Interpolation & Revisiting Homogeneous Coordinate System

CSE606: Computer Graphics
Jaya Sreevalsan Nair, IIIT Bangalore
March 17 and April 21, 2025

Interpolation

Introduction to Parametrization

Parametrization is a computation usually done to reduce the dimensionality of representation to be the same as the true geometrical dimensionality. e.g. a line and triangle in 3-dimensional space must be represented using 1-dimensional and 2-dimensional mapping, respectively.

True geometrical dimensionality corresponds to the number of independent variables to define the primitive.

Parametrization

We use parametrization for interpolation. e.g. t is a parameter in a line-segment AB, where $t(A) = 0$, $t(B) = 1$.

The parametrization can be computed using fractions of lengths, areas, volumes, etc.

What is Interpolation?

Interpolation is a method used for finding value of an attribute at an interior point of a geometric primitive, based on the values of the attribute at the vertices of the primitive.

- The attributes include (x-, y-, z-) coordinates of position, coordinates of normal vector, color channels (e.g. red, green, blue), etc.

Interpolation Methods

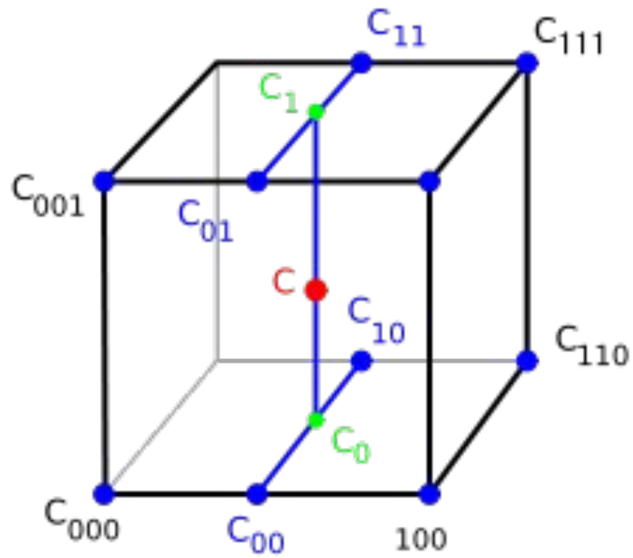
Interpolation uses an interpolant, or polynomial of parameters of at least degree d for a d-dimensional space.

- e.g. linear interpolant is of degree 1, quadratic interpolant is of degree 2, bilinear interpolant is of degree 2 (linear in 2 dimensions).
- Linear interpolants are preferred owing to reduced number of arithmetic operations (multiplications and additions) involved in computing the interpolant.

Linear, Bilinear and Trilinear Interpolation

Using parametric representation: $0 \leq s, t, u \leq 1$.

Image courtesy: http://en.wikipedia.org/wiki/Trilinear_interpolation



Linear Interpolation:

$$f(C) = (1-u)\cdot f(C_0) + u \cdot f(C_1)$$

Bilinear interpolation:

$$f(C_0) = (1-t)\cdot f(C_{00}) + t \cdot f(C_{01})$$
$$f(C_1) = (1-t)\cdot f(C_{10}) + t \cdot f(C_{11})$$

Trilinear interpolation:

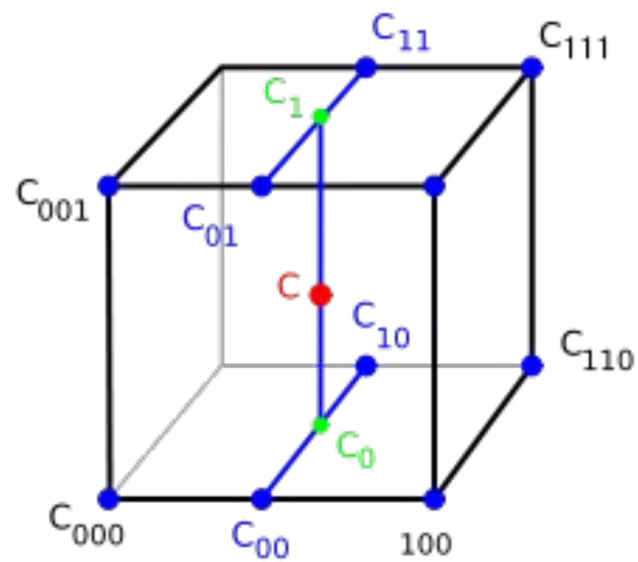
$$f(C_{00}) = (1-s)\cdot f(C_{000}) + s \cdot f(C_{100})$$
$$f(C_{01}) = (1-s)\cdot f(C_{001}) + s \cdot f(C_{101})$$
$$f(C_{11}) = (1-s)\cdot f(C_{011}) + s \cdot f(C_{111})$$
$$f(C_{10}) = (1-s)\cdot f(C_{010}) + s \cdot f(C_{110})$$

Note: we compute the parameter using fractions of lengths, e.g. $u = d(C,C_1)/d(C_0,C_1)$

Linear, Bilinear and Trilinear Interpolation

Using parametric representation: $0 \leq s, t, u \leq 1$.

Image courtesy: http://en.wikipedia.org/wiki/Trilinear_interpolation



Linear Interpolation:

$$f(C) = (1-u)\cdot f(C_0) + u \cdot f(C_1)$$

Bilinear interpolation:

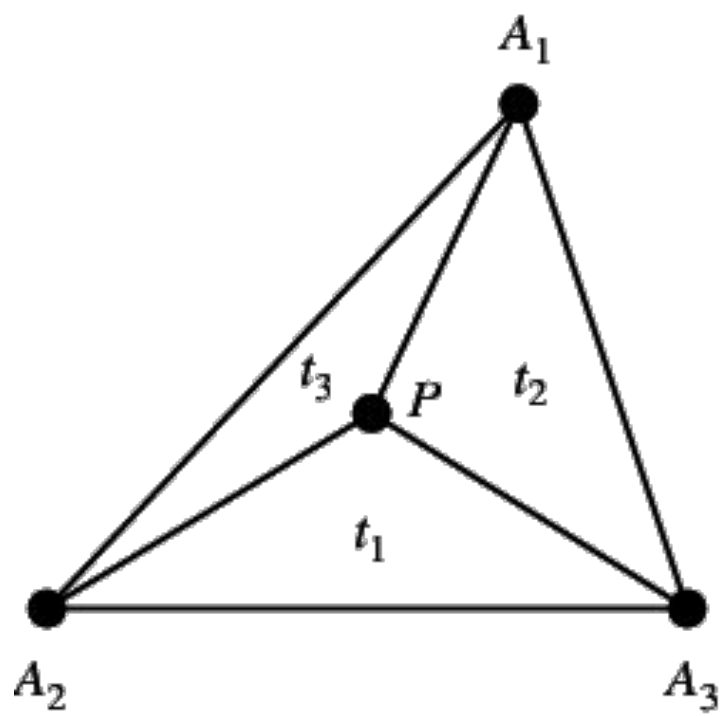
$$\begin{aligned} f(C) = {} & (1-t)(1-u)\cdot f(C_{00}) + t(1-u)\cdot f(C_{10}) + \\ & tu \cdot f(C_{11}) + (1-t)u \cdot f(C_{01}) \end{aligned}$$

Trilinear interpolation:

$$\begin{aligned} f(C) = {} & (1-s)(1-t)(1-u)\cdot f(C_{000}) + s(1-t)(1-u)\cdot f(C_{100}) + \\ & st(1-u)\cdot f(C_{101}) + (1-s)t(1-u)\cdot f(C_{001}) + \\ & (1-s)tu \cdot f(C_{011}) + stu \cdot f(C_{111}) + \\ & st(1-u)\cdot f(C_{110}) + (1-s)t(1-u)\cdot f(C_{010}) \end{aligned}$$

Barycentric Coordinate System

Image courtesy: http://en.wikipedia.org/wiki/Trilinear_interpolation



The linear, bilinear and trilinear interpolation is used for finding value (of position coordinates, color channels, normal coordinates, etc.) at an interior point in a line-segment, rectangle, and cuboid, respectively.

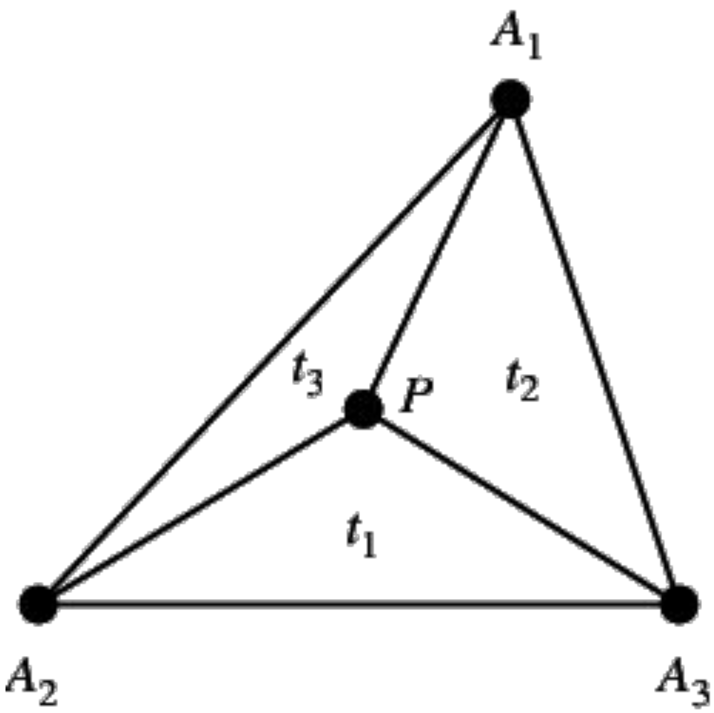
For other geometric entities in 1D, 2D, and 3D, e.g. curved elements, one would parametrically map them to line-segment, rectangle, and cuboid, respectively, and then perform the interpolation.

The linear interpolation is done on triangles using barycentric coordinate system.

Barycentric Coordinates and Linear Interpolation

Image courtesy: http://en.wikipedia.org/wiki/Trilinear_interpolation

Since triangles are 2D, we need 2 independent variables, but since there are 3 vertices, we can have an additional dependent variable.



Thus we have: $\alpha(P) + \beta(P) + \gamma(P) = 1$

for an interior point P in $\Delta(A_1 A_2 A_3)$ where two of the variables α, β, γ are independent and the remaining one becomes dependent.

Barycentric Coordinates and Linear Interpolation

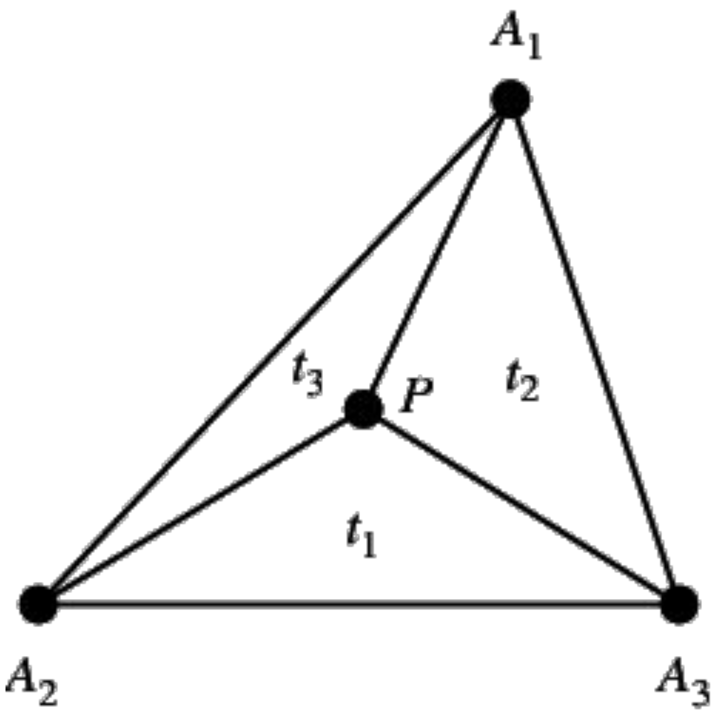
Image courtesy: http://en.wikipedia.org/wiki/Trilinear_interpolation

Using (areal) barycentric coordinates, we get:

$$\alpha(P) = \frac{Area(\Delta(A_2 P A_3))}{Area(\Delta(A_1 A_2 A_3))}$$

$$\beta(P) = \frac{Area(\Delta(A_3 P A_1))}{Area(\Delta(A_1 A_2 A_3))}$$

$$\gamma(P) = \frac{Area(\Delta(A_1 P A_2))}{Area(\Delta(A_1 A_2 A_3))}$$



$$\alpha(P) + \beta(P) + \gamma(P) = \frac{Area(\Delta(A_2 P A_3)) + Area(\Delta(A_3 P A_1)) + Area(\Delta(A_1 P A_2))}{Area(\Delta(A_1 A_2 A_3))}$$

$$= \frac{Area(\Delta(A_1 A_2 A_3))}{Area(\Delta(A_1 A_2 A_3))} = 1$$

Application of Interpolation – Color Mapping

Colormap generation – Identifying color palettes ⇒ a sequence of colors matching an interval of values (continuous spectrum).

Generation techniques:

- Color cube,
- Rainbow spectrum,
- Grayscale spectrum,
- Linear interpolation

Figure 12. Various color maps applied to a face image. The color maps in the bottom row have monotonically increasing lightness, resulting in a natural, recognizable image.

Courtesy: Rainbow Color Map Critiques: An Overview and Annotated Bibliography, Steve Eddins, MathWorks.

Color Scales

(Left) Viridis (R Package) color scales; (right) ggplot color scales.

https://www.youtube.com/watch?v=u9a4NO3iGgA
[All colormaps (Matplotlib) (2015) by Nathaniel Smith.]

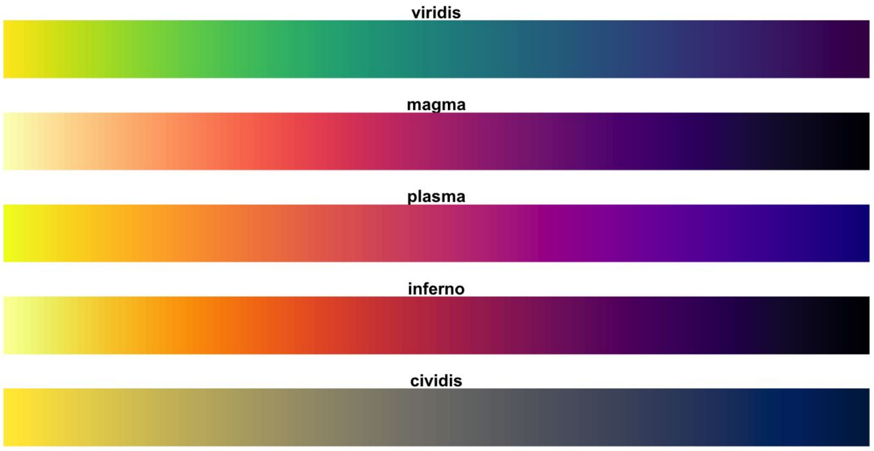


Image courtesy: https://cran.r- project.org/web/packages/viridis/vignettes/intro- to- viridis.html

Applications of Interpolation – Color Mapping

- Parametric mapping using linear interpolation
  - For a line segment PQ, any point in the interior of the line segment $R = (1-\alpha)P + \alpha Q$, where the parameter corresponding to R is α = length(R,P)/length(Q,P).
  - Apart from position coordinates, all properties with linear behaviour can use this formula: $C(R) = (1-\alpha)C(P) + \alpha C(Q)$.
- Colour spectrum: 2-colour, 3-colour, n-colour (rainbow colour spectrum)

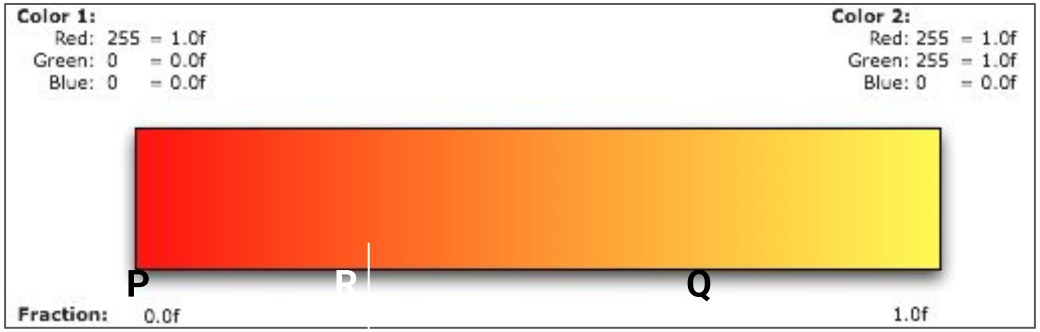


Image courtesy: https://harmoniccode.blogspot.com/2011/04/bilinear-color-interpolation.html

Revisiting Transformations

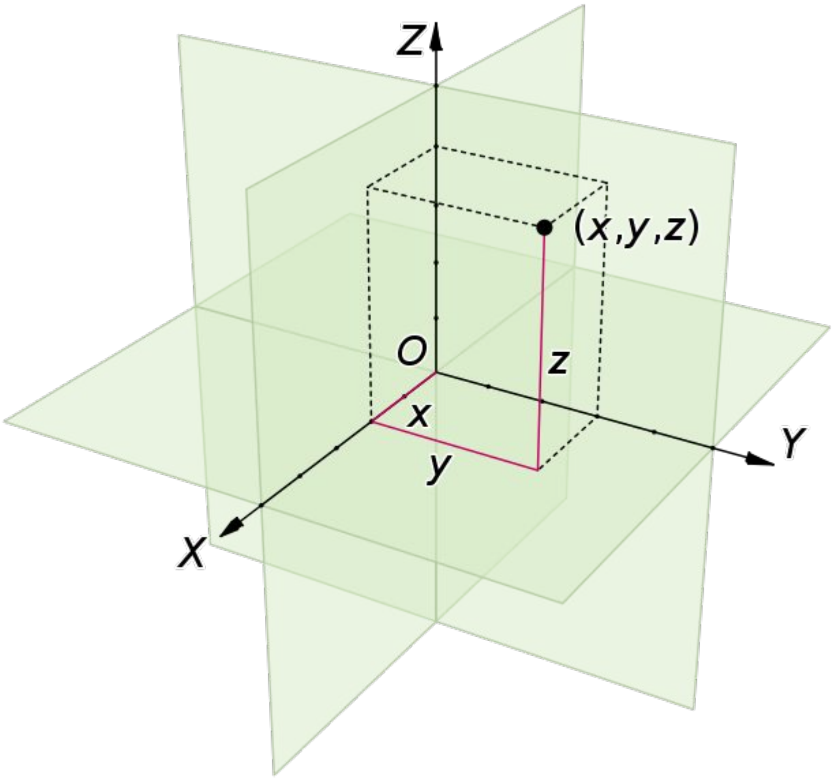
Euclidean Space

Euclidean n-space ($\mathbb{R}^n$ or Cartesian space): Space of all n-tuples of real numbers, which

- corresponds to physical space.
- contains elements which are n-vectors.

Examples:

- Real line: $\mathbb{R}^1$
- Euclidean plane: $\mathbb{R}^2$
- Points: $[2, 3]^T \in \mathbb{R}^2$; $[2, 3, 4]^T \in \mathbb{R}^3$



Three-dimensional Coordinate System. (Image courtesy: Wikimedia Commons.)

Vector Space

A vector space is a set that is closed under finite vector addition and scalar multiplication.

- Vector space is defined over a field F; where the scalars belong to F.
- Field: any set of elements that satisfies field axioms for both addition and multiplication and is a commutative division algebra.
    - Complex numbers, rational numbers, and real numbers form a field; but not integers.
- Field axioms: Commutativity, Associativity, Distributivity, Identity, Inverse.

Vector Space

A set of vectors is defined to be **linearly independent**, if none of the vectors can be expressed as scalar-vector addition of others.

Dimension of a vector space: Maximum number of linearly independent vectors in the space.

Affine Spaces

Let, V : a vector space over a field K ; A: a nonempty set.

Define addition $(p + a) \in A$, for any vector $(a, b \in V)$ and element $p \in A$, which are subject to:

- $p + 0 = p$.
- $(p + a) + b = p + (a + b)$.
- For any $q \in A$, there exists a unique vector $a \in V$, such that $q = p + a$.

Then A is called an **affine space** and K is called the **coefficient field**.

Summary of Vector and Affine Spaces

(Linear) vector space: contains vectors & scalars, and most importantly an "origin".

Affine space: Geometric structure similar to a vector space that does not include the origin; but has "point" objects.

Euclidean space: consists of an affine space (V,K) and a scalar product on V, and additionally includes measure of size (e.g. length of line segment, angle of sector).

Summary of Vector and Affine Spaces

(Linear) vector space: contains vectors & scalars, and most importantly an "origin".

Affine space: Geometric structure similar to a vector space that does not include the origin; but has "point" objects.

Euclidean space: consists of an affine space (V,K) and a scalar product on V, and additionally includes measure of size (e.g. length of line segment, angle of sector).

- In affine spaces: point-point subtraction yields a vector; vector-point addition yields a new point.
- In object-oriented programming, we can use Abstract Data Types (ADT) to define an affine space.

Affine Addition

Point-point addition and scalar-point multiplication are not allowed in affine spaces.

- However, this can be accomplished using affine addition.
- For points P, Q, R, vector v , and scalar α:
  - $P = Q + \alpha v$;
  - $v = R - Q$

Thus, $P = Q + \alpha(R - Q) = (1 - \alpha)Q + \alpha R$, which is a **linear combination of *points***.

Affine Addition

Point-point addition and scalar-point multiplication are not allowed in affine spaces.

- However, this can be accomplished using affine addition.
- For points P, Q, R, vector v , and scalar α:
  - $P = Q + \alpha v$;
  - $v = R - Q$

Thus, $P = Q + \alpha(R - Q) = (1 - \alpha)Q + \alpha R$, which is a **linear combination of *points***.

Applications in *parametric definitions* of lines, planes, and convexity

Affine Addition Application – Lines

In parametric form, for a ray starting at point P_0 ,

$$P(\alpha) = P_0 + \alpha.\mathbf{d}$$

for an arbitrary vector **d**, and scalar α.

For a line segment between points P_0 and P_1 , the parametric form using affine sums is, for $(0 \leq \alpha \leq 1)$:

$$P(\alpha) = P_0 + \alpha.(P_1 - P_0) = (1-\alpha).P_0 + \alpha.P_1$$

Affine Addition Application – Convexity

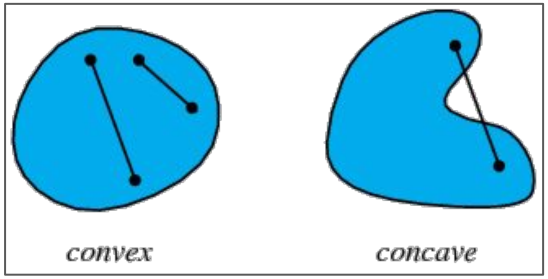
For a convex set of points $P_1, P_2, \ldots, P_n$, a point P lies in the convex hull, given,

$$P = \alpha_1.P_1 + \alpha_2.P_2 + \ldots + \alpha_n.P_n, \qquad \textit{if and only if,}$$

coefficients form **partition of unity**, i.e.,

$$\alpha_1+\alpha_2+\ldots+\alpha_n = 1.0 \text{ and } (\alpha_i \geq 0), \ \forall\ (i \in [1,n]).$$

- Partition of unity guarantees an interior point.
- Convex hull: Minimal set of points obtained from shrink-wrapping the convex set of points.



Convex vs. Concave Set (Image courtesy: Wolfram Mathworld.)

Affine Addition Application – Planes

For vectors, **u**, **v**,

$$\text{Dot Product: } \mathbf{u} \cdot \mathbf{v} = |\mathbf{u}|.|\mathbf{v}|.\cos\theta$$

$$\text{Cross Product: } |\mathbf{u} \times \mathbf{v}| = |\mathbf{u}|.|\mathbf{v}|.|\sin\theta|$$

where **u**, **v**, and (**u × v**) form a right-handed coordinate system.

Parametric form of a plane containing point P**0** and non-parallel vectors, **u**, **v**, and scalars, α, β: $T(\alpha,\beta) = P_0 + \alpha\mathbf{u} + \beta\mathbf{v}$

- Its equivalent vector form, $(\mathbf{u} \times \mathbf{v})\cdot(P-P_0) = 0$

Coordinate Systems

Basis vectors of a coordinate system: Linearly independent vectors whose linear combinations can be used for representing all points in the coordinate system.

$$w = \alpha_1\mathbf{v_1} + \alpha_2\mathbf{v_2} + \alpha_3\mathbf{v_3}$$

where, α_1 , α_2 , α_3 are components of vector w with respect to the basis vectors $\mathbf{v_1}$, $\mathbf{v_2}$, $\mathbf{v_3}$.

Rewriting it, if a = [α_1 , α_2 , α_3] and v=[$\mathbf{v_1}$, $\mathbf{v_2}$, $\mathbf{v_3}$], then

$$w = a^T.v$$

N-Tuple Representations

Basis vectors are represented as unit vectors in the form of 3-tuples:

$$\mathbf{v}_1, \mathbf{v}_2, \mathbf{v}_3 = [1,0,0]^T, [0,1,0]^T, [0,0,1]^T$$

Similarly for Euclidean space, $\mathbb{R}^n$, basis vectors are n-tuples.

n-tuple representation is thus, equivalent to vector representation of the vector space.

Frames

Frame: constituted by origin (reference point) and basis vectors.

A frame is required to uniquely define all points and all vectors in a coordinate system.

$$\text{Vector } \mathbf{w} = \alpha_1\mathbf{v_1} + \alpha_2\mathbf{v_2} + \alpha_3\mathbf{v_3} \quad \Longrightarrow \quad \mathbf{w} = a^T.v$$

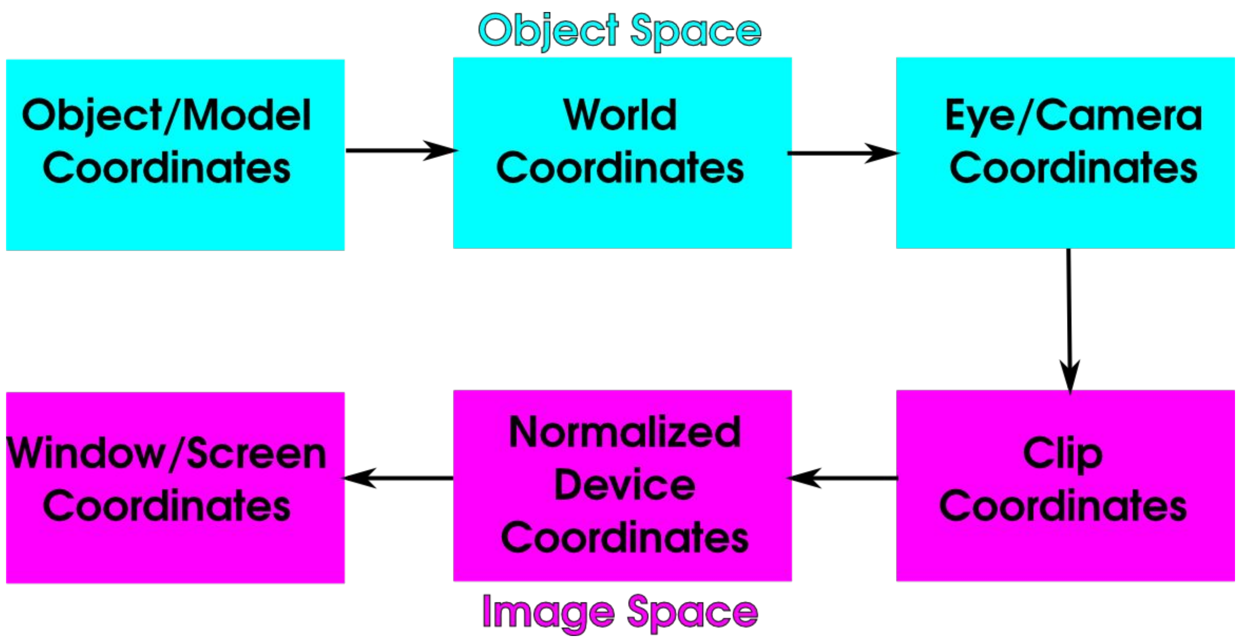
$$\text{Point } P = P_0 + \beta_1\mathbf{v_1} + \beta_2\mathbf{v_2} + \beta_3\mathbf{v_3} \quad \Longrightarrow \quad P = P_0 + b^T.v$$

Frame is represented as $[\mathbf{v_1}, \mathbf{v_2}, \mathbf{v_3}, P_0]$.

Note: Points and vectors are always distinct geometric types.

Frame Transformations in OpenGL Pipeline

Change in coordinate system $\Longrightarrow$ change in vector representation based on change in basis vectors.



Transformation

Definition: A function that maps a point (or vector) to another point (or vector).

In functional form, Q = T(P), **v** = R(**u**)

Transformations are too general to be useful, hence a (restricted) class of transformations is used in computer graphics, namely, **the linear transformations**.

Linear transformation: $f(\alpha p + \beta q) = \alpha f(p) + \beta f(q)$

Linear Transformations

Transformations of linear combination of entities is a **linear combination of transformation** of the entities.

It has the advantage of saving several recalculations.

However, linearity is a restriction, as computer graphics also has non-linear behavior, e.g. perspective projection.

Affine Transformations

An **affine transformation** is equivalent to a linear transformation combined with vector addition.

$$f(\mathbf{u}) = \mathbf{v} = T(\mathbf{u}) + \mathbf{w}$$

for $f : X \longrightarrow Y$ for affine spaces, X and Y, such that $\mathbf{u} \in X$ and $\mathbf{w} \in Y$.

Properties of Affine Transformations:

- Parallel lines remain parallel.
- The midpoint of a line segment remains a midpoint.
- All points on a straight line remain on a straight line..

Affine Transformations in Computer Graphics

Most of the transformations in Computer Graphics are affine.

- Basic types of affine transformations include translation, rotation, scaling, shear.
- All affine transformations can be constructed as sequence of basic types.

Rigid-body transformations: Transformations with no alteration to shape or volume of object. e.g., Translation, Rotation.

Non-rigid-body transformations: Transformations that alter shape or volume of an object. e.g., Uniform & Non-uniform scaling, Shear.

Homogeneous Coordinate System
(in detail)

Transformation Matrices

Matrix representation is used in graphics processing for:

1. Change in coordinate system (i.e. frame transformations)
2. Affine transformations of vectors

Application 1 – Change of Coordinate System

In a transformation, basis of new coordinate system can be expressed in terms of the old coordinate system in the following way:

$$
\begin{aligned}
u_1 &= \gamma_{11} v_1 + \gamma_{12} v_2 + \gamma_{13} v_3 \\
u_2 &= \gamma_{21} v_1 + \gamma_{22} v_2 + \gamma_{23} v_3 \\
u_3 &= \gamma_{31} v_1 + \gamma_{32} v_2 + \gamma_{33} v_3
\end{aligned}
$$

$$
\begin{bmatrix} u_1 \\ u_2 \\ u_3 \end{bmatrix} = \begin{bmatrix} \gamma_{11} & \gamma_{12} & \gamma_{13} \\ \gamma_{21} & \gamma_{22} & \gamma_{23} \\ \gamma_{31} & \gamma_{32} & \gamma_{33} \end{bmatrix} \begin{bmatrix} v_1 \\ v_2 \\ v_3 \end{bmatrix}
$$

Thus, $u = M.v$

Vector Transformation due to Frame Transformation

Starting from: $u = M.v$

Now, a vector **w** is represented in the old and new coordinate systems:

$$
\begin{aligned}
w &= \beta_1 u_1 + \beta_2 u_2 + \beta_3 u_3 \\
&= \alpha_1 v_1 + \alpha_2 v_2 + \alpha_3 v_3 \\
\Rightarrow w &= b^T u = a^T v
\end{aligned}
$$

Vector Transformation due to Frame Transformation

We just got: $u = M.v$

Thus, a vector **w** is represented in the old and new coordinate systems:

$$
\begin{aligned}
w &= \beta_1 u_1 + \beta_2 u_2 + \beta_3 u_3 \\
&= \alpha_1 v_1 + \alpha_2 v_2 + \alpha_3 v_3 \\
\Rightarrow w &= b^T u = a^T v
\end{aligned}
$$

Also, $b^T u = b^T M v, \Rightarrow a = M^T b$

Thus using the matrix inverse, we get: $b = (M^T)^{-1} a$

- These rotations leave the origin unchanged.
- How can linear transformations accommodate change of origin?

Application 2 – Affine Transformations of a Vector

Rotation by an angle θ about +z axis, scaling, translation.

$$\text{Rotation}: \begin{bmatrix} x \\ y \end{bmatrix} \rightarrow \begin{bmatrix} x\cos\theta - y\sin\theta \\ x\sin\theta + y\cos\theta \end{bmatrix}; \Rightarrow M_{R_z} = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix}$$

$$\text{Scaling}: \begin{bmatrix} x \\ y \end{bmatrix} \rightarrow \begin{bmatrix} k_x x \\ k_y y \end{bmatrix}; \Rightarrow M_S = \begin{bmatrix} k_x & 0 \\ 0 & k_y \end{bmatrix}$$

$$\text{Translation}: \begin{bmatrix} x \\ y \end{bmatrix} \rightarrow \begin{bmatrix} x + t_x \\ y + t_y \end{bmatrix}; \Rightarrow M_T = ?$$

Rotation and scaling can be represented as (matrix representations of) linear transformations, but translation cannot be represented such.

Need for a New Representation System

1. To differentiate between representation of a point and direction vector.
2. To represent translation as a matrix transformation, in order to have uniform representation for all object transformations.
    - To enable representing affine transformations in the form of linear transformations, as the latter is more efficient/ convenient to manage.
3. To represent frame transformations in the form of linear transformations, even after adding translation of the origin.

[#1 is for representation, #2 and #3 are for transformation matrices.]

Solution – Homogeneous Coordinates

Solution is the use of homogeneous coordinate system, which is the (n+1)-dimensional representation of n-dimensional space, thus giving (2n+1) extra elements in the corresponding matrix to represent n-dimensional space.

Homogeneous Coordinates: Point/Vector Representation

In a frame specified by ($\mathbf{v}_1$, $\mathbf{v}_2$, $\mathbf{v}_3$, P_0), a point P and vector **w** are given by:

$$P = \begin{bmatrix} \alpha_1 & \alpha_2 & \alpha_3 & 1 \end{bmatrix} \begin{bmatrix} v_1 \\ v_2 \\ v_3 \\ P_0 \end{bmatrix}; \; w = \begin{bmatrix} \delta_1 & \delta_2 & \delta_3 & 0 \end{bmatrix} \begin{bmatrix} v_1 \\ v_2 \\ v_3 \\ P_0 \end{bmatrix}$$

Thus, in a given frame we can represent P and **w** as column matrices.

$$P = \begin{bmatrix} \alpha_1 & \alpha_2 & \alpha_3 & 1 \end{bmatrix}^T$$
$$w = \begin{bmatrix} \delta_1 & \delta_2 & \delta_3 & 0 \end{bmatrix}^T.$$

Homogeneous Coordinates: Frame Transformation Matrix

Consider change of frames: $(v_1, v_2, v_3, P_0) \rightarrow (u_1, u_2, u_3, Q_0)$

For:

$$u = \begin{bmatrix} u_1 & u_2 & u_3 & Q_0 \end{bmatrix}^T$$

$$v = \begin{bmatrix} v_1 & v_2 & v_3 & P_0 \end{bmatrix}^T$$

$$u = Mv$$

The transformation matrix is given by:

$$M = \begin{bmatrix} \gamma_{11} & \gamma_{12} & \gamma_{13} & 0 \\ \gamma_{21} & \gamma_{22} & \gamma_{23} & 0 \\ \gamma_{31} & \gamma_{32} & \gamma_{33} & 0 \\ \gamma_{41} & \gamma_{42} & \gamma_{43} & 1 \end{bmatrix}$$

When transforming vectors, the coefficient vectors transform as:

$$a = M^T b$$

Homogeneous Coordinates: Frame Transformation Matrix

Consider change of frames: $(v_1, v_2, v_3, P_0) \rightarrow (u_1, u_2, u_3, Q_0)$

For:
$$u = \begin{bmatrix} u_1 & u_2 & u_3 & Q_0 \end{bmatrix}^T$$
$$v = \begin{bmatrix} v_1 & v_2 & v_3 & P_0 \end{bmatrix}^T$$
$$u = Mv$$

The transformation matrix is given by:
$$M = \begin{bmatrix} \gamma_{11} & \gamma_{12} & \gamma_{13} & 0 \\ \gamma_{21} & \gamma_{22} & \gamma_{23} & 0 \\ \gamma_{31} & \gamma_{32} & \gamma_{33} & 0 \\ \gamma_{41} & \gamma_{42} & \gamma_{43} & 1 \end{bmatrix}$$

The values are: $(P_0 - Q_0) = \begin{bmatrix} \gamma_{41} & \gamma_{42} & \gamma_{43} & 1 \end{bmatrix}^T$

If the origin is not translated, then: $\gamma_{41} = \gamma_{42} = \gamma_{43} = 0$

Homogeneous Coordinates: Linear Transformations

In functional form, linear transformations are represented for points and vectors separately, as:

$$Q = T(P), \mathbf{v} = R(\mathbf{u})$$

If using homogeneous coordinates – the same transformation function can be used for both points and vectors.

$$\mathbf{q} = f(\mathbf{p}); \mathbf{v} = f(\mathbf{u})$$

Homogeneous Coordinates: Affine Transformations

Rotation by an angle θ about +z axis, scaling, translation.

$$\mathrm{R}: \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \rightarrow \begin{bmatrix} x\cos\theta - y\sin\theta \\ x\sin\theta + y\cos\theta \\ 1 \end{bmatrix}; \Rightarrow M_{R_z} = \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$\mathrm{S}: \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \rightarrow \begin{bmatrix} k_x x \\ k_y y \\ 1 \end{bmatrix}; \Rightarrow M_S = \begin{bmatrix} k_x & 0 & 0 \\ 0 & k_y & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$\mathrm{T}: \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \rightarrow \begin{bmatrix} x + t_x \\ y + t_y \\ 1 \end{bmatrix}; \Rightarrow M_T = \begin{bmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{bmatrix}$$

[Recap] 3D Transformation Matrices: Translation

$$\begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} \rightarrow \begin{bmatrix} x + \alpha_x \\ y + \alpha_y \\ z + \alpha_z \\ 1 \end{bmatrix}$$

$$\Rightarrow T(\alpha_x, \alpha_y, \alpha_z) = \begin{bmatrix} 1 & 0 & 0 & \alpha_x \\ 0 & 1 & 0 & \alpha_y \\ 0 & 0 & 1 & \alpha_z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$\Rightarrow T^{-1}(\alpha_x, \alpha_y, \alpha_z) = T(-\alpha_x, -\alpha_y, -\alpha_z)$$

$$= \begin{bmatrix} 1 & 0 & 0 & -\alpha_x \\ 0 & 1 & 0 & -\alpha_y \\ 0 & 0 & 1 & -\alpha_z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

[Recap] 3D Transformation Matrices: Scaling

Origin is the scaling-invariant (fixed) point.

$$\begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} \rightarrow \begin{bmatrix} \beta_x x \\ \beta_y y \\ \beta_z z \\ 1 \end{bmatrix}$$

$$\Rightarrow S(\beta_x, \beta_y, \beta_z) = \begin{bmatrix} \beta_x & 0 & 0 & 0 \\ 0 & \beta_y & 0 & 0 \\ 0 & 0 & \beta_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$\Rightarrow S^{-1}(\beta_x, \beta_y, \beta_z) = S(\frac{1}{\beta_x}, \frac{1}{\beta_y}, \frac{1}{\beta_z})$$

[Recap] 3D Transformation Matrices: Rotation

Transformation matrices for rotation about x-, y-, z-axes by an angle of θ_x, θ_y, θ_z, respectively: $R_x(\theta_x)$, $R_y(\theta_y)$, $R_z(\theta_z)$.

$$R_x(\theta_x) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\theta_x & -\sin\theta_x & 0 \\ 0 & \sin\theta_x & \cos\theta_x & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$R_y(\theta_y) = \begin{bmatrix} \cos\theta_y & 0 & \sin\theta_y & 0 \\ 0 & 1 & 0 & 0 \\ -\sin\theta_y & 0 & \cos\theta_y & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$R_z(\theta_z) = \begin{bmatrix} \cos\theta_z & -\sin\theta_z & 0 & 0 \\ \sin\theta_z & \cos\theta_z & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

[Recap] 3D Transformation Matrices: Rotation

Inverse of any rotation matrix: $R^{-1}(\theta) = R(-\theta)$

Since $\cos(-\theta) = \cos\theta$ and $\sin(-\theta) = -\sin\theta$

we get: Inverse = Transpose.

Thus, a rotation transformation matrix is an **orthogonal** matrix.

- Computation of inverse of rotation matrices thus becomes easier.

[Recap] 3D Transformation Matrices: Concatenation of Rotations

To construct desired rotation (about any arbitrary axis): Define origin as the fixed point, and implement sequence of rotations, such that:

$$R = R_i \cdot R_j \cdot R_k \ldots$$

Even in the concatenation of rotations, the concatenated matrix is orthogonal.

$$R^{-1} = R^T$$

Summary

- Parametrization
  - Interpolation Methods
  - Application: Color Mapping
- Coordinate Systems
  - Spaces
  - Affine Addition
- Homogeneous Coordinate Systems
  - Transformation Matrices
  - Need for a New Representation System
  - Solution
  - Applications