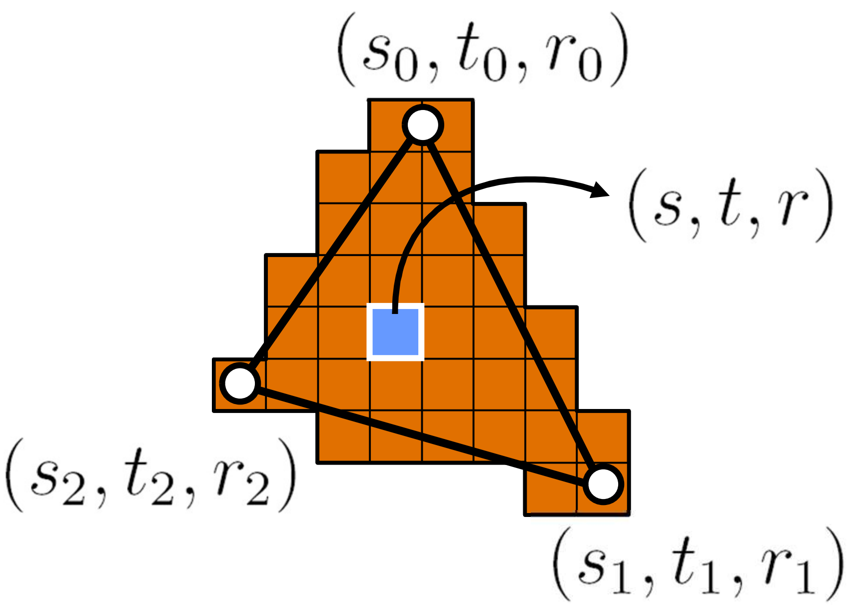


For each fragment:
interpolate the
texture coordinates
(barycentric)
Or:
Use arbitrary, computed coordinates

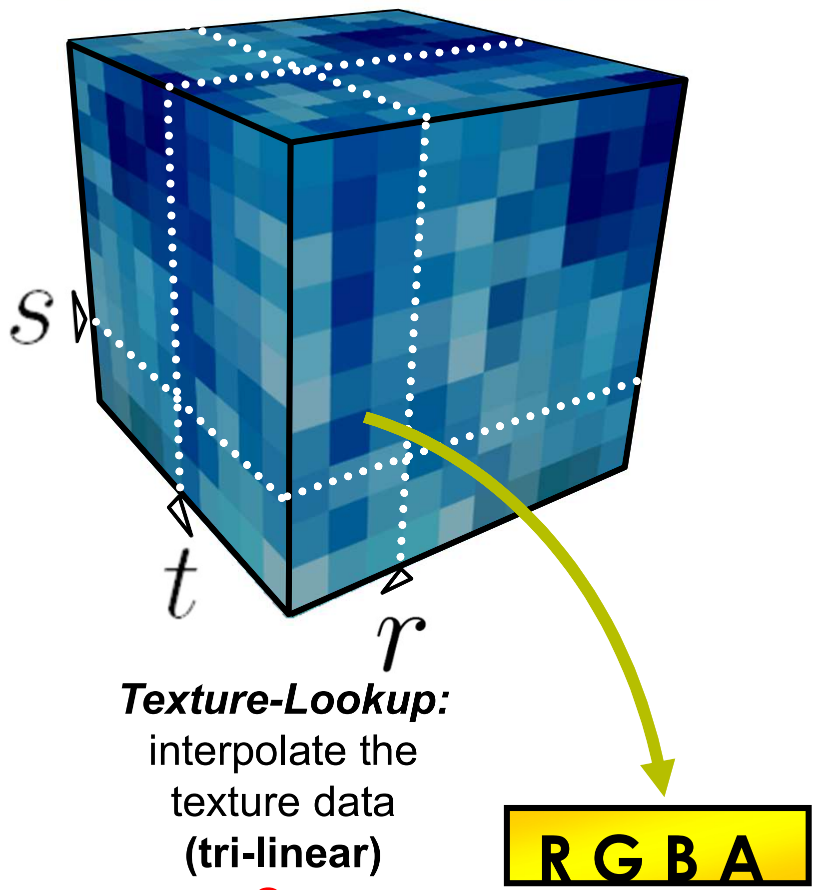


Texture-Lookup:
interpolate the
texture data
(bi-linear)
Or:
Nearest-neighbor for “array lookup”

3D Texture Mapping



For each fragment:
interpolate the
texture coordinates
(barycentric)
Or:
Use arbitrary, computed coordinates



Texture-Lookup:
interpolate the
texture data
(tri-linear)
Or:
Nearest-neighbor for "array lookup"

Interpolation Type + Purpose #1:

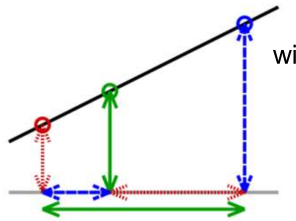
Interpolation of Texture Coordinates

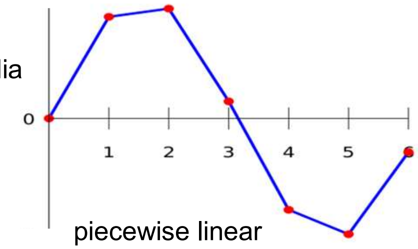
(Linear / Rational-Linear Interpolation)

Linear Interpolation / Convex Combinations

Linear interpolation in 1D:

$$f(\alpha) = (1-\alpha)v_1 + \alpha v_2$$





Line embedded in 2D (linear interpolation of vertex coordinates/attributes):

$$f(\alpha_1, \alpha_2) = \alpha_1 v_1 + \alpha_2 v_2 \qquad f(\alpha) = v_1 + \alpha(v_2 - v_1)$$

$$\alpha_1 + \alpha_2 = 1 \qquad \alpha = \alpha_2$$

Line segment: $\alpha_1, \alpha_2 \geq 0$ (→ convex combination)

Compare to line parameterization with parameter t:

$$v(t) = v_1 + t(v_2 - v_1)$$

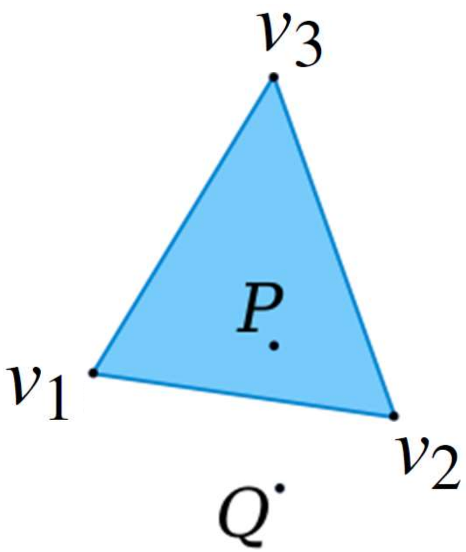
Linear Interpolation / Convex Combinations

Linear combination (n-dim. space):

$$\alpha_1 v_1 + \alpha_2 v_2 + \ldots + \alpha_n v_n = \sum_{i=1}^{n} \alpha_i v_i$$

Affine combination: Restrict to $(n-1)$-dim. subspace:

$$\alpha_1 + \alpha_2 + \ldots + \alpha_n = \sum_{i=1}^{n} \alpha_i = 1$$



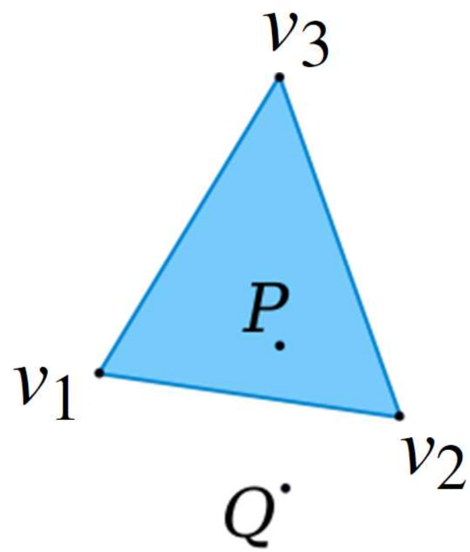
Convex combination: $\alpha_i \geq 0$

(restrict to simplex in subspace)

Linear Interpolation / Convex Combinations

$$\alpha_1 v_1 + \alpha_2 v_2 + \ldots + \alpha_n v_n = \sum_{i=1}^{n} \alpha_i v_i$$

$$\alpha_1 + \alpha_2 + \ldots + \alpha_n = \sum_{i=1}^{n} \alpha_i = 1$$



Re-parameterize to get affine coordinates:

$$\begin{aligned} \alpha_1 v_1 + \alpha_2 v_2 + \alpha_3 v_3 &= \\ \tilde{\alpha}_1 (v_2 - v_1) + \tilde{\alpha}_2 (v_3 - v_1) + v_1 & \\ \tilde{\alpha}_1 &= \alpha_2 \\ \tilde{\alpha}_2 &= \alpha_3 \end{aligned}$$

Linear Interpolation / Convex Combinations

The weights α_i are the (normalized) barycentric coordinates

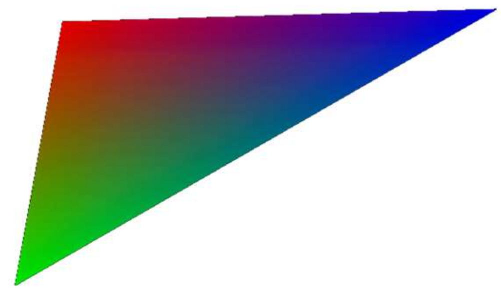
→ linear attribute interpolation in simplex

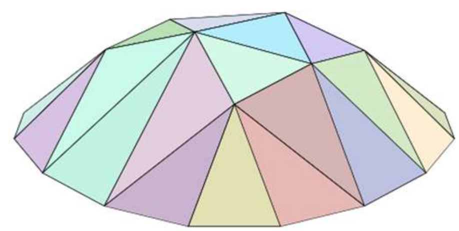
$$\alpha_1 v_1 + \alpha_2 v_2 + \ldots + \alpha_n v_n = \sum_{i=1}^{n} \alpha_i v_i$$

$$\alpha_1 + \alpha_2 + \ldots + \alpha_n = \sum_{i=1}^{n} \alpha_i = 1$$

$$\alpha_i \geq 0$$

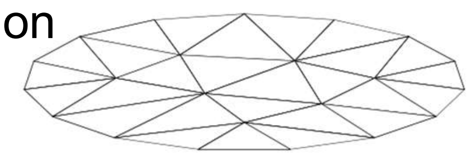
attribute interpolation





spatial position interpolation

wikipedia



Homogeneous Coordinates (1)

Projective geometry

- (Real) projective spaces RP^n:

  Real projective line RP^1, real projective plane RP^2, ...

- A point in RP^n is a line through the origin (i.e., all the scalar multiples of the same vector) in an (n+1)-dimensional (real) vector space

Homogeneous coordinates of 2D projective point in RP^2

- Coordinates differing only by a non-zero factor λ map to the same point

  (λx, λy, λ) dividing out the λ gives (x, y, 1), corresponding to (x,y) in R^2

- Coordinates with last component = 0 map to "points at infinity"

  (λx, λy, 0) division by last component not allowed; but again this is the same point if it only differs by a scalar factor, e.g., this is the same point as (x, y, 0)

Homogeneous Coordinates (2)

Examples of usage

- Translation (with translation vector $\vec{b}$)
- Affine transformations (linear transformation + translation)

$$\vec{y} = A\vec{x} + \vec{b}.$$

- With homogeneous coordinates:

$$\begin{bmatrix} \vec{y} \\ 1 \end{bmatrix} = \left[\begin{array}{ccc|c} & A & & \vec{b} \\ 0 & \ldots & 0 & 1 \end{array}\right] \begin{bmatrix} \vec{x} \\ 1 \end{bmatrix}$$

- Setting the last coordinate = 1 and the last row of the matrix to [0, ..., 0, 1] results in translation of the point $\vec{x}$ (via addition of translation vector $\vec{b}$)
- The matrix above is a linear map, but because it is one dimension higher, it does not have to move the origin in the (n+1)-dimensional space for translation

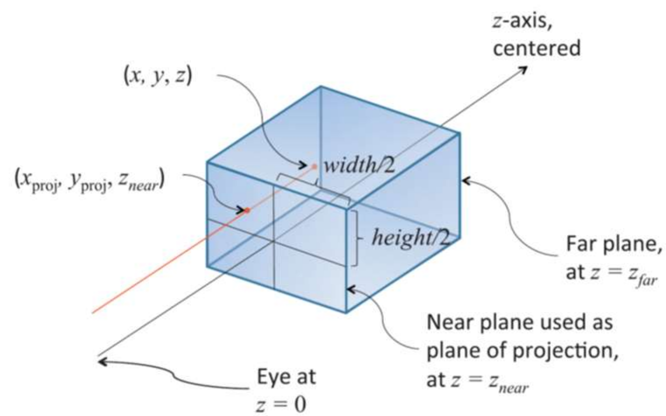
Homogeneous Coordinates (3)

Examples of usage

- Projection (e.g., OpenGL projection matrices)

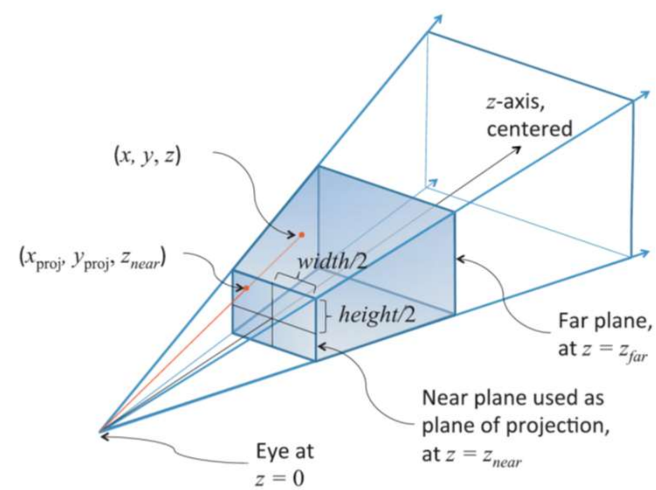
$$\begin{bmatrix} \frac{2}{\text{right}-\text{left}} & 0 & 0 & -\frac{\text{right}+\text{left}}{\text{right}-\text{left}} \\ 0 & \frac{2}{\text{top}-\text{bottom}} & 0 & -\frac{\text{top}+\text{bottom}}{\text{top}-\text{bottom}} \\ 0 & 0 & \frac{-2}{\text{far}-\text{near}} & -\frac{\text{far}+\text{near}}{\text{far}-\text{near}} \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

orthographic



$$\begin{bmatrix} \frac{z_{near}}{width/2} & 0.0 & \frac{left+right}{width/2} & 0.0 \\ 0.0 & \frac{z_{near}}{height/2} & \frac{top+bottom}{height/2} & 0.0 \\ 0.0 & 0.0 & -\frac{z_{far}+z_{near}}{z_{far}-z_{near}} & \frac{2z_{far}z_{near}}{z_{far}-z_{near}} \\ 0.0 & 0.0 & -1.0 & 0.0 \end{bmatrix}$$

perspective



Thank you.