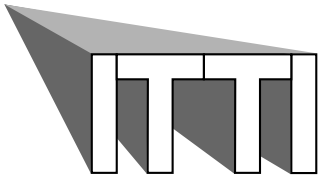


Computer Graphics and Visualisation

Geometry for Computer Graphics

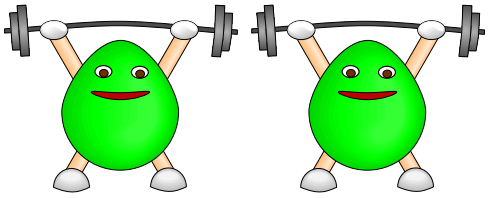
Overhead Projection (OHP) Overviews

Developed by

F Lin
K Wyrwas
J Irwin
C Lilley
W T Hewitt
T L J Howard

Computer Graphics Unit
Manchester Computing Centre
University of Manchester

Department of Computer Science
University of Manchester



THE UNIVERSITY *of* MANCHESTER

Produced by the ITTI Gravigs Project
Computer Graphics Unit
Manchester Computing Centre
MANCHESTER M13 9PL

First published by UCoSDA in May 1995

ISBN 1 85889 059 4

For further information on this and other modules please contact:
The ITTI Gravigs Project, Computer Graphics Unit, Manchester Computing Centre.
Tel: 0161 275 6095 Email: *gravigs@mcc.ac.uk*

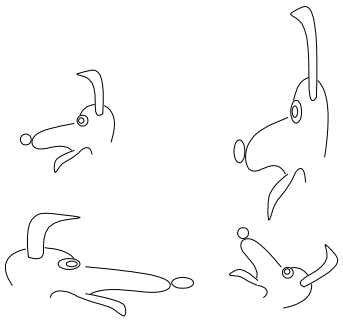
To order further copies of this or other modules please contact:
Mrs Jean Burgan, UCoSDA.
Tel: 0114 272 5248 Email: *j.burgan@sheffield.ac.uk*

These materials have been produced as part of the Information Technology Training Initiative, funded by the Information Systems Committee of the Higher Education Funding Councils.

The authors would like to thank Janet Edwards for her assistance in the preparation of these documents.

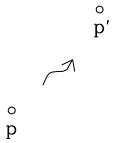
Printed by the Reprographics Department, Manchester Computing Centre from PostScript source supplied by the authors.

Two-Dimensional Transformations

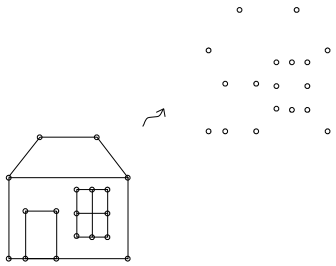


Two-Dimensional Transformations

- Transforming a point

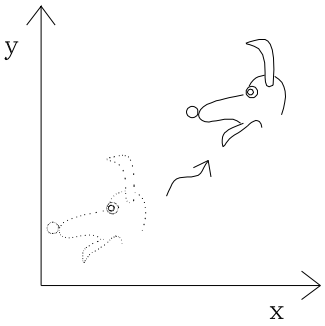


- Transforming an object



Types of Transformation

- Translation

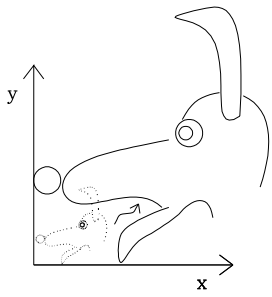


$$x' = x + t_x$$

$$y' = y + t_y$$

Types of Transformation

- Scaling



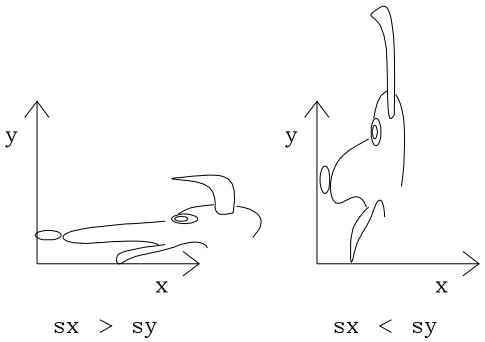
Sx=Sy=3

$$x' = x \cdot s_x$$

$$y' = y \cdot s_y$$

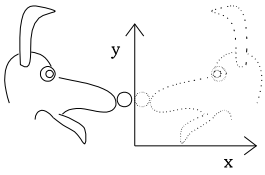
Symmetric vs asymmetric scaling

- Symmetric scaling: $s_x = s_y$
- Asymmetric scaling: $s_x > s_y$ or $s_x < s_y$

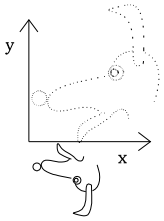


Scaling to achieve reflection

- Reflection in *y* axis: $s_x < 0$



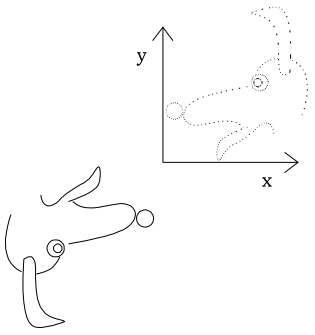
- Reflection in *x* axis: $s_y < 0$



Scaling to achieve reflection

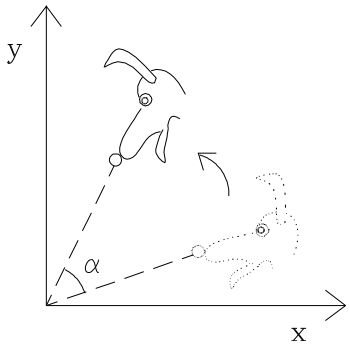
- Reflection in *x* and *y* axes:

  $s_y < 0$ *and* $s_x < 0$

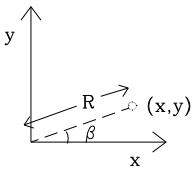


Types of Transformation

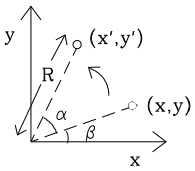
- Rotation



Rotation



$$x = R \cdot \cos\beta$$
$$y = R \cdot \sin\beta$$



$$x' = R \cdot \cos(\alpha + \beta)$$
$$y' = R \cdot \sin(\alpha + \beta)$$

Rotation

$$x' = R \cdot \cos(\alpha + \beta)$$
$$y' = R \cdot \sin(\alpha + \beta)$$

Expanding the formulae for cos(α+β) and sin(α+β):

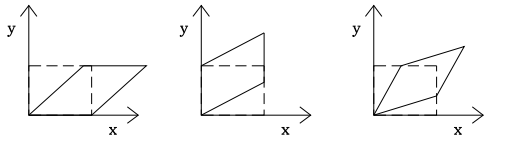
$$x' = R \cdot \cos\alpha \cdot \cos\beta - R \cdot \sin\alpha \cdot \sin\beta$$
$$y' = R \cdot \sin\alpha \cdot \cos\beta + R \cdot \sin\beta \cdot \cos\alpha$$

Substituting for $R \cdot \cos\beta$ and $R \cdot \sin\beta$:

$$x' = x \cdot \cos\alpha - y \cdot \sin\alpha$$
$$y' = x \cdot \sin\alpha + y \cdot \cos\alpha$$

Types of Transformation

- Shearing



shear in x shear in y shear in x and y

$$x' = x + y \cdot a$$
$$y' = y + x \cdot b$$

  - Shear in x: $a \neq 0$
  - Shear in y: $b \neq 0$
  - Shear in x and y: $a \neq 0$ and $b \neq 0$

Matrix Representation of Transformations

- Translate:

$$x' = x + t_x$$
$$y' = y + t_y$$

- Scale:

$$x' = x \cdot s_x$$
$$y' = y \cdot s_y$$

- Rotate:

$$x' = x \cdot \cos\alpha - y \cdot \sin\alpha$$
$$y' = x \cdot \sin\alpha + y \cdot \cos\alpha$$

Matrix Representation of Transformation

- Shear:

$$x' = x + y \cdot a$$
$$y' = y + x \cdot b$$

- In general:

$$x' = a \cdot x + b \cdot y + c$$
$$y' = d \cdot x + e \cdot y + f$$

Matrix Representation of Transformations

$$x' = a \cdot x + b \cdot y + c$$
$$y' = d \cdot x + e \cdot y + f$$

Using matrices:

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} a & b \\ d & e \end{bmatrix} \bullet \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} c \\ f \end{bmatrix}$$

Include a - f in one matrix:

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} a & b & c \\ d & e & f \end{bmatrix} \bullet \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

Matrix Representation of Transformations

$$x' = a \cdot x + b \cdot y + c$$
$$y' = d \cdot x + e \cdot y + f$$

Using a square matrix:

$$\begin{bmatrix} x' \\ y' \\ w' \end{bmatrix} = \begin{bmatrix} a & b & c \\ d & e & f \\ g & h & i \end{bmatrix} \bullet \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

Matrix Representation of Transformations

- Translate: $\begin{bmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{bmatrix}$

- Scale: $\begin{bmatrix} s_x & 0 & 0 \\ 0 & s_y & 0 \\ 0 & 0 & 1 \end{bmatrix}$

- Rotate: $\begin{bmatrix} \cos\alpha & -\sin\alpha & 0 \\ \sin\alpha & \cos\alpha & 0 \\ 0 & 0 & 1 \end{bmatrix}$

Matrix Representation of Transformations

- Shear: $\begin{bmatrix} 1 & a & 0 \\ b & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$
- Identity matrix:

$$\begin{bmatrix} x' \\ y' \\ w' \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \bullet \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

$$x' = x$$
$$y' = y$$
$$w' = 1$$

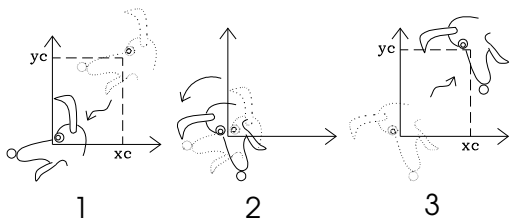
Combining Transformations

Can perform complex transformations by combining simple ones.

For example, rotating an object about its centre:

Combining Transformations

1. Translate by $(-x_c, -y_c)$
2. Rotate about the origin
3. Translate by (x_c, y_c)



Combining Transformations

Rotating an object about its centre:

1. Translate by $(-x_c,-y_c)$:

$$\begin{bmatrix} x_1 \\ y_1 \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & -x_c \\ 0 & 1 & -y_c \\ 0 & 0 & 1 \end{bmatrix} \bullet \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

2. Rotate about the origin:

$$\begin{bmatrix} x_2 \\ y_2 \\ 1 \end{bmatrix} = \begin{bmatrix} \cos\alpha & -\sin\alpha & 0 \\ \sin\alpha & \cos\alpha & 0 \\ 0 & 0 & 1 \end{bmatrix} \bullet \begin{bmatrix} x_1 \\ y_1 \\ 1 \end{bmatrix}$$

Combining Transformations

3. Translate by (x_c, y_c):

$$\begin{bmatrix} x_3 \\ y_3 \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & x_c \\ 0 & 1 & y_c \\ 0 & 0 & 1 \end{bmatrix} \bullet \begin{bmatrix} x_2 \\ y_2 \\ 1 \end{bmatrix}$$

Total transformation is:

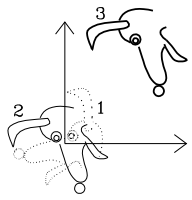
$$\begin{bmatrix} 1 & 0 & x_c \\ 0 & 1 & y_c \\ 0 & 0 & 1 \end{bmatrix} \bullet \begin{bmatrix} \cos\alpha & -\sin\alpha & 0 \\ \sin\alpha & \cos\alpha & 0 \\ 0 & 0 & 1 \end{bmatrix} \bullet \begin{bmatrix} 1 & 0 & -x_c \\ 0 & 1 & -y_c \\ 0 & 0 & 1 \end{bmatrix}$$

Ordering Transformations

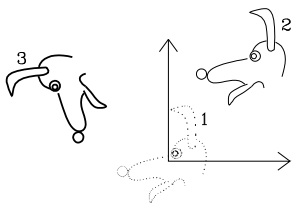
- Matrix multiplication is NOT commutative, $\mathbf{M}_1 \bullet \mathbf{M}_2 \neq \mathbf{M}_2 \bullet \mathbf{M}_1$
- Order of transformations is important

Ordering Transformations

- Rotate then translate



- Translate then rotate



Ordering Transformations

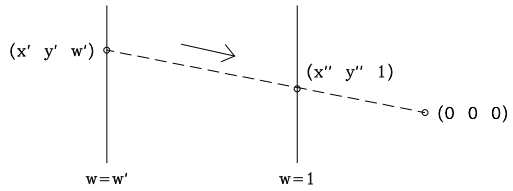
- POSTCONCATENATE $\mathbf{M}_2$ with $\mathbf{M}_1$:
  $\mathbf{p}' = \mathbf{M}_2 \bullet \mathbf{M}_1 \bullet \mathbf{p}$ ($\mathbf{M}_1$ applied first)
- PRECONCATENATE $\mathbf{M}_2$ with $\mathbf{M}_1$:
  $\mathbf{p}' = \mathbf{M}_1 \bullet \mathbf{M}_2 \bullet \mathbf{p}$ ($\mathbf{M}_2$ applied first)
- Premultiply ≡ Postconcatenate
- Postmultiply ≡ Preconcatenate

Homogeneous Coordinates

$$\begin{bmatrix} x' \\ y' \\ w' \end{bmatrix} = \begin{bmatrix} a & b & c \\ d & e & f \\ g & h & i \end{bmatrix} \bullet \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

- Point in **2D** space expressed in **3D homogeneous** coordinates
- If bottom row of matrix is (0 0 1), $w' = 1$
- If $w' \neq 1$ project point (x', y', w') onto plane $w=1$ by **homogeneous division** (using the origin as the centre of projection)

Homogeneous Coordinates



- Real world coordinates are x'' and y'' where

$$x'' = x'/w'$$
$$y'' = y'/w'$$

Homogeneous Coordinates

$$\begin{bmatrix} x' \\ y' \\ w' \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 4 \end{bmatrix} \bullet \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

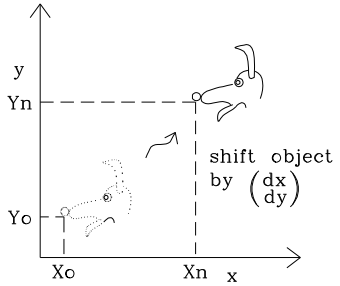
Perform homogeneous division to get "real world" coordinates:

$$x'' = x'/w' = x/4$$
$$y'' = y'/w' = y/4$$

Effect is OVERALL scaling.

Object vs Axis Transformation

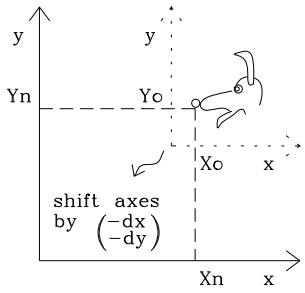
Object transformation



- Object transformed
- Axes fi xed

Object vs Axis Transformation

Axis transformation



- Object fi xed in space
- Axes transformed

Object vs Axis Transformation

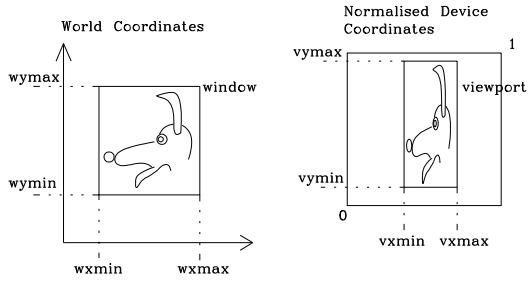
- Object translation by (d_x, d_y)

  ≡ Axis translation by $(-d_x, -d_y)$
- Object scale by (s_x, s_y)

  ≡ Axis scale by $(1/s_x, 1/s_y)$
- Object rotation by α

  ≡ Axis rotation by $-\alpha$

Normalization Transformation in GKS



1. Translate bottom left hand corner of window to origin:

$$\begin{bmatrix} x_1 \\ y_1 \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & -w_{xmin} \\ 0 & 1 & -w_{ymin} \\ 0 & 0 & 1 \end{bmatrix} \bullet \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

Normalization Transformation in GKS

2. Scale window to size of viewport:

$$\begin{bmatrix} x_2 \\ y_2 \\ 1 \end{bmatrix} = \begin{bmatrix} s_x & 0 & 0 \\ 0 & s_y & 0 \\ 0 & 0 & 1 \end{bmatrix} \bullet \begin{bmatrix} x_1 \\ y_1 \\ 1 \end{bmatrix}$$

where

$$s_x = \frac{V_{xmax} - V_{xmin}}{W_{xmax} - W_{xmin}}$$

$$s_y = \frac{V_{ymax} - V_{ymin}}{W_{ymax} - W_{ymin}}$$

Normalization Transformation in GKS

3. Translate to bottom left hand corner of viewport:

$$\begin{bmatrix} x_3 \\ y_3 \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & v_{xmin} \\ 0 & 1 & v_{ymin} \\ 0 & 0 & 1 \end{bmatrix} \bullet \begin{bmatrix} x_2 \\ y_2 \\ 1 \end{bmatrix}$$

Normalization transformation is:

$$\begin{bmatrix} 1 & 0 & v_{xmin} \\ 0 & 1 & v_{ymin} \\ 0 & 0 & 1 \end{bmatrix} \bullet \begin{bmatrix} s_x & 0 & 0 \\ 0 & s_y & 0 \\ 0 & 0 & 1 \end{bmatrix} \bullet \begin{bmatrix} 1 & 0 & -w_{xmin} \\ 0 & 1 & -w_{ymin} \\ 0 & 0 & 1 \end{bmatrix}$$

Two-Dimensional Transformations

- Different types: translation, scaling, rotation, shearing.
- Object vs axis transformations
- Matrix representation

$$\begin{bmatrix} x' \\ y' \\ w' \end{bmatrix} = \begin{bmatrix} a & b & c \\ d & e & f \\ g & h & i \end{bmatrix} \bullet \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

- Combine transformations by multiplying matrices
- Homogeneous division to get "real world" coordinates

$$x'' = x'/w', y'' = y'/w'$$

Three-Dimensional Transformations

Three-Dimensional Transformations

- For manipulating pictures (as in 2D)
- Help us to understand 3D shape
- Right-handed coordinate system

Homogeneous Coordinates

$$\begin{bmatrix} x' \\ y' \\ z' \\ w' \end{bmatrix} = \begin{bmatrix} a & b & c & d \\ e & f & g & h \\ i & j & k & l \\ m & n & o & p \end{bmatrix} \bullet \begin{bmatrix} x \\ y \\ z \\ w \end{bmatrix}$$

Obtain "real world" coordinates by homogeneous division:

$$x'' = x'/w'$$
$$y'' = y'/w'$$
$$z'' = z'/w'$$

Types of Three-Dimensional Transformation

- Translate:

$$\begin{bmatrix} 1 & 0 & 0 & t_x \\ 0 & 1 & 0 & t_y \\ 0 & 0 & 1 & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

- Scale:

$$\begin{bmatrix} s_x & 0 & 0 & 0 \\ 0 & s_y & 0 & 0 \\ 0 & 0 & s_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Types of Three-Dimensional Transformation

- Shear:

$$\begin{bmatrix} 1 & b & c & 0 \\ e & 1 & g & 0 \\ i & j & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Types of Three-Dimensional Transformation

- Rotation:

$$\begin{bmatrix} a & b & c & 0 \\ e & f & g & 0 \\ i & j & k & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

- Rotation about *x*-axis:

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\alpha & -\sin\alpha & 0 \\ 0 & \sin\alpha & \cos\alpha & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Types of Three-Dimensional Transformation

- Rotation about *y*-axis:

$$\begin{bmatrix} \cos\alpha & 0 & \sin\alpha & 0 \\ 0 & 1 & 0 & 0 \\ -\sin\alpha & 0 & \cos\alpha & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

- Rotation about *z*-axis:

$$\begin{bmatrix} \cos\alpha & -\sin\alpha & 0 & 0 \\ \sin\alpha & \cos\alpha & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Rotation about an arbitrary axis

1. Translate so axis of rotation passes through origin
2. Rotate so that axis of rotation coincides with one of the coordinate axes
3. Perform specified rotation about coordinate axis
4. Apply inverse rotation from (2)
5. Apply inverse translation from (1)

Perspective Transformations

$$\begin{bmatrix} x' \\ y' \\ z' \\ w' \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & \gamma & 1 \end{bmatrix} \bullet \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

$$= \begin{bmatrix} x \\ y \\ z \\ \gamma \cdot z + 1 \end{bmatrix}$$

Perspective Transformations

After homogeneous division

$$x'' = \frac{x}{\gamma \cdot z + 1}$$

$$y'' = \frac{y}{\gamma \cdot z + 1}$$

$$z'' = \frac{z}{\gamma \cdot z + 1}$$

As $z \to \infty$

$$x'' \to 0$$
$$y'' \to 0$$
$$z'' \to 1/\gamma$$

Perspective Transformations

- BEFORE transformation lines parallel to *z*-axis
- AFTER transformation lines pass through point $(0, 0, 1/\gamma)$
- Vanishing point is $(0, 0, 1/\gamma)$
- Eye point/centre of projection is $(0, 0, -1/\gamma)$
- One point perspective transformation

Perspective Transformations

- One point:

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ \alpha & 0 & 0 & 1 \end{bmatrix}$$

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & \beta & 0 & 1 \end{bmatrix}$$

Perspective Transformations

- Two point:

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ \alpha & 0 & \gamma & 1 \end{bmatrix}$$

- Three point:

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ \alpha & \beta & \gamma & 1 \end{bmatrix}$$

Perspective Transformations

Points behind the Eye Point

Perspective transformation with eye point at $(0, 0, c)$:

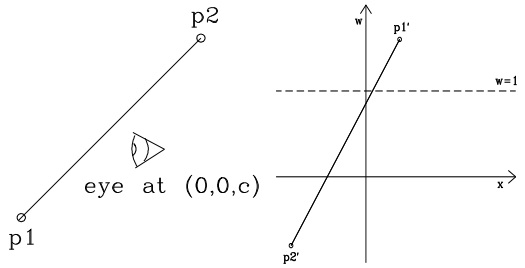
$$\begin{bmatrix} x' \\ y' \\ z' \\ w' \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & -1/c & 1 \end{bmatrix} \bullet \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

$$= \begin{bmatrix} x \\ y \\ z \\ (c - z)/c \end{bmatrix}$$

Perspective Transformations

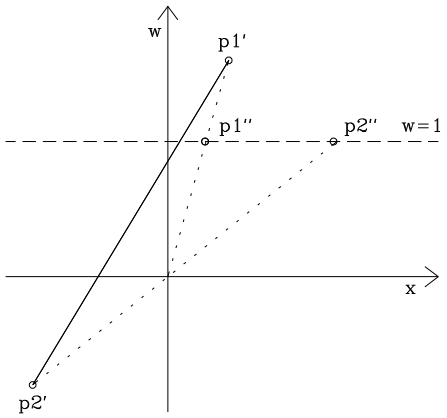
Points behind the Eye Point

- $w' > 0$ for $z < c$
- $w' = 0$ for $z = c$
- $w' < 0$ for $z > c$



Perspective Transformations

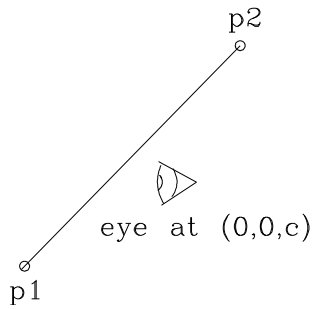
Points behind the Eye Point



Perspective Transformations

Points behind the Eye Point

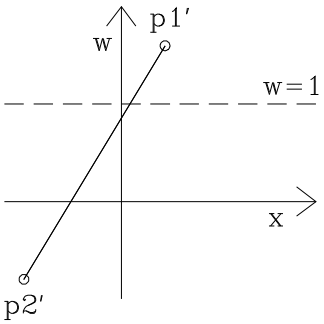
- Want to simulate what eye would see
- Clip BEFORE perspective transformation ($z \geq c$)



Perspective Transformations

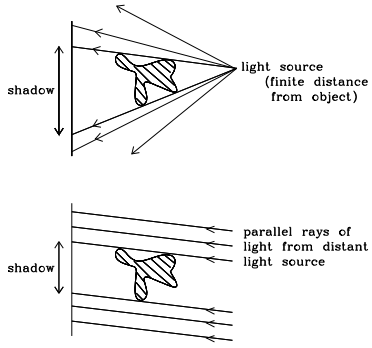
Points behind the Eye Point

- Clip after perspective transformation but BEFORE homogeneous division ($w \leq 0$)



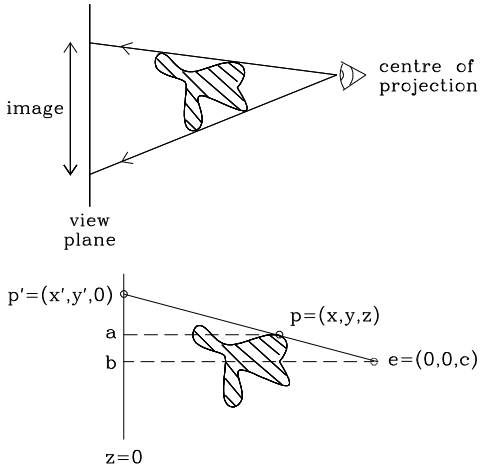
Projections

- Translation, rotation etc. all 3D→3D
- Projection is 3D→2D
- Similar to casting shadows



Perspective Projections

- Produces realistic images



Perspective Projections

- To calculate new x and y coordinates use similar triangles:

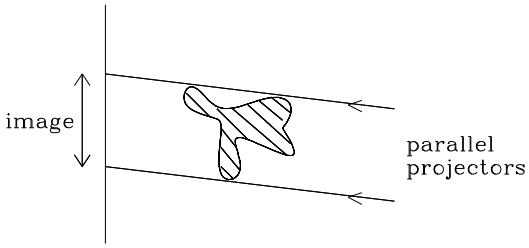
$$\frac{x'}{c} = \frac{x' - x}{z} \rightarrow x' = \frac{x}{1 - z/c}$$

similarly

$$y' = \frac{y}{1 - z/c}$$

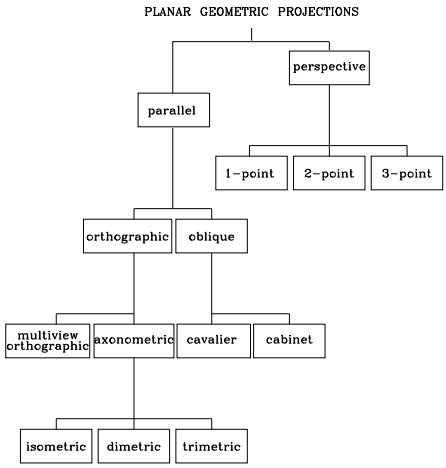
- Like perspective transformation

Parallel Projections

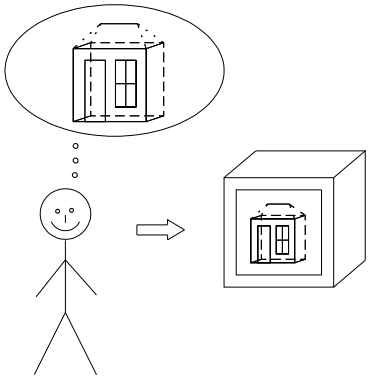


- Perspective projection with centre of projection at ∞
- Less realistic
- Retains size information for measurements

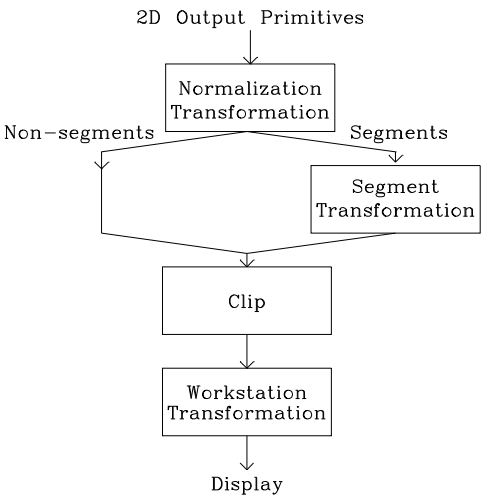
Classifi cation of Projections



Transformations and Viewing in GKS-3D and PHIGS



GKS Output Pipeline

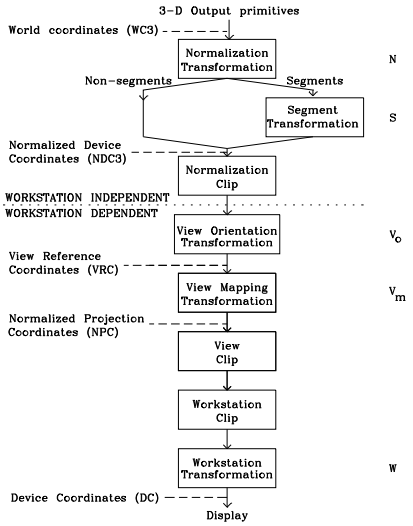


GKS Output Pipeline

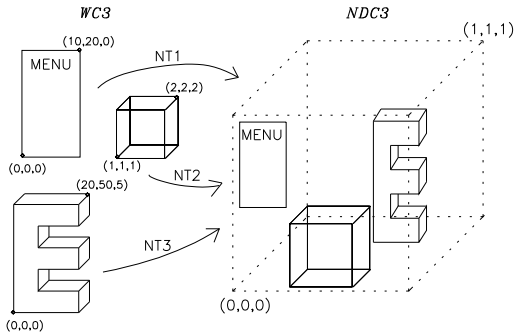
- Converts user-defi ned graphical information to device coordinates for display.
- GKS-3D and PHIGS need also to perform viewing transformations.

GKS-3D Output Pipeline

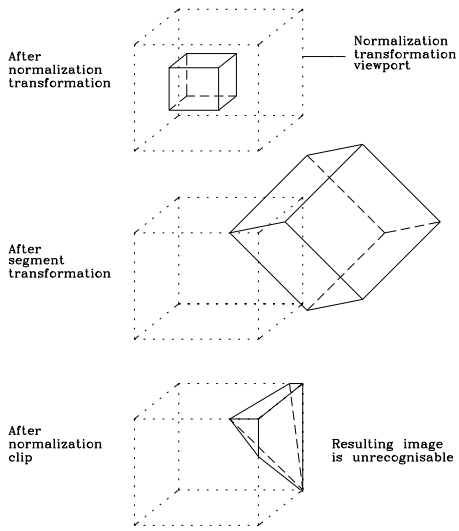
$$\mathbf{p}_d = \mathbf{W} \bullet \mathbf{V}_m \bullet \mathbf{V}_o \bullet \mathbf{S} \bullet \mathbf{N} \bullet \mathbf{p}_w$$



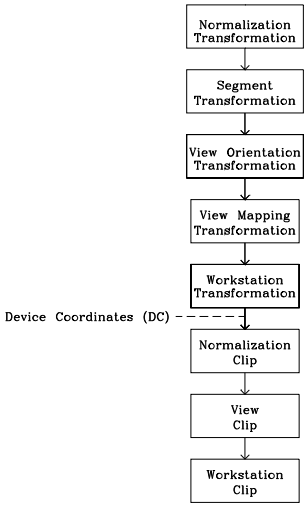
Normalization Transformations in GKS-3D



Segment Transformation and Normalization Clip

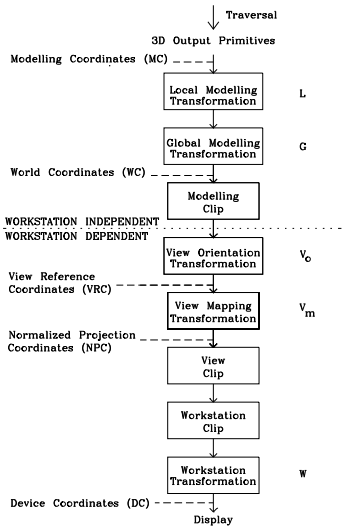


Possible Implementation of the GKS-3D Pipeline

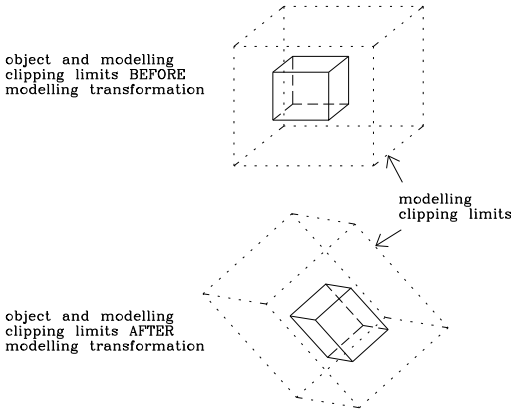


PHIGS Output Pipeline

$$p_d = W \bullet V_m \bullet V_o \bullet G \bullet L \bullet p_m$$

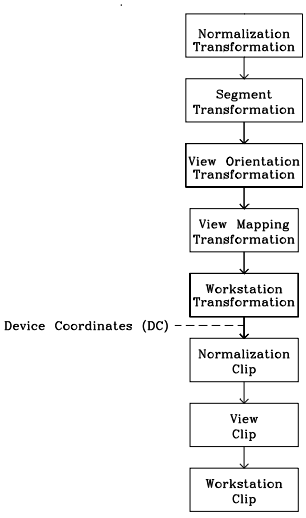


Modelling Clip

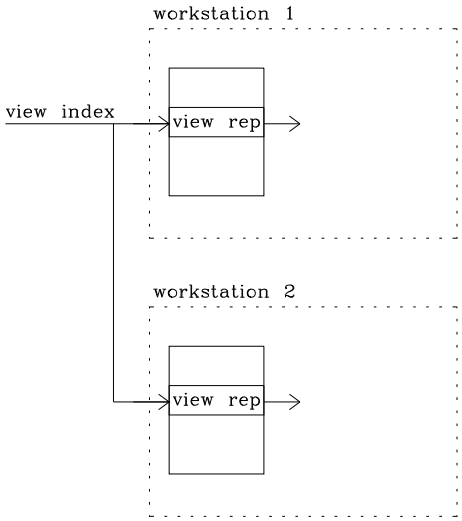


- Clipping limits are affected by modelling transformation

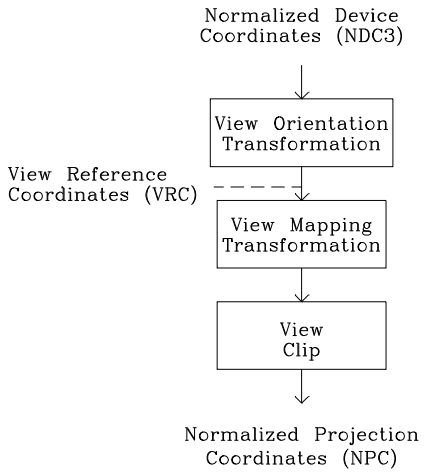
Possible Implementation of the PHIGS Pipeline



Different Views on Different Workstations



The Viewing Pipeline



The Viewing Pipeline

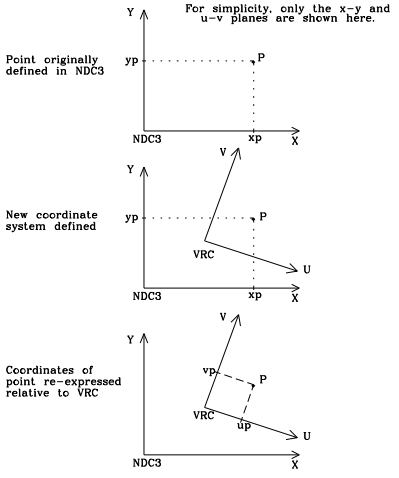
- View orientation and mapping transformations specified as 4×4 matrices
- Gives maximum flexibility to the user
- Hard to set the matrices!
- Viewing utility routines
- EVALUATE VIEW ORIENTATION MATRIX
- EVALUATE VIEW MAPPING MATRIX
- Viewing model

Model for View Orientation

- View orientation transformation maps from NDC3 to VRC
- VRC is an intermediate coordinate system
- VRC defined so that view plane is parallel to the VRC *xy*-plane

View Orientation Transformation

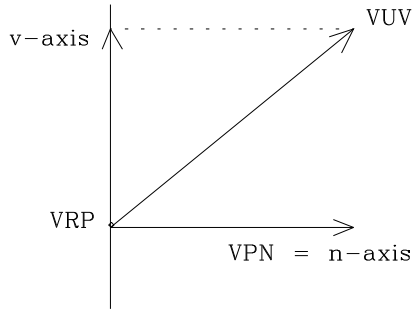
AXIS transformation



EVALUATE VIEW ORIENTATION MATRIX

- Parameters defi ne position and orientation of VRC with respect to NDC3
- View reference point (VRP) - origin of VRC
- View plane normal (VPN) - defi nes n-axis of VRC
- View up vector (VUV) - defi nes v-axis of VRC

EVALUATE VIEW ORIENTATION MATRIX



Deriving the View Orientation Matrix

- Inverse of axis transformation = equivalent object transformation
- Inverse view orientation transformation maps unit vectors on NDC3 axes onto VRC axes
- Rotate vectors to align with VRC axes (**R**)
- Translate by VRP, $\mathbf{T} = \begin{bmatrix} 1 & 0 & 0 & VPR_x \\ 0 & 1 & 0 & VPR_y \\ 0 & 0 & 1 & VPR_z \\ 0 & 0 & 0 & 1 \end{bmatrix}$

Deriving the View Orientation Matrix

It can be shown that

$$\mathbf{R} = \begin{bmatrix} u_x & v_x & n_x & 0 \\ u_y & v_y & n_y & 0 \\ u_z & v_z & n_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

where (u_x, u_y, u_z) are the components of **u** in world-coordinates and similarly for **v** and **n**

Deriving the View Orientation Matrix

- Inverse view orientation transformation = $\mathbf{T} \bullet \mathbf{R}$
- View orientation transformation

$$= \mathbf{R}^{-1} \bullet \mathbf{T}^{-1}$$

$$= \begin{bmatrix} u_x & u_y & u_z & 0 \\ v_x & v_y & v_z & 0 \\ n_x & n_y & n_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \bullet \begin{bmatrix} 1 & 0 & 0 & -VPR_x \\ 0 & 1 & 0 & -VPR_y \\ 0 & 0 & 1 & -VPR_z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Model for View Mapping

- Creates view required
- Parallel or perspective projection retaining third dimension for HLHSR
- Maps contents of view volume onto projection viewport in NPC

EVALUATE VIEW MAPPING MATRIX

Parameters describe:

- View volume in VRC
- View plane distance (VPD)
- Front plane distance (FPD)
- Back plane distance (BPD)
- View window limits (U_{min}, U_{max}, V_{min}, V_{max})
- Projection reference point (PRP)
- Projection type: PARALLEL or PERSPECTIVE
- Projection viewport in NPC

EVALUATE VIEW MAPPING MATRIX

- Creating the view
- Parallel or perspective projection on *x* and *y* (defi ned by PRP and view plane)
- Maintains relative *z* information for HLHSR
- Mapping contents of view volume to projection viewport
- After viewing transformation, view volume is a cuboid
- Window to viewport mapping

Using the Viewing Model

Orthographic parallel projections

- Projectors perpendicular to view plane
- Projector direction defi ned by vector from PRP to centre of view window

$$PRP_x - (U_{min} + U_{max})/2 = 0$$

$$PRP_y - (V_{min} + V_{max})/2 = 0$$

Using the Viewing Model

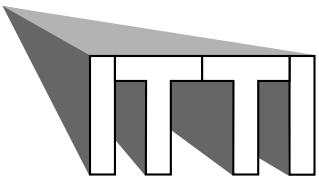
Perspective projections

- Lines not parallel to the view plane converge to a vanishing point
- One point
  - 1 vanishing point → 1 axis not parallel to view plane
  - view plane normal is one of $(a, 0, 0)$, $(0, b, 0)$, $(0, 0, c)$
- Two point
  - 2 vanishing points → 2 axes not parallel to view plane
  - view plane normal is one of $(a, b, 0)$, $(0, b, c)$, $(a, 0, c)$

Using the Viewing Model

Perspective projections

- Three point
  - 3 vanishing points → all 3 axes not parallel to view plane
  - view plane normal is (a, b, c)



These materials have been produced as part of the Information Technology Training Initiative, funded by the Information Systems Committee of the Higher Education Funding Councils.

For further information on this and other modules please contact:

The ITTI Gravigs Project, Computer Graphics Unit, Manchester Computing Centre.
Tel: 0161 275 6095 Email: *gravigs@mcc.ac.uk*

To order additional copies of this or other modules please contact:

Mrs Jean Burgan, UCoSDA.
Tel:0114 272 5248 Email: *j.burgan@sheffield.ac.uk*

ISBN 1 85889 059 4