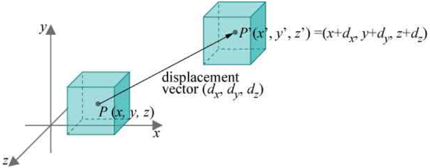
Transformations

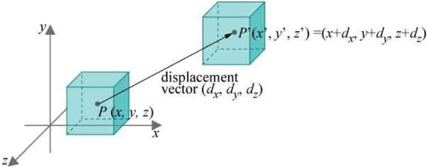
Moving Objects

We need to move our objects in 3D space.



Moving Objects

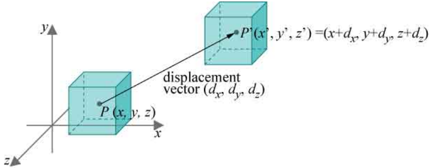
We need to move our objects in 3D space.



- An object/model (box, car, building, character,. . .) is defined in one position (often centered around origo). Will be needed in another position in the scene.

Moving Objects

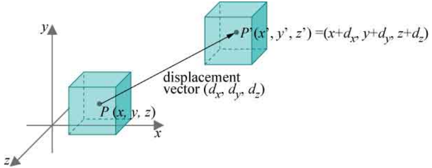
We need to move our objects in 3D space.



- An object/model (box, car, building, character,...) is defined in one position (often centered around origo). Will be needed in another position in the scene.
- Maybe in several places in one scene (town with houses and cars).

Moving Objects

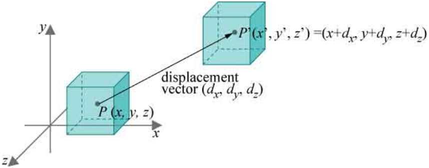
We need to move our objects in 3D space.



- An object/model (box, car, building, character,. . .) is defined in one position (often centered around origo). Will be needed in another position in the scene.
- Maybe in several places in one scene (town with houses and cars).
- Maybe in different places in different scenes/frames (animation).

Moving Objects

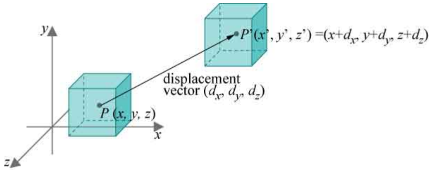
We need to move our objects in 3D space.



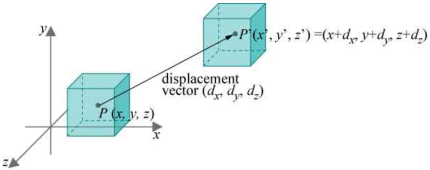
- An object/model (box, car, building, character,...) is defined in one position (often centered around origo). Will be needed in another position in the scene.
- Maybe in several places in one scene (town with houses and cars).
- Maybe in different places in different scenes/frames (animation).

Move model $\Leftrightarrow$ move triangles $\Leftrightarrow$ move points (vertices) $\Leftrightarrow f : \mathbb{R}^3 \rightarrow \mathbb{R}^3$

Translation

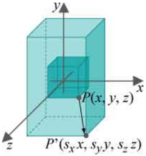


Translation

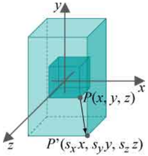


$$f\begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{pmatrix} x + d_x \\ y + d_y \\ z + d_z \end{pmatrix}$$

Scaling

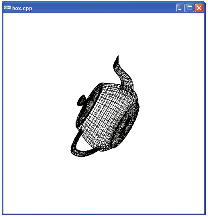


Scaling



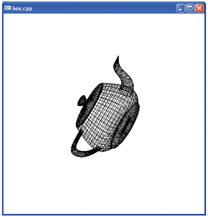
$$f\begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{pmatrix} s_x \cdot x \\ s_y \cdot y \\ s_z \cdot z \end{pmatrix}$$

Rotation



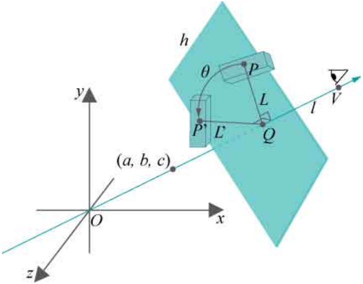
Euler [1775]: for enhver orientering af en model findes der en linie l gennem (0,0,0) og en vinkel ϕ, således at denne orientering opnås ved at rotere ϕ grader om l.

Rotation

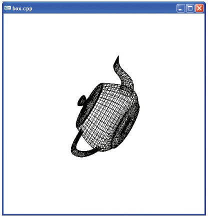


Euler [1775]: for enhver orientering af en model findes der en linie l gennem (0,0,0) og en vinkel ϕ, således at denne orientering opnås ved at rotere ϕ grader om l.

Rotation around line through origin:

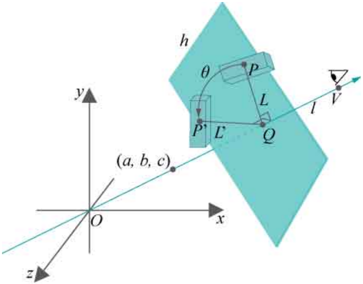


Rotation



Euler [1775]: for enhver orientering af en model findes der en linie l gennem (0,0,0) og en vinkel ϕ, således at denne orientering opnås ved at rotere ϕ grader om l.

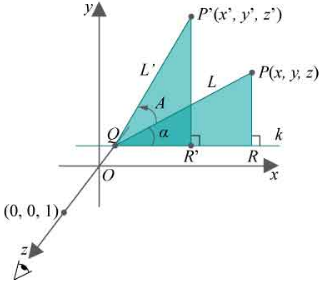
Rotation around line through origin:



$$f \begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{pmatrix} ? \\ ? \\ ? \end{pmatrix}$$

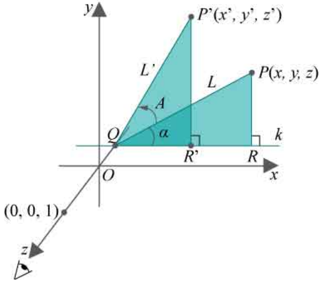
Rotation

Simpler case: Rotation around z-axis.



Rotation

Simpler case: Rotation around z-axis.



From formula for rotation in 2D:

$$f\begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{pmatrix} x\cos\phi - y\sin\phi \\ x\sin\phi + y\cos\phi \\ z \end{pmatrix}$$

Rotation

Similar: Rotation around x-axis and y-axis.

$$f\begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{pmatrix} x \\ y\cos\phi - z\sin\phi \\ y\sin\phi + z\cos\phi \end{pmatrix}$$

$$f\begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{pmatrix} z\sin\phi + x\cos\phi \\ y \\ z\cos\phi - x\sin\phi \end{pmatrix}$$

Euler

Theorem (Euler, 1775): any rotation with axis through origo can be created as three succesive rotations around the three coordinate axes.

The angles of the three coordinate axis rotations are called Euler angles.

Using Euler angles to specify generic rotations is often intuitive, but also has drawbacks. We will return to that later.

Matrices

Move model $\Leftrightarrow$ move triangles $\Leftrightarrow$ move points (vertices) $\Leftrightarrow f : \mathbb{R}^3 \to \mathbb{R}^3$

Matrices

Move model $\Leftrightarrow$ move triangles $\Leftrightarrow$ move points (vertices) $\Leftrightarrow$ $f : \mathbb{R}^3 \to \mathbb{R}^3$

Any matrix induces a (linear) funktion $f : \mathbb{R}^3 \to \mathbb{R}^3$:

$$f \begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix} \cdot \begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{pmatrix} 1x + 2y + 3z \\ 4x + 5y + 6z \\ 7x + 8y + 9z \end{pmatrix}$$

Matrices

Move model $\Leftrightarrow$ move triangles $\Leftrightarrow$ move points (vertices) $\Leftrightarrow f : \mathbb{R}^3 \to \mathbb{R}^3$

Any matrix induces a (linear) funktion $f : \mathbb{R}^3 \to \mathbb{R}^3$:

$$f\begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix} \cdot \begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{pmatrix} 1x + 2y + 3z \\ 4x + 5y + 6z \\ 7x + 8y + 9z \end{pmatrix}$$

Recall: Matrix multiplication is associative: $A \cdot (B \cdot C) = (A \cdot B) \cdot C$.

Matrices

Move model $\Leftrightarrow$ move triangles $\Leftrightarrow$ move points (vertices) $\Leftrightarrow f : \mathbb{R}^3 \to \mathbb{R}^3$

Any matrix induces a (linear) funktion $f : \mathbb{R}^3 \to \mathbb{R}^3$:

$$f\begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix} \cdot \begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{pmatrix} 1x + 2y + 3z \\ 4x + 5y + 6z \\ 7x + 8y + 9z \end{pmatrix}$$

Recall: Matrix multiplication is associative: $A \cdot (B \cdot C) = (A \cdot B) \cdot C$.
Hence:

$$A \cdot (B \cdot (C \cdot (E \cdot (F \cdot \begin{pmatrix} x \\ y \\ z \end{pmatrix})))) = ((((A \cdot B) \cdot C) \cdot E) \cdot F) \cdot \begin{pmatrix} x \\ y \\ z \end{pmatrix}$$

Matrices

Move model $\Leftrightarrow$ move triangles $\Leftrightarrow$ move points (vertices) $\Leftrightarrow f : \mathbb{R}^3 \to \mathbb{R}^3$

Any matrix induces a (linear) funktion $f : \mathbb{R}^3 \to \mathbb{R}^3$:

$$f\begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix} \cdot \begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{pmatrix} 1x + 2y + 3z \\ 4x + 5y + 6z \\ 7x + 8y + 9z \end{pmatrix}$$

Recall: Matrix multiplication is associative: $A \cdot (B \cdot C) = (A \cdot B) \cdot C$.
Hence:

$$A \cdot (B \cdot (C \cdot (E \cdot (F \cdot \begin{pmatrix} x \\ y \\ z \end{pmatrix})))) = ((((A \cdot B) \cdot C) \cdot E) \cdot F) \cdot \begin{pmatrix} x \\ y \\ z \end{pmatrix}$$

Saves calculations: 3D object = many triangles = many points. All points go through the same sequence of transformations (moves). Calculate the matrix product once.

Matrices

Move model $\Leftrightarrow$ move triangles $\Leftrightarrow$ move points (vertices) $\Leftrightarrow$ $f : \mathbb{R}^3 \to \mathbb{R}^3$

Any matrix induces a (linear) funktion $f : \mathbb{R}^3 \to \mathbb{R}^3$:

$$f\begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix} \cdot \begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{pmatrix} 1x + 2y + 3z \\ 4x + 5y + 6z \\ 7x + 8y + 9z \end{pmatrix}$$

Recall: Matrix multiplication is associative: $A \cdot (B \cdot C) = (A \cdot B) \cdot C$.
Hence:

$$A \cdot (B \cdot (C \cdot (E \cdot (F \cdot \begin{pmatrix} x \\ y \\ z \end{pmatrix})))) = ((((A \cdot B) \cdot C) \cdot E) \cdot F) \cdot \begin{pmatrix} x \\ y \\ z \end{pmatrix}$$

Saves calculations: 3D object = many triangles = many points. All points go through the same sequence of transformations (moves). Calculate the matrix product once.

Question: can all our needed transformations be expressed as matrices?

Transformations as Matrices

Transformations as Matrices

- Scaling

$$f\begin{pmatrix}x\\y\\z\end{pmatrix}=\begin{pmatrix}s_1x\\s_2y\\s_3z\end{pmatrix}=\begin{bmatrix}s_1&0&0\\0&s_2&0\\0&0&s_3\end{bmatrix}\cdot\begin{pmatrix}x\\y\\z\end{pmatrix}$$

Transformations as Matrices

- Scaling

$$f\begin{pmatrix}x\\y\\z\end{pmatrix} = \begin{pmatrix}s_1 x\\s_2 y\\s_3 z\end{pmatrix} = \begin{bmatrix}s_1 & 0 & 0\\0 & s_2 & 0\\0 & 0 & s_3\end{bmatrix}\cdot\begin{pmatrix}x\\y\\z\end{pmatrix}$$

- Rotation angle ϕ around the z-axis

$$f\begin{pmatrix}x\\y\\z\end{pmatrix} = \begin{pmatrix}x\cos\phi - y\sin\phi\\x\sin\phi + y\cos\phi\\z\end{pmatrix} = \begin{bmatrix}\cos\phi & -\sin\phi & 0\\\sin\phi & \cos\phi & 0\\0 & 0 & 1\end{bmatrix}\cdot\begin{pmatrix}x\\y\\z\end{pmatrix}$$

Transformations as Matrices

- Scaling

$$f\begin{pmatrix}x\\y\\z\end{pmatrix}=\begin{pmatrix}s_1x\\s_2y\\s_3z\end{pmatrix}=\begin{bmatrix}s_1&0&0\\0&s_2&0\\0&0&s_3\end{bmatrix}\cdot\begin{pmatrix}x\\y\\z\end{pmatrix}$$

- Rotation angle ϕ around the z-axis

$$f\begin{pmatrix}x\\y\\z\end{pmatrix}=\begin{pmatrix}x\cos\phi-y\sin\phi\\x\sin\phi+y\cos\phi\\z\end{pmatrix}=\begin{bmatrix}\cos\phi&-\sin\phi&0\\\sin\phi&\cos\phi&0\\0&0&1\end{bmatrix}\cdot\begin{pmatrix}x\\y\\z\end{pmatrix}$$

- Translation?

$$f\begin{pmatrix}x\\y\\z\end{pmatrix}=\begin{pmatrix}x+x_0\\y+y_0\\z+z_0\end{pmatrix}=\begin{bmatrix}?&?&?\\?&?&?\\?&?&?\end{bmatrix}\cdot\begin{pmatrix}x\\y\\z\end{pmatrix}$$

Transformations as Matrices

- Scaling

$$f\begin{pmatrix}x\\y\\z\end{pmatrix}=\begin{pmatrix}s_1x\\s_2y\\s_3z\end{pmatrix}=\begin{bmatrix}s_1&0&0\\0&s_2&0\\0&0&s_3\end{bmatrix}\cdot\begin{pmatrix}x\\y\\z\end{pmatrix}$$

- Rotation angle ϕ around the z-axis

$$f\begin{pmatrix}x\\y\\z\end{pmatrix}=\begin{pmatrix}x\cos\phi-y\sin\phi\\x\sin\phi+y\cos\phi\\z\end{pmatrix}=\begin{bmatrix}\cos\phi&-\sin\phi&0\\\sin\phi&\cos\phi&0\\0&0&1\end{bmatrix}\cdot\begin{pmatrix}x\\y\\z\end{pmatrix}$$

- Translation?

$$f\begin{pmatrix}x\\y\\z\end{pmatrix}=\begin{pmatrix}x+x_0\\y+y_0\\z+z_0\end{pmatrix}=\begin{bmatrix}?&?&?\\?&?&?\\?&?&?\end{bmatrix}\cdot\begin{pmatrix}x\\y\\z\end{pmatrix}$$

No. For (non-trivial) translation we have $f(0,0,0)\neq(0,0,0)$, but all functions induced by matrices have $f(0,0,0)=(0,0,0)$.

Homogeneous Coordinates

Go to 4D:

$$\begin{pmatrix} x \\ y \\ z \end{pmatrix} \rightarrow \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix}$$

Homogeneous Coordinates

Go to 4D:

$$\begin{pmatrix} x \\ y \\ z \end{pmatrix} \to \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix}$$

And back:

$$\begin{pmatrix} x \\ y \\ z \\ w \end{pmatrix} \to \begin{pmatrix} x/w \\ y/w \\ z/w \end{pmatrix}$$

Homogeneous Coordinates

Translations (in 3D) can now be expressed as matrix multiplication:

$$\begin{bmatrix} 1 & 0 & 0 & x_0 \\ 0 & 1 & 0 & y_0 \\ 0 & 0 & 1 & z_0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix} = \begin{pmatrix} x + x_0 \\ y + y_0 \\ z + z_0 \\ 1 \end{pmatrix}$$

Homogeneous Coordinates

Translations (in 3D) can now be expressed as matrix multiplication:

$$\begin{bmatrix} 1 & 0 & 0 & x_0 \\ 0 & 1 & 0 & y_0 \\ 0 & 0 & 1 & z_0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix} = \begin{pmatrix} x + x_0 \\ y + y_0 \\ z + z_0 \\ 1 \end{pmatrix}$$

All 3x3 matrices are still available (incl. scaling and rotation):

$$\begin{bmatrix} 1 & 2 & 3 & 0 \\ 4 & 5 & 6 & 0 \\ 7 & 8 & 9 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix} = \begin{pmatrix} 1x + 2y + 3z \\ 4x + 5y + 6z \\ 7x + 8y + 9z \\ 1 \end{pmatrix}$$

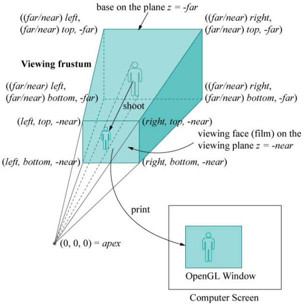
Projection

Projection to screen: $f : \mathbb{R}^3 \to \mathbb{R}^2$.

Projection

Projection to screen: $f : \mathbb{R}^3 \to \mathbb{R}^2$.

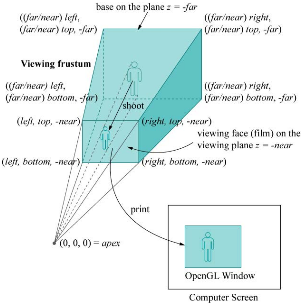
Prespective projection:



Projection

Projection to screen: $f : \mathbb{R}^3 \to \mathbb{R}^2$.

Prespective projection:



Expressed as 4x4 matrix multiplication ($d = -$near):

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 1/d & 0 \end{bmatrix} \cdot \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix} = \begin{pmatrix} x \\ y \\ z \\ z/d \end{pmatrix} \to \begin{pmatrix} xd/z \\ yd/z \\ d \end{pmatrix}$$

Transformations in OpenGL

OpenGL uses 4x4-matrices/homogeneous coordinates internally. Matrices are normally created by more intuitive commands:

- `glTranslatef(dx,dy,dz)`
- `glScalef(sx,sy,sz)`
- `glRotatef(angle,ax,ay,az)`

Transformations in OpenGL

OpenGL uses 4x4-matrices/homogeneous coordinates internally. Matrices are normally created by more intuitive commands:

- `glTranslatef(dx,dy,dz)`
- `glScalef(sx,sy,sz)`
- `glRotatef(angle,ax,ay,az)`

Each command generates the corresponding matrix, and right-multiplies it on the current matrix.

So last transformaton specified in code is first applied to vertices.

Cf. the math notation $f(g(h(x)))$ (where h is applied first to x, then g, then f).

Transformations in OpenGL

OpenGL uses 4x4-matrices/homogeneous coordinates internally. Matrices are normally created by more intuitive commands:

- `glTranslatef(dx,dy,dz)`
- `glScalef(sx,sy,sz)`
- `glRotatef(angle,ax,ay,az)`

Each command generates the corresponding matrix, and right-multiplies it on the current matrix.

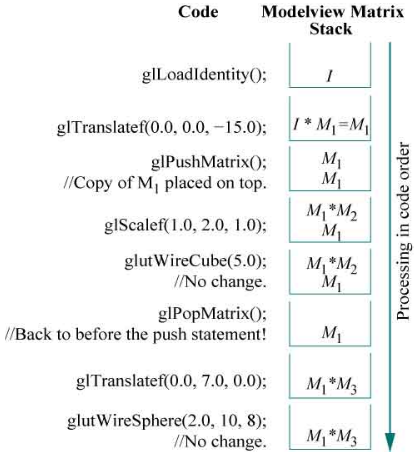
So last transformaton specified in code is first applied to vertices.

Cf. the math notation $f(g(h(x)))$ (where h is applied first to x, then g, then f).

There is a current matrix for model-view transformations, for projections, and for textures. Each has a stack.

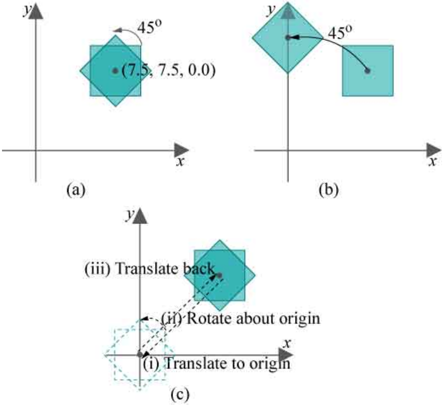
(Note: legacy code. In shader-based code, similar techniques are used.)

Matrix Stack



"The Trick"

Note that rotations are always around origo. To get the effect of a), a single rotation will not work, but will give the effect of b). Instead, do as in c) (translate to origo, rotate, translate back).



This kind of thinking is referred to as "the trick" in the textbook. Similar considerations relate to scaling.