

1. Overview

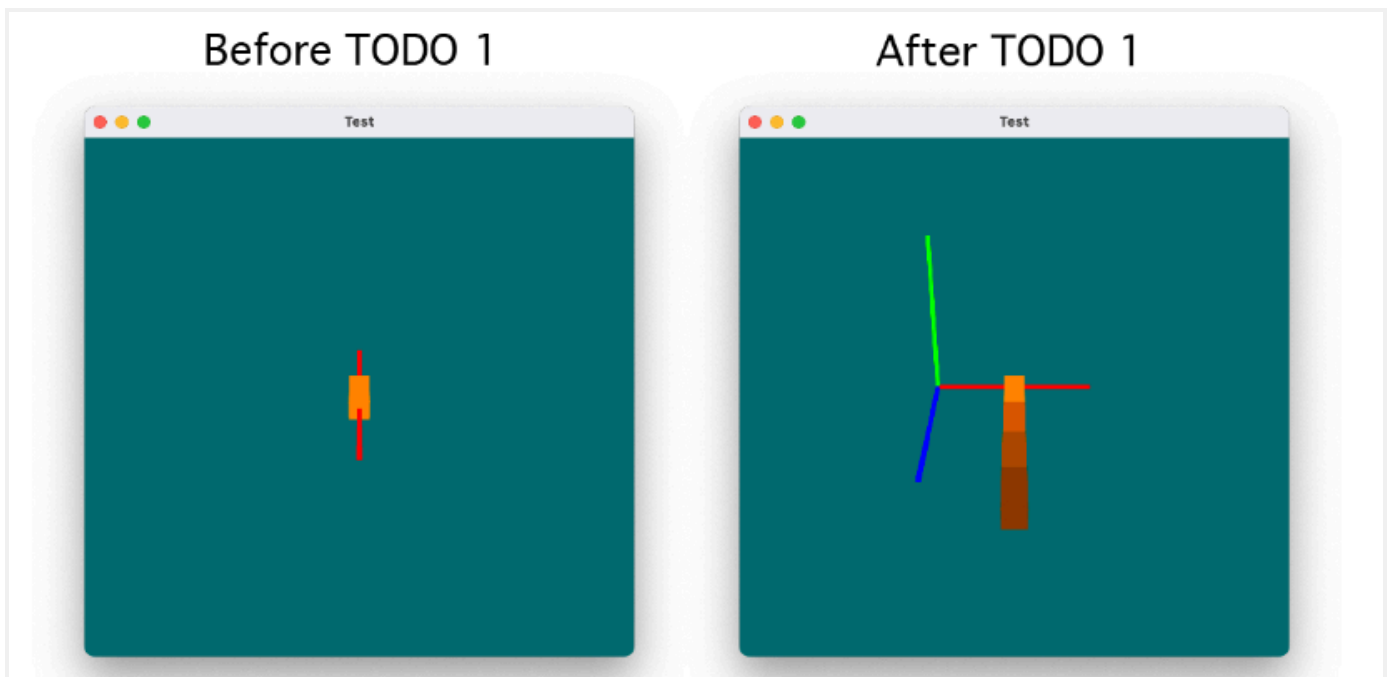
This programming assignment is meant to acquaint you with 3D modeling and model transformation. You will implement a creature model and make it move by performing transformations on its joints. You will also need to submit a simple reference image or concept drawing and a screenshot of your model with your code.

Basic Requirements

Before coding, please have a look at the Component class defined in "Component.py". We recommend calling functions of the Component class in your keyboard event functions to make the joints/body of your creature move. After reading the code in the "Component.py" file, you should start implementing features in the following blocks/files in the skeleton code:

1. TODO 1 in "Component.py": complete the Component.update() function that sets the transformation matrix of each component. Most of this function has already been written: you only need to provide one line of code that shows the correct multiplication order of the matrices that have already been created for you. Specifically, you need to account for translations, rotations, and scaling. You can multiply numpy matrices together using the @ operator (e.g. `self.transformationMat = C @ B @ A`).

Here is what the program will look like once the correct transformations are applied:



2. TODO 2 in "ModelLinkage.py": implement your own creature class by following the form of the provided ModelLinkage class. You will need to make use of the geometric primitives defined in Shapes.py. Combine these shapes in a hierarchical fashion to construct a sufficiently complex 3D model of a creature (insect, spider, scorpion, etc.) Please see "[Model Requirements](#)" for additional details on what is required of your model. You will need to figure out how to store components in a hierarchical structure and set them in proper position by chaining transformations. It is helpful to modularize your design in one class — you may wish to reuse your model in future programming assignments.
3. TODO 3 in "Sketch.py": After building your model, you need to create an instance of it in Sketch.py. See the InitGL function.
4. TODO 4 in "Sketch.py" or "ModelLinkage.py": Set up the joint behaviors of your creature. Please refer to the Component class to see what functions are already available to you. Limit the angles at each joint within

- reasonable ranges so that the creature's legs, neck, head and any other parts don't intersect or bend in unnatural ways. **The orientation of joint rotations for the left and right limbs must mirror each other.**
5. TODO 5 in "Sketch.py": Set up the keyboard events that make your creature act in different ways (change poses, rotate limbs, etc.). **Add a multi-select feature to the interface** that allows you to control several joints at the same time. You will need to set up at least 5 different poses for your creature as test cases. Please check your keyboard events and test cases to see if they work as expected and satisfy our requirements. See "[Multi-Select Requirements](#)" below for additional details.
 6. TODO 6 in "Sketch.py" (**CS680 required, CS480 Extra Credit**): Implement an eye that always looks at the position of the mouse. An eye should consist of a movable pupil and still sclera. The eye rotation only needs to work correctly when the creature is looking toward the viewer — you do not need to account for other camera orientations.

Extra Credit

7. Use a **single quaternion transformation** rather than a series of Euler rotations to implement the eye tracking feature for additional credit.

Multi-Select Requirements

Your multi-select interface should behave as follows:

- Any subset of your model's Components can be selected. When inducing a rotational change via the up/down arrow keys or scroll wheel/trackpad, the rotation will be applied to all Components that are currently selected.
- Any individual Component can be added to the current selection or removed from the current selection without adding/removing other Components.
- Selected Components are indicated **visually** to the user by setting their color to that of the current rotational axis. This matches the behavior of the single-select interface that is provided to you.
- Once deselected, Components regain their original color.

The rest of the implementation is left up to you! One straightforward approach is to assign each Component of your model to a key (you can use 1 through 0 and/or A through L for this purpose, since they don't conflict with any other program controls). When a Component's assigned key is pressed, that Component is added/removed from the current selection.

Model Requirements

In addition to roughly following the reference image/drawing you will provide, your model should be suitably complex:

- at least 6 limbs required (tails and wings count)
- at least one limb must have at least 3 joints
- at least two colors should be used
- at least two shape types must be used
- a pair of opposing limbs must be present (to test mirrored motions)

Programming Style

8. For any modified or newly added source file, you should include a brief description of how this file was changed. Add this information to the file heading along with your name and ID. Your code should be readable with **sufficient comments**. You should use consistent variable naming and keep reasonable indentation. You may add helper functions to improve efficiency and readability.

README

With each assignment, please upload a README text or markdown file with the following information:

- your name
- any collaborators that you spoke to
- the class
- the assignment number
- a brief summary of the code and implementations you have written for the assignment

For the last point, you should outline the method you used to solve the problem, describe the variables and data structures used in the program, any error conditions that might be encountered in the solution, and how these error conditions are handled by the solution. This is a general description of your work, detailed comments are still required in the code.

If these details are described in the comments above the relevant functions or files, you may be brief here. You may call this file "README_Student" or something analogous to differentiate it from the README in the skeleton code.

Please include a mention of any resources that you consulted while completing your assignment.

2 Resources

2.1 Starter code

A Python Program skeleton, which includes basic classes, methods, and main pipeline, is provided for you. You are expected to complete the sketch program by completing/modifying Sketch.py. There are comments in the skeleton code that will help guide you in writing your subroutines. Some of them are noted as "TODO" or "BONUS" which suggests you should complete the corresponding block.

Download the source files: [PA2](#)

2.2 Environment setup

Installing the appropriate programming environment should be covered in a lab session. For more step-by-step instructions, please check the "[Environment Setup](#)".

2.3 User Interface

The user interface to the program is provided through mouse buttons and keyboard keys:


- **ENTER/RETURN**: Cycle through components
- **LEFT-ARROW, RIGHT-ARROW**: Iterate through different rotation axis for the currently selected component
- **UP-ARROW**, mouse **SCROLL-UP**: Increase selected component's rotation angle along the selected axis
- **DOWN-ARROW**, mouse **SCROLL-DOWN**: Decrease selected component's rotation angle along the selected axis
- mouse **Left-Drag**: change the viewing angle
- mouse **Right-Drag**, mouse **Middle-Drag**: move the camera up/down and left/right
- **r**: reset the viewing angle
- **R**: reset everything in the scene

After modifications, your interface may be different from the example program provided here.

Don't forget to add a multi-select feature so multiple joints can be changed simultaneously.

2.4 Demo

A video demo has been prepared to help you better understand your tasks and speed up your debugging process:

 PA2_Demo.mp4

3. Submission: Due Tuesday 10/8, 11:59 PM EST(midnight)

3.1 Files to Submit

Your program's source files are to be submitted electronically on **Gradescope**. Please wrap everything in your project folder (including asset files) to a zip file and submit it. Note that for this assignment, you will also need to submit a simple reference image or concept drawing of your model.

3.2 Demo

Part of your grade for this programming assignment will be based on your giving a short demo (5 minutes) during the CS480/680 scheduled labs following the assignment due date. You will be expected to talk about how your program works, and will be asked related comprehension questions.

3.3 Reference image/drawing and screenshot

Please submit a reference image or drawing of your design and a corresponding screenshot. The image/drawing can be relatively simple, and is simply to motivate some modeling with intent.

4. Grading

Students can receive at maximum 10 points of extra credit.

Task	480	680
3D Creature model constructed satisfies requirements for shapes, hierarchies, and color	25	25
Proper rotation at the joints	25	25
Limit rotation so that limbs do not bend in unnatural ways	10	10
Your predefined 5 creature poses work properly	25	25
The creature design is similar to your reference image/drawing	5	5
Eye movement	5 (extra)	15
Eye movement with quaternions	10 (extra)	10 (extra)
Programming style	10	10

5. Code Distribution Policy

You acknowledge this code is only for the course learning purpose. You should never distribute any part of this assignment, especially the completed version, to any publicly accessible website or open repository without our permission. Keep the code in your local computer or private repository is allowed. You should never share, sell, gift, or copy the code in any format with any other person without our permission.

Appendix: Default Rotational Behavior

In order to simplify the process of constructing your model, the rotational origin of each Shape has been offset by $-1/2 * \Delta z$, where Δz is the total length of the shape along its z-axis. In other words, the rotational origin lies along **the smallest local z-value** rather than being at the translational origin, or the object's true center. This allows Shapes to rotate "at the joint" when chained together, much like segments of a limb. **In general, this means that you should construct each component such that it is longest in its local z-direction: otherwise, rotations may not behave as expected.**

If you would like to disable this behavior for a particular object, rotating it about its translational origin instead, you can pass "limb = False" to the Shape constructor when creating the object. This is particularly useful for body parts that *do* rotate about their centers, such as ball joints and eyes.

Below are some illustrations to show how this behavior works:

