HW2 Objects and Camera

CS 148 Fall 2020-2021
Due Date: Monday, 28 September 2020 by 7 pm PT

Follow the instructions carefully. If you encounter any problems in the setup, please do not hesitate to reach out to CAs on Piazza or attend office hours on Nooks.

Be aware of the Checkpoints below. Make sure you complete each one since we will do the grading based on them.

I. Objects and Transforms

In Blender, we can translate, rotate, and scale objects. We can also parent one object to another to move them around together. The "object space" and "world space" corresponds to "local coordinates" and "global coordinates" in Blender.

First, add a cube to your scene. If you open Blender with the default scene, it should come with a cube already. You can use that cube for this part of the homework.

(Optional Reading) How to transform objects in Blender

You can use the tools in the Toolbar of the 3D viewport: Move, Rotate, Scale, and Transform Tool to transform the object (fig 1). When you select an object, the orange dot indicates the origin of the object.

The keyboard shortcuts are G (for grab), R (for rotate), and S (for scale). You can press X, Y, or Z afterward to lock the transform to that specific axis. For example, if you press G, then X, then move your mouse, the object will only translate along the X-axis. You can also type in numbers after pressing the keys to specify the exact values you want to move, rotate, or scale. For example, if you press G, Z, 2, the object will translate 2 units along the Z-axis.

Make sure your cursor is in the 3D viewport area when pressing the keys. Blender uses cursor location to determine which area you want to send your keystrokes to.

You will be able to view/change the local transform of the object in two places: 1) the sidebar of the 3D viewport by pressing N, and 2) Property Editor (bottom right) → Object Properties. Under Transform, you should see the Location, Rotation, and Scale of the active object (fig 2). You can change the values and see how the object changes. You can change the value by clicking on the value and dragging it left or right, or reset the value by hitting backspace. Remember these values specify the local transform, not global transform.
1. Order of Rotations

To explore the ideas we introduced in the lecture, we will first investigate why the order of rotation matters. You will calculate the rotation matrices and verify your result with one of the vertices on the cube.

The default cube in Blender has a size of 2 units, which means its vertices are at locations of positive and negative 1s. We are going to use the vertex \( v \) with location \( p = \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix} \) (colored in red in fig 1) to see how rotations are constructed. The world (global) axis is shown by the navigation gizmo: X-axis points to the bottom left, Y-axis points to the bottom right, and Z-axis points upward.

First, we rotate the cube along the global X-axis by \( +45^\circ \). Then rotate along the global Y-axis by \( +45^\circ \) (fig 2). Let \( p_{xy} \) denote the world location of vertex \( v \) after rotation. (Shortcut: R, X, 45, Enter. R, Y, 45, Enter)

Next, let us go back to the default cube, but this time, we rotate the cube along the global Y-axis first by \( +45^\circ \). Then rotate along the global X-axis by \( +45^\circ \) (fig 3). Let \( p_{yx} \) denote the world location of vertex \( v \) after rotation. (Shortcut: R, Y, 45, Enter. R, X, 45, Enter)

Write down the rotation matrices for each step, and then use them to:

- **Checkpoint 1:** Write down \( p_{xy} \). (be prepared to show your work during grading)
- **Checkpoint 2:** Write down \( p_{yx} \). (be prepared to show your work during grading)

You are allowed to use calculator/write your own code to calculate the result.

**Hint**

The final location of vertex \( p \) is calculated by multiplying the rotation matrix for each step (be aware of the order!) to the left of \( p = \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix} \).

**Hint 2**

The rotation matrices for rotating along X-axis and Y-axis for \( 45^\circ \) are:

\[
R_x = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos \frac{\pi}{4} & -\sin \frac{\pi}{4} \\ 0 & \sin \frac{\pi}{4} & \cos \frac{\pi}{4} \end{bmatrix}, \quad R_y = \begin{bmatrix} \cos \frac{\pi}{4} & 0 & \sin \frac{\pi}{4} \\ 0 & 1 & 0 \\ -\sin \frac{\pi}{4} & 0 & \cos \frac{\pi}{4} \end{bmatrix}
\]
You may notice that you can choose various rotation modes for your object rotation. When you first rotate along Y-axis then X-axis, the rotation shows up as \((54.7, 30, 35.3)\) in XYZ Euler mode instead of \((45, 45, 0)\). If you change the rotation mode to YXZ Euler, the numbers will become \((45, 45, 0)\). This is due to the different order of rotations when defining Euler angles. For more information, see rotation mode in Blender documentation.

2. Parenting Objects

In lecture, we talked about hierarchical transforms, which can be done in Blender through parenting. Here we will explore how the location of child objects change with their parent.

First, set up the scene. Reset the rotation of the cube (put zeros into the Cube’s Rotation, or use Alt/Option R).

Set its location to \(t_{\text{cube}} = \begin{pmatrix} 2 \\ -1 \\ 1 \end{pmatrix}\). Add a plane (menu bar Add→Mesh→Plane), and in the pop-up panel in the bottom left, change its size to 8 (fig 1). The scene should look like fig 2, with the cube sitting on top of the plane.

Next, parent the cube to the plane. Select the cube, then hold shift and select the plane. Both objects should be in an orange outline, and the plane should have a lighter orange.

To learn more about multiple selections and active objects, see Blender Documentation.

Then go to the menu bar on top and select Object→Parent→Object (or leave your mouse inside the 3D viewport, and use the shortcut Ctrl/Cmd P) (fig 3). A dotted line should appear between the origin of the cube and the origin of the plane (fig 4), indicating an existing relationship between these two objects. You can verify this relationship by transforming the plane; the cube should follow the exact same transformation and stay on the plane.
After parenting, the translation of the cube relative to the plane is \( t_{\text{cube}}^{\text{local}} = \begin{pmatrix} 2 \\ 0 \\ -1 \end{pmatrix} \).

If we set the location of the plane to \( t_{\text{plane}}^{\text{world}} = \begin{pmatrix} 1 \\ 1 \\ 0 \end{pmatrix} \) what would the global location of the cube \( t_{\text{cube}}^{\text{world}} \) be?

If we then rotate the plane along its local \( X \)-axis by \( +45^\circ \), what would the global location of the cube \( t_{\text{cube}}^{\text{world}} \) be?

**Checkpoint 3:** Write down \( t_{\text{cube}}^{\text{world}} \) . (be prepared to show your work during grading)

**Checkpoint 4:** Write down \( t_{\text{cube}}^{\text{world}} \) . (be prepared to show your work during grading)

\[ \begin{bmatrix} 1 & 0 & 0 & 1 \\ 0 & \sqrt{2}/2 & -\sqrt{2}/2 & 1 \\ 0 & \sqrt{2}/2 & \sqrt{2}/2 & 2 \\ 0 & 0 & 0 & 1 \end{bmatrix} \]

and the homogenous coordinate for the location of the cube is:

\[ \begin{pmatrix} 2 \\ -1 \\ 1 \\ 1 \end{pmatrix} \]

**More Hint**

You can unparent the child object to check your answer. (Go to menu Object→Parent or shortcut Alt P, then select Clear and Keep Transformation)

II. Camera

Cameras are crucial to the virtual world. When you render, everything in the scene will be projected on to the camera film (screen space) to create your final image. In this section, we are going to play with the perspective camera and understand the effect of focal length. This is exactly how dolly zoom (or vertigo effect), a famous camera effect, works.

First, we are going to set up the scene. Open a new Blender file, delete the cube. Add a monkey (fig 1). Set the camera location to \((0, -4, 0)\) and rotation to \((90, 0, 0)\).

The scene should look like fig 2. Press the camera icon to go to the camera view.

Select the camera, and change its focal length to 25mm (fig 3).
Set the render engine to Workbench, and render the image.
Change the camera location to \((0, -8, 0)\) and focal length to 50 mm. Render the image.
Change the camera location to \((0, -16, 0)\) and focal length to 100 mm. Render the image.
Here we change the camera location along with focal length to keep the subject the same size.

- **Checkpoint 5**: Save the rendered images under these three camera settings.
- **Checkpoint 6**: Compare the three images in Checkpoint 5. Discuss the effect of changing the focal length.
  (No write-up needed, just get ready to answer this question during the grading session)

**Common focal lengths for real-world cameras (prime lens):** 24mm, 35mm, 50mm, 85mm, 105mm.

### III. Flat Shading

By default, the Workbench engine uses studio lighting. This assumes a pre-existing lighting environment for the scene, which enables us to better see the shape of the monkey by shading each face with its own color. We will discuss shading and lighting in future lectures. For now, we can change the settings to render the scene without any lighting and see what the render looks like.

Open a new Blender file, add at least two objects in the scene with different colors. (Detailed instructions in HW1/I/3). Transform them so they do not overlap while still being visible through the camera.

Change the lighting of Workbench to flat lighting (fig 1). Render the image.

- **Checkpoint 7**: Save the image with flat lighting.

### IV. Grading (5 pts total)

This assignment will be graded on the following requirements

- **Complete all the checkpoints (4 pts)**
  1. (0.5 pt) Write down \(p_{xy}\).
  2. (0.5 pt) Write down \(p_{yz}\).
  3. (0.5 pt) Write down \(t\)\(_{world}\).
  4. (0.5 pt) Write down \(t\)\(_{cube}\).
  5. (0.5 pt) Save the rendered images under these three camera settings.
6. (1 pt) Compare the three images in Checkpoint 5. Discuss the effect of changing the focal length.
7. (0.5 pt) Save the image with flat lighting.

Quiz Question (1 pt)
The TAs will choose one of the following questions during the grading session. Please be prepared to give a one-minute answer with your partner.

1. What is an advantage of using ray tracing instead of scanline rendering?
2. How is the viewing frustum used to keep objects in the scene in relative perspective for the viewer -- i.e. how does perspective projection work?
3. How does ray tracing work and how does it utilize triangles for lighting?
4. Why do we use triangles rather than other polygons in rendering? give at least 2 advantages.
5. When does aliasing occur? How do you fix this?
6. What is the primary role of the pupil and why is it important?
7. What 3D transformation can not be represented by a 3×3 matrix? How do we incorporate that into matrix representation?
8. Can we use the screen space barycentric coordinates to linearly interpolate the “z value” at a pixel? Why? (No mathematical details, be prepared to explain the high-level idea)