COS 344: L6 Chapter 7: 3D Shapes and Transformations

Cobus Redelinghuys

University of Pretoria

07/03/2024

Introduction

- Today we will look at modeling and animations for 3D objects.
- After today's lecture, you should be able to start planning your model for Practical 3 and 4.
- Today's lecture contains a set of examples, which are posted on ClickUp.
- Not all of this week's content is in the textbook!

Section 7.1.5: Composition and Decomposition of Transformations

- Back in L5 we discussed how to rotate an object about itself when it is not at the origin.
  - What were the steps?

Section 7.1.5: Composition and Decomposition of Transformations

- Back in L5 we discussed how to rotate an object about itself when it is not at the origin.
  - What were the steps?
- Assume that we have the following matrices:
  - $\mathbf{T_1}$ which moves the object to the origin.
  - $\mathbf{R}$ which is the rotation matrix.
  - $\mathbf{T_2}$ which moves the object back to its original position.
- Assume the object we want to rotate has n vertices.

- Assume that each matrix is multiplied sequentially to each of the n vertices. How many multiplications do we perform?

- Assume that each matrix is multiplied sequentially to each of the n vertices. How many multiplications do we perform?
  - We multiply four matrices (assuming the point is a $D \times 1$ matrix) together n times.
  - Is there a way we can improve this?

- Assume that each matrix is multiplied sequentially to each of the n vertices. How many multiplications do we perform?
  - We multiply four matrices (assuming the point is a $D \times 1$ matrix) together n times.
  - Is there a way we can improve this?
- Note, the three transformation matrices $(\mathbf{T}_1, \mathbf{R}, \mathbf{T}_2)$ do not change from point to point.
- Why not calculate all the transformations once, and then apply to all of the n vertices?
  - This gives us how many calculations?

- Assume that each matrix is multiplied sequentially to each of the n vertices. How many multiplications do we perform?
  - We multiply four matrices (assuming the point is a $D \times 1$ matrix) together n times.
  - Is there a way we can improve this?
- Note, the three transformation matrices $(\mathbf{T}_1, \mathbf{R}, \mathbf{T}_2)$ do not change from point to point.
- Why not calculate all the transformations once, and then apply to all of the n vertices?
  - This gives us how many calculations?
    - Multiplying three matrices once.
    - Then multiplying two matrices n times.
- Example:
  - Consider that the object has a 100 vertices:
    - Inefficient: Multiply four matrices 100 times.
    - Efficient: Multiply three matrices once and two matrices 100 times.

- Using the efficient method, we obtain:

$$\mathbf{M} = \mathbf{T}_2\mathbf{R}\mathbf{T}_1$$

- Is the order of matrix multiplication important?

- Using the efficient method, we obtain:

$$\mathbf{M} = \mathbf{T}_2\mathbf{R}\mathbf{T}_1$$

- Is the order of matrix multiplication important?
  - Yes, you multiply the matrices in the order of transformations applied, starting from the right.
  - Firstly $\mathbf{T}_1$ is applied, then $\mathbf{R}$, and lastly $\mathbf{T}_2$
- Note: $\mathbf{T}_2\mathbf{R}\mathbf{T}_1 \neq \mathbf{T}_1\mathbf{T}_2\mathbf{R}$ for arbitrary matrices.
- Section 7.1.6 is skipped.

Matrices: Scale

- The scale matrix is expanded from 2D to 3D by adding a third dimension and an extra parameter.

$$scale(s_x, s_y) \rightarrow scale(s_x, s_y, s_z)$$

which implies:

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

- In 2D, the rotation was around the z-axis.
- In 3D, we can rotate around three distinct axes.
- Z-axis:

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

- X-axis:

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

- Y-axis:

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

Matrices: Shear

- In 3D, you can shear along the coordinate axes, just as with 2D.
- Z-axis:

$$shear_z(d_x, d_y) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ d_x & d_y & 1 \end{bmatrix}$$

- X-axis:

$$shear_x(d_y, d_z) = \begin{bmatrix} 1 & d_y & d_z \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

- Y-axis:

$$shear_y(d_x, d_z) = \begin{bmatrix} 1 & 0 & 0 \\ d_x & 1 & d_z \\ 0 & 0 & 1 \end{bmatrix}$$

- The normals of surfaces do not transform correctly when the standard transformation matrix $\mathbf{M}$ is applied.
- An alternative representation is required to transform the normal, such that it stays orthogonal to the surface.
- The textbook covers the derivation in Section 7.2.2.
- Use the formula:

$$\mathbf{n}_N = (\mathbf{M}^{-1})^T \mathbf{n}$$

  where:
  - $\mathbf{n}_N$ is the new normal.
  - $\mathbf{M}$ is the transformation matrix.
  - $\mathbf{n}$ is the old normal.
- As the length of the new normal can change, just remember to normalize the new normal for lighting/shading calculations.

Section 7.3: Translation and Affine Transformations

- Just as with 2D translations, we can create homogeneous coordinates for 3D translations.

$$\mathbf{T}(\mathbf{d}) = \begin{bmatrix} 1 & 0 & 0 & d_x \\ 0 & 1 & 0 & d_y \\ 0 & 0 & 1 & d_z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Section 7.4: Inverses of Transformation matrices

- Often times, it is needed to undo transformations.
- For example, $\mathbf{T}_2$, from our rotation example.
- Two methods exist:
  - **Option 1:** Taking the inverse of the transformation matrix $\mathbf{M}$.

$$\mathbf{M}\mathbf{M}^{-1} = \mathbf{I}$$

  - Thus, if we have a vertex $\mathbf{v}$ and we transform and undo the transformation, we obtain:

$$\mathbf{v}' = \mathbf{M}^{-1}\mathbf{M}\mathbf{v}$$

$$\mathbf{v}' = \mathbf{I}\mathbf{v}$$

$$\mathbf{v}' = \mathbf{v}$$

Section 7.4: Inverses of Transformation matrices

- Often times, it is needed to undo transformations.
- For example, $\mathbf{T}_2$, from our rotation example.
- Two methods exist:
    - **Option 2:** Creating a transformation matrix that will undo the operation.
    - Example:
        - If $\mathbf{T}(\mathbf{d})$ was applied then to undo it we can apply $\mathbf{T}(-\mathbf{d})$
        - If $\mathbf{R}_x(45^\circ)$ was applied then to undo it we can apply $\mathbf{R}_x(-45^\circ)$

- Consider we have a sphere centred at $\begin{bmatrix} 1 \\ 0.5 \\ -0.25 \end{bmatrix}$. We want to rotate this sphere by 25° around its x-axis.
- We construct the transformation matrix as follows:
  - First we need to translate the sphere such that the centre of the sphere is at the origin:

$$\mathbf{T}\left(-\begin{bmatrix} 1 \\ 0.5 \\ -0.25 \end{bmatrix}\right) = \begin{bmatrix} 1 & 0 & 0 & -1 \\ 0 & 1 & 0 & -0.5 \\ 0 & 0 & 1 & 0.25 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

  - Next the rotation matrix:

$$\mathbf{R}_x(25^\circ) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos(25^\circ) & -\sin(25^\circ) & 0 \\ 0 & \sin(25^\circ) & \cos(25^\circ) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

- Lastly, we need the matrix that will move the sphere back to the origin.

- Option A:

$$\mathbf{T}\left(-\begin{bmatrix}1\\0.5\\-0.25\end{bmatrix}\right)^{-1} = \begin{bmatrix}1 & 0 & 0 & 1\\0 & 1 & 0 & 0.5\\0 & 0 & 1 & -0.25\\0 & 0 & 0 & 1\end{bmatrix}$$

- Option B:

$$\mathbf{T}\left(\begin{bmatrix}1\\0.5\\-0.25\end{bmatrix}\right)^{-1} = \begin{bmatrix}1 & 0 & 0 & 1\\0 & 1 & 0 & 0.5\\0 & 0 & 1 & -0.25\\0 & 0 & 0 & 1\end{bmatrix}$$

- Pulling it all together:

$$\mathbf{M} = \mathbf{T}\left(\begin{bmatrix} 1 \\ 0.5 \\ -0.25 \end{bmatrix}\right)^{-1} \times \mathbf{R}_x(25^\circ) \times \mathbf{T}\left(-\begin{bmatrix} 1 \\ 0.5 \\ -0.25 \end{bmatrix}\right)$$

$$\mathbf{M} = \begin{bmatrix} 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0.5 \\ 0 & 0 & 1 & -0.25 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos(25^\circ) & -\sin(25^\circ) & 0 \\ 0 & \sin(25^\circ) & \cos(25^\circ) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & -1 \\ 0 & 1 & 0 & -0.5 \\ 0 & 0 & 1 & 0.25 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

- Then multiply **M** with every vertex of the sphere.

- This part is not explicitly clear in the textbook.

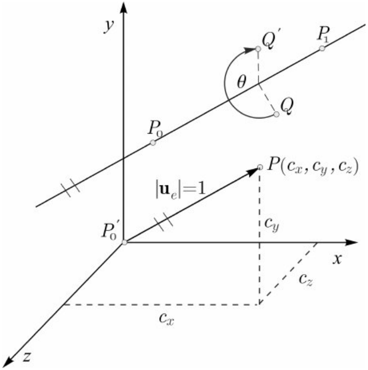


Figure: Rotation about an arbitrary axis

- We require:
  - The center of the object, $\mathbf{P_0}$.
  - The vector along which to rotate, $\mathbf{Q}$.
  - The angle of rotation, θ.
- $\mathbf{Q}$ can be obtained by:

$$\mathbf{Q} = \mathbf{P_1} - \mathbf{P_0}$$

.

- Always choose $\mathbf{Q}$ such that θ is positive.
- We need to normalize $\mathbf{Q}$:

$$\mathbf{u} = \frac{\mathbf{Q}}{|\mathbf{Q}|} = \begin{bmatrix} \alpha_x \\ \alpha_y \\ \alpha_z \end{bmatrix}$$

- We need two translation matrices:
  - $\mathbf{T}(-\mathbf{P_0})$
  - $\mathbf{T}(\mathbf{P_0})$
  - What is the purpose of each?
- Strategy:
  - Perform two rotations to align **u** with the z-axis.
  - Rotate by θ.
  - Undo the alignment rotations.
- Final result:

$$\mathbf{R} = \mathbf{R}_x(-\theta_x)\mathbf{R}_y(-\theta_y)\mathbf{R}_z(\theta)\mathbf{R}_y(\theta_y)\mathbf{R}_x(\theta_x)$$

- All we need now is θ_x and θ_y.

- Since **u** is a unit-length vector, we can exploit the following:

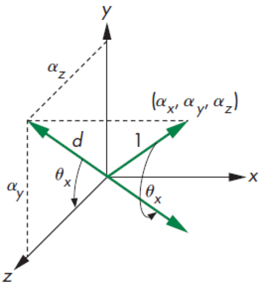
$$\alpha_x^2 + \alpha_y^2 + \alpha_z^2 = 1$$

- Now draw **u** and draw perpendicular lines from the point $(\alpha_x \alpha_y, \alpha_z)$ to each axis.
- In Figure 16 (Slide 16), these are represented by c_x, c_y and c_z.
- The directional angles between **u** and each axes is expressed as: ϕ_x, ϕ_y and ϕ_z.
- The directional cosines are:
  - $cos(\phi_x) = \alpha_x$
  - $cos(\phi_y) = \alpha_y$
  - $cos(\phi_z) = \alpha_z$

- As $cos(\phi_x)^2 + cos(\phi_y)^2 + cos(\phi_z)^2 = 1$, we can calculate θ_x and θ_y using the line segment.
- First, we need to rotate line segment into plane y=0.
  - Before rotation, if the line segment is projected onto the $x = 0$ plane, the line segment has a length of d.
  - We can calculate d as follows:

$$d = \sqrt{\alpha_x^2 + \alpha_z^2}$$

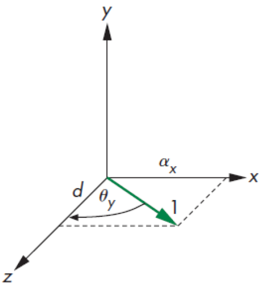
  - As such, we don't need to calculate θ_x or θ_y, we can rather use simple trigonometry.



This yields the following matrix:

$$R_x(\theta_x) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \frac{\alpha_z}{d} & \frac{-\alpha_y}{d} & 0 \\ 0 & \frac{\alpha_y}{d} & \frac{-\alpha_z}{d} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Rotate the line into the plane x=0.



This yields the following matrix:

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

This produces the final complete matrix concatenation as:

$$\mathbf{R} = \mathbf{T}(\mathbf{P}_0)\mathbf{R}_x(-\theta_x)\mathbf{R}_y(-\theta_y)\mathbf{R}_z(\theta)\mathbf{R}_y(\theta_y)\mathbf{R}_x(\theta_x)\mathbf{T}(-\mathbf{P}_0)$$

3D shapes

- How can we draw 3D shapes?

3D shapes

- How can we draw 3D shapes?
  - Using a set of 2D shapes.
- How would we draw the following shapes:
  - Pyramid
  - Cube
  - Rectangle

Joke of the day - By ChatGPT

Why did the 3D object refuse to go dancing?

Joke of the day - By ChatGPT

Why did the 3D object refuse to go dancing?

Because it couldn't handle the spin moves without getting dizzy!