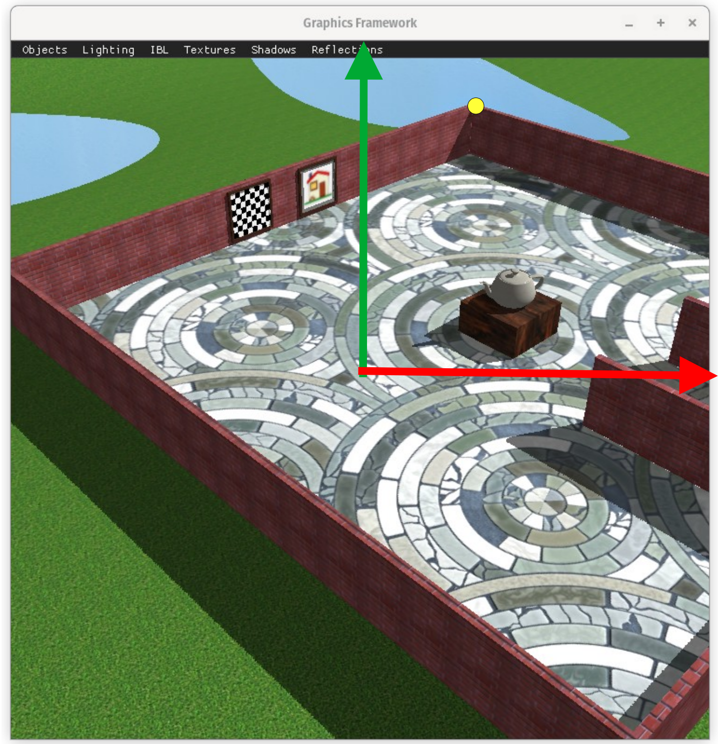
Transformations
(from Linear Algebra)

WORLD COORDINATE SYSTEM
Yellow dot at approx (8,8,2)



SCREEN COORDINATE SYSTEM
Yellow dot at approx (0.2, 0.9, ?)

Projections *world* → *screen*

A good start might be to think **Linear Transformations**:
find a 3x3 matrix that does this

$$\begin{bmatrix} 0.2 \\ 0.9 \\ ? \end{bmatrix} = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix} \begin{bmatrix} 8 \\ 8 \\ 2 \end{bmatrix}$$

and similarly for all points in the scene.

This fails for several of reasons:

- **Perspective Projection** is not (quite) a Linear Transformation
- Linear Transformations **must** map the origin to the origin. We need more than that.

Solution:

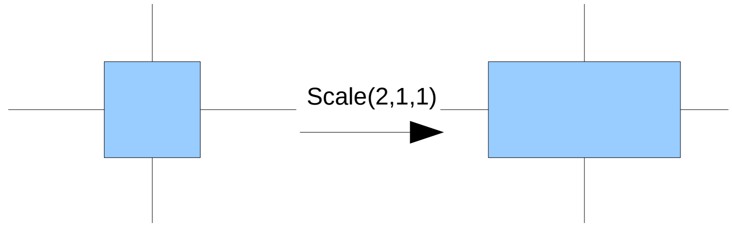
Go up to 4 dimensions! $$\begin{bmatrix} 0.2 \\ 0.9 \\ ? \\ ? \end{bmatrix} = \begin{bmatrix} a_{11} & a_{12} & a_{13} & ? \\ a_{21} & a_{22} & a_{23} & ? \\ a_{31} & a_{32} & a_{33} & ? \\ ? & ? & ? & ? \end{bmatrix} \begin{bmatrix} 8 \\ 8 \\ 2 \\ ? \end{bmatrix}$$

Called **homogeneous** coordinates/vectors/transformations/projections.

The primitive building block transformations

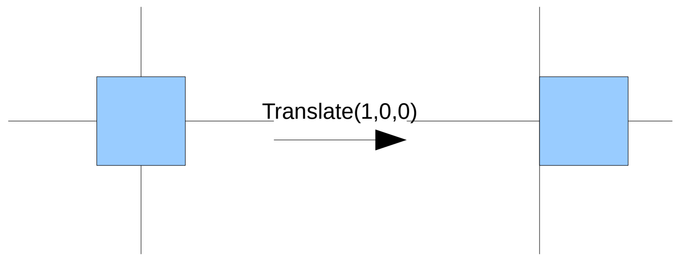
Scale transformation

$$S(s_X, s_Y, s_Z) = \begin{bmatrix} s_x & 0 & 0 & 0 \\ 0 & s_y & 0 & 0 \\ 0 & 0 & s_Z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$



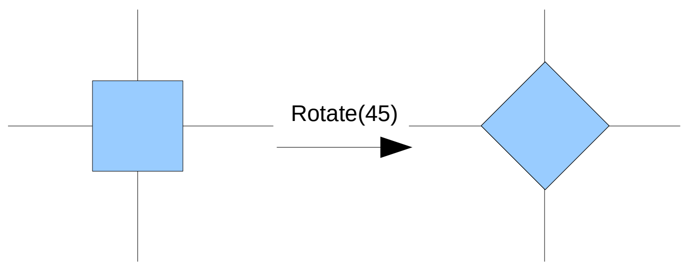
Translate transformation

$$T(t_X, t_Y, t_Z) = \begin{bmatrix} 1 & 0 & 0 & t_x \\ 0 & 1 & 0 & t_y \\ 0 & 0 & 1 & t_Z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$



3 Rotate transformations

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



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

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

Algorithm for building all three rotations matrices:

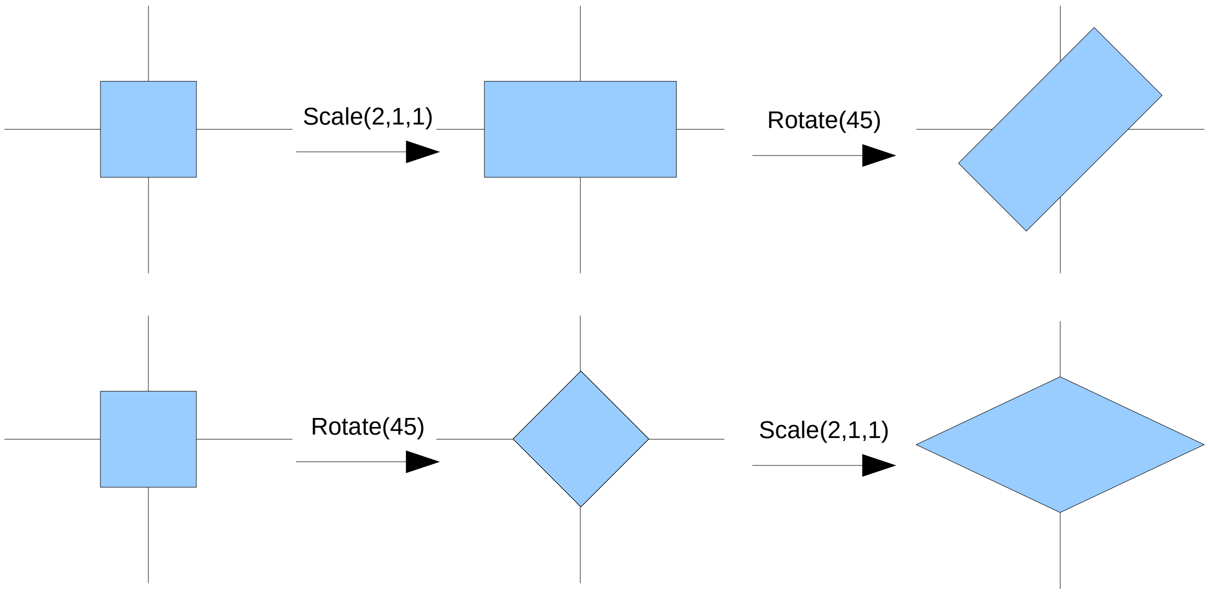
Let the rotation axis be specified with i=0,1,2 for the X,Y,Z axes (respectively). Compute j = (i+1) mod 3 and k = (i+2) mod 3 and build the matrix like this:

Programming note: Math notation is in row-major form, but GLM uses column-major matrices

$$R_{jj} = \cos\theta$$
$$R_{kk} = \cos\theta$$
$$R_{ii} = R_{33} = 1$$
$$R_{jk} = -\sin\theta \quad \textit{Programming note}: R[k][j] = -\sin\theta$$
$$R_{kj} = \sin\theta \quad \textit{Programming note}: R[j][k] = \sin\theta$$
$$\textit{others} = 0$$

3D Transformation Interactions

Scale vs Rotate

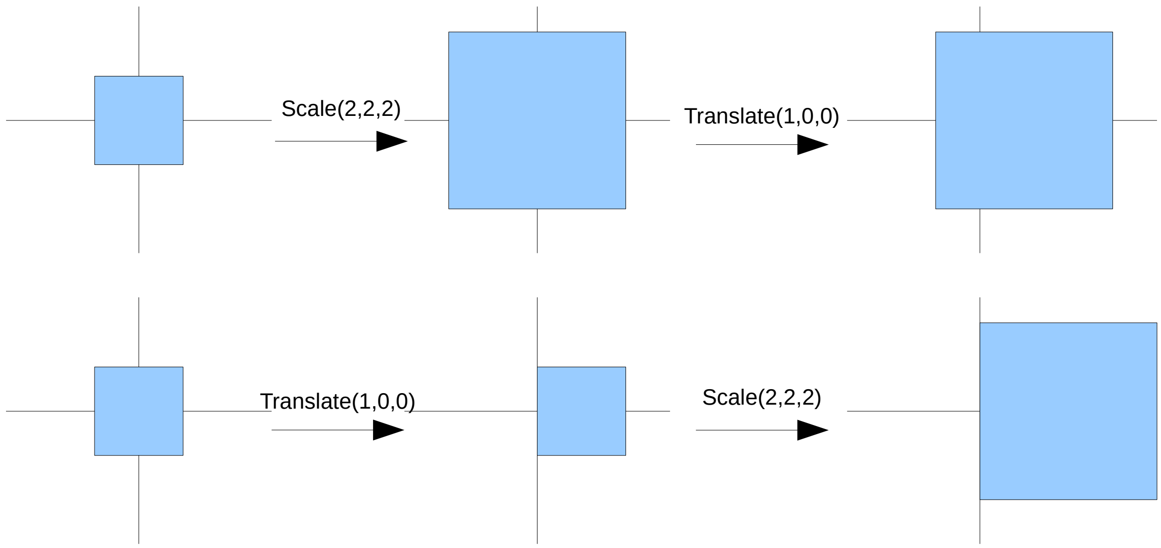


In matrix form, these two series of transformations are:

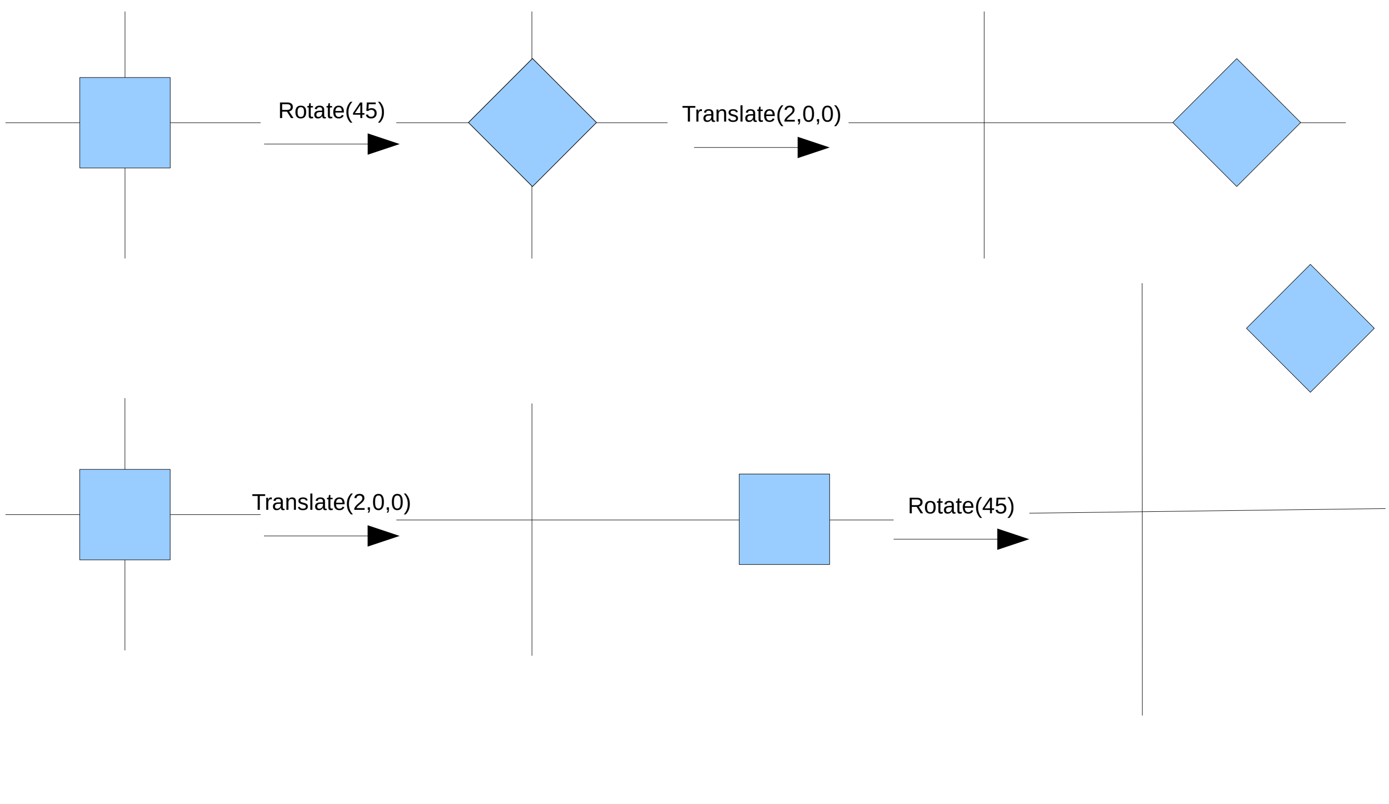
$$\begin{bmatrix} \cos 45 & -\sin 45 & 0 & 0 \\ \sin 45 & \cos 45 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 2 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 2\cos 45 & -\sin 45 & 0 & 0 \\ 2\sin 45 & \cos 45 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$\begin{bmatrix} 2 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos 45 & -\sin 45 & 0 & 0 \\ \sin 45 & \cos 45 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 2\cos 45 & -2\sin 45 & 0 & 0 \\ \sin 45 & \cos 45 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Scale vs Translate



Translate vs Rotate



Points, transformations, and homogeneous coordinates.

Outside of computer graphics, points and vectors are usually written as (x,y,z).
In computer graphics they are more often written as column vectors and usually with a fourth coordinate of 1 (or sometimes 0):

usually: $\begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$ and occasionally: $\begin{bmatrix} x \\ y \\ z \\ 0 \end{bmatrix}$ Generally: $\begin{bmatrix} x \\ y \\ z \\ w \end{bmatrix}$

This extra 4^{th} coordinate is called the homogeneous coordinate, and plays a very specific roll. (more on that later.) A point with a homogeneous coordinate is called a homogeneous point.

From linear algebra, matrices and matrix multiplication can be used as **transformations** of points

$$\begin{bmatrix} x' \\ y' \\ z' \\ w' \end{bmatrix} = \begin{bmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ a_{21} & a_{22} & a_{23} & a_{24} \\ a_{31} & a_{32} & a_{33} & a_{34} \\ a_{41} & a_{42} & a_{43} & a_{44} \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ w \end{bmatrix} \quad \text{where} \quad \begin{bmatrix} x' \\ y' \\ z' \\ w' \end{bmatrix} = \begin{bmatrix} a_{11}\,x + a_{12}\,y + a_{13}\,z + a_{14}\,w \\ a_{21}\,x + a_{22}\,y + a_{23}\,z + a_{24}\,w \\ a_{31}\,x + a_{32}\,y + a_{33}\,z + a_{34}\,w \\ a_{41}\,x + a_{42}\,y + a_{43}\,z + a_{44}\,w \end{bmatrix}$$

Homogeneous coordinates

Use 4D points (x,y,z,w) to represent 3D points like this

$(x, y, z, w) \rightarrow (x/w, y/w, z/w) \qquad \text{if } w \neq 0$

and so

$(x, y, z, 1) \rightarrow (x, y, z)$

If $w=0$, consider this sequence:

$$\begin{array}{ccc} (x,\ y,\ z,\ 1/10) & \rightarrow & (10\,x,\ 10\,y,\ 10\,z) \\ (x,\ y,\ z,\ 1/100) & \rightarrow & (100\,x,\ 100\,y,\ 100\,z) \\ (x,\ y,\ z,\ 1/1000) & \rightarrow & (1000\,x,\ 1000\,y,\ 1000\,z) \\ \vdots & \rightarrow & \vdots \end{array}$$

We can interpret $w=0$ as:

points at infinity, or
vectors (directions)

We will rig w to contain a useful quantity for perspective

Two Interactive approaches

Object mode: Object sits on a turntable tilt and zoom controlled by mouse
Navigation mode: Game-like navigation controls through a world scene.

Object mode:

Object sits on a turntable, with rotate, tilt and zoom controlled by mouse

C : Object sits on turntable centered at point C.
α : angle of turntable spin
β : angle of turntable up/down tilt
γ : (optional) angle of turntable spindle (up) projection
d : distance of viewing

Transformations: $T(0,0,d)\cdot R_z(\gamma)\cdot R_x(\beta)\cdot R_z(\alpha)\cdot T(-C)$

Navigation mode:

World like scene with ground, viewed from an interactively controlled eye/camera
Direction of view is controlled by the mouse: Left-Right turns head, Up-Down tilts head
WSAD keys move forward, backward, left, right, respecting ground height.

α : angle of head spin
β : angle of head up/down tilt
EYE: 3D position of eye

Transformations: $R_x(\beta)\cdot R_z(\alpha)\cdot T(-EYE)$

Complex transformations example

The individual transformations above are useful for building more complex combination transformations.

This makes use of a property of the **associative** property of matrix multiplication:

$$(BA)P = B(AP)$$

The right hand side says

Transform P by A, and then that result by B.

The left hand side says

Transform P by one product matrix AB.

Example: Rotate around a point other than the origin

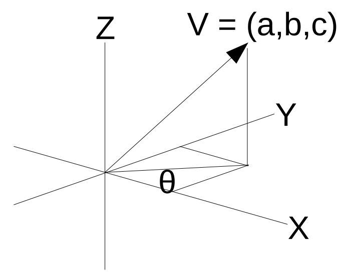
To rotate around a point which is not the origin:

1. Translate to the origin T(-x,-y)
2. Rotate R(45)
3. Translate back T(x,y)

For a total transformation of

T(x,y) R(45) T(-x,-y)

Example: Rotate around a vector other than an axis:



Let's build a rotation by α around a vector $V=(a,b,c)$.

Since we know only how to rotate around the axes:

1. Rotate V to the XZ plane ($R_Z(-\theta)$)
2. Rotate result to Z axis ($R_Y(\phi)$)
3. Do the desired rotate ($R_Z(\alpha)$)
4. Undo the 2nd rotate ($R_Y(-\phi)$)
5. Undo the 1st rotate ($R_Z(\theta)$)

Final transformation is product of 5 rotates. (Notice the order!)

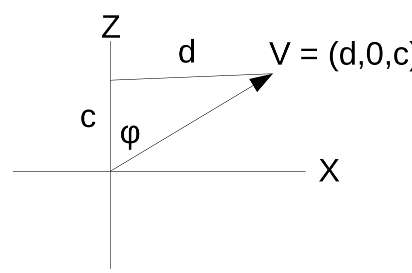
$$R_Z(\theta)\ R_Y(-\phi)\ R_Z(\alpha)\ R_Y(\phi)\ R_Z(-\theta)$$

What about θ and ϕ?

We don't need the angles,

we only need their sine and cosine.

A little trigonometry gets those values:



$$d=\sqrt{(a^2+b^2)}$$
$$l=\sqrt{(d^2+c^2)}=\sqrt{(a^2+b^2+c^2)}$$
$$\cos(\theta)=b/d$$
$$\sin(\theta)=a/d$$
$$\cos(\phi)=c/l$$
$$\sin(\phi)=d/l$$

Lookat transformaton

Specified by

E : Eye position

V : Direction of view

U : UP vector

Compute an orthonormal frame to produce the needed rotation:

Normalize V

Normalize U

Normalize $W = V \times U$ // If this normalization fails, set U=[0,1,0] and try again

$B = W \times V$ // Will be unit length as a property of the two cross products.

Step 1: translate to origin:

$T(-E)$

Step 2: rotate (expand this to a 4x4 matrix for actual use):

$$R = \begin{bmatrix} W_x & W_y & W_z \\ B_x & B_y & B_z \\ -V_x & -V_y & -V_z \end{bmatrix}$$ Note that: $$R\,W^T = [1,0,0]^T \quad R\,U^T = [0,1,0]^T \quad R\,V^T = [0,0,-1]^T$$

Why is that called a "rotate?

The Lookat transformation is $R * Translate(-E)$ which has the usual view transformation properties:

- The eye E is transformed to the origin
- The view direction V is transformed to the $-Z$ axis
- The up vector U projects into the screen Y axis.
- The product can be simplified to

$$R * Translate(-E) = \begin{bmatrix} W_x & W_y & W_z & -W \cdot E \\ B_x & B_y & B_z & -B \cdot E \\ -V_x & -V_y & -V_z & V \cdot E \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Orthonormal bases and rigid transformations:

Rows of R, taken as vectors are:

normal: $X \cdot X = Y \cdot Y = Z \cdot Z = 1$

mutually orthogonal: $X \cdot Y = Y \cdot Z = Z \cdot X = 0$

Such transformations are called *rigid* because:

canonical orthonormal vectors transform to orthonormal vectors

Has a matrix interpretation:

Like this: $R R^T = I$,

but: $R R^{-1} = I$,

so also: $R^T = R^{-1}$

This is a feature of all rotates:

$$R(\theta) = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix}$$

$$R(-\theta) = \left(R(\theta)\right)^{-1} = \left(R(\theta)\right)^T = \begin{bmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{bmatrix}$$

And of all products of rotates:

If A, B are rigid transformations, then AB is also:

$A^{-1} = A^T$, and $B^{-1} = B^T$

so

$(AB)^{-1} = B^{-1} A^{-1} = B^T A^T = (AB)^T$