UTD CS 4361 Assignment 3: Transformation

Due on Oct 19, 2025

1 Introduction

In this Assignment you will utilize your knowledge of transformations to make things move. Here we will study how to build and animate with object hierarchies.

1.1 Getting the Code

Assignment code is released on the E-learning page. Please download it to your local machine, navigate to the folder where you intend to keep your assignment code, and run the following command from the terminal or command line:

```
unzip a3-release.zip
```

1.2 Template

- `A3.html` is the launcher of the assignment. Open it in your preferred browser to run the assignment, to get started. You do not need to modify this file.
- `A3.js` contains the JavaScript code used to set up the scene and the rendering environment. You will do most of your work here for this assignment.
- `glsl/` contains the vertex and fragment shaders for eye, sphere, and laser. You need to modify all of them.
- `js/` contains the required JavaScript libraries. You need to modify 'js/setup.js' to load the glTF dog model.
- `glb/` contains the geometric model which needs to be loaded in the scene.
- `images/` contains the texture images used.

1.3 Anti-AI/Plagiarism Measures

To prevent plagiarism, 10 students will be randomly selected. These students are required to present their final submitted code to TAs during office hours and point out the locations of core code for the corresponding problems.

2 Work to be done (100 points)

Here we assume you already have a working development environment which allows you to run your code from a local server. (if you do not, check out instructions from Assignment 1 for details). Once you have set up the environment, Study the template to get a sense of how it works.

2.1 Part 1: Required Features

a. **(15 points)** Load the glTF dog model.

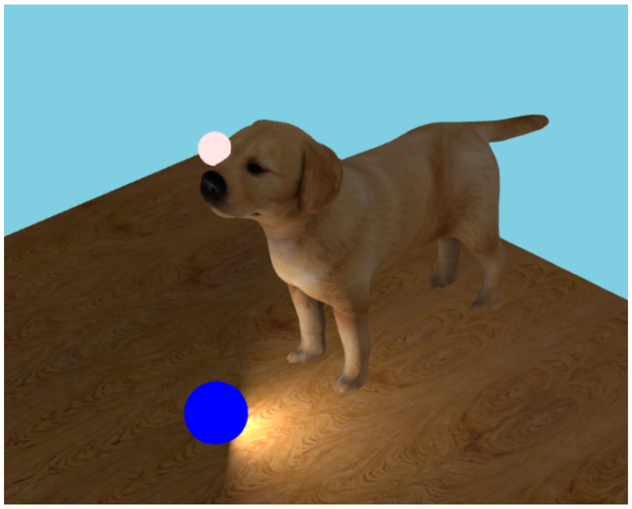

Figure 1: Question 1.a. Starter scene.

In this assignment, we'll shift from an OBJ model to a glTF model which is more powerful. A rigged glTF dog is provided in `glb/`. Your task is to load the dog into the scene as in Figure 1. Specifically, fill in function `loadAndPlaceGLB()` in `js/setup.js` and make necessary changes to `A3.js`. You can check THREE.js's website for glTF loader and function `loadAndPlaceOBJ()` for reference.

Resource: https://threejs.org/docs/#examples/en/loaders/GLTFLoader

b. **(20 points)** doges have Eyes. Your task here is to bless the dog with a pair of googly eyes. One eyeball model is already provided. Your job is to place two copies of this eyeball onto where the eyeballs should be on the dog using an appropriate `modelMatrix`. The result should look similar to Figure 2.

Hint 1: You can do this entire question in `A3.js`.
Hint 2: Don't worry if the eyes are initially pointing backwards into the dog's head; in the next question you will have the opportunity to orient them correctly.

c. **(20 points)** Staring at the sphere. What good are eyes if they don't `lookAt` anything? Direct the dog's gaze towards the nefarious floating sphere. Ensure that the eyes look at

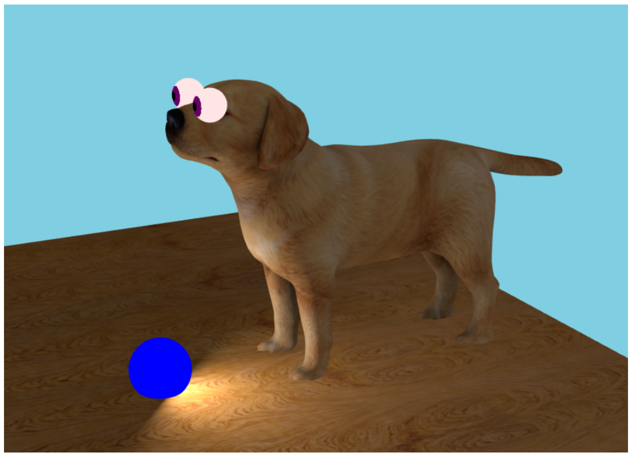

Figure 2: Question 1.b: Give the dog eyes.

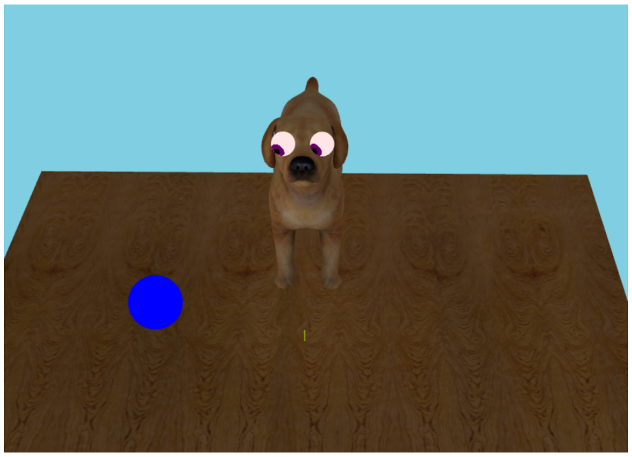

Figure 3: Question 1.c: Rotate the eyes to look at the sphere.

the sphere as it moves around the scene as shown in Figure 3. Also, notice that in Figure 3 each eye contains a pupil and an iris that help you with verifying your `lookAt` works. You can modify the provided eyes' fragment shader under `glsl/` to achieve this effect.

Hint 1: `THREE.Matrix4 and THREE.Object3D` have a method called `lookAt` that may be of use.

d. **(20 points)** Laser eyes! Arm the dog with eye lasers. Shoot the lasers from the dog's eyes at the sphere when it gets too close. A distance threshold (units in world space) named `LaserDistance` is provided in `A3.js`. To be more specific, create two lasers from the eyes to the sphere. You need to create the geometries and shaders by yourself. Figure 4 shows a possible result.

Hint 1: You may use any kind of material for the lasers (even a `ShaderMaterial`).
Hint 2: Note that the distance threshold between the sphere and the eye is given in the world frame, but the scale of the laser will likely be in a different frame.

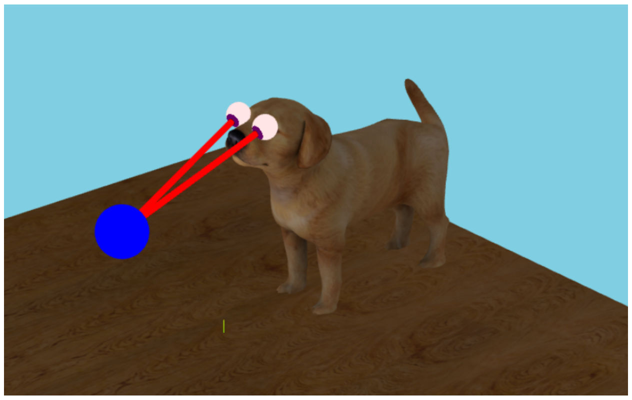

Figure 4: Question 1.d. Give the dog laser eyes!

e. **(25 points)** You're too close! Make the dog wave its tail, and wave more vigorously when the Orb is "too close" to it, as if it is scared. By "too close" we mean when the horizontal distance (measured on the x-z plane in world space) between the sphere and the dog's center is within a threshold distance `waveDistance` (units in world space). Also, when the distance is within the threshold, make the waving even more vigorous as the Orb gets closer. Also, when the sphere is close enough to the dog for the tail to start waving faster, make the dog look at the camera instead of the ball as if to let you know to stop bringing the ball that close!

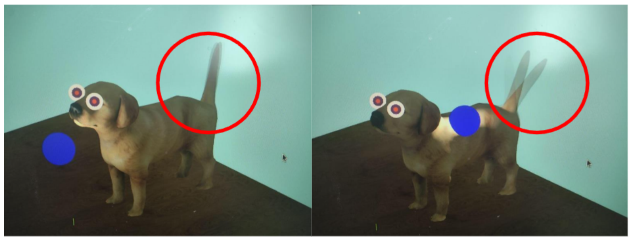

Figure 5: Question 1.e. Closer to the dog with more vigorous waving!

Hint 1: To animate the dog, you may want to access bones within the dog's skeletal structure. To do this, you can make changes at where you called `loadAndPlaceGLB()`.
Hint 2: Beware that Javascript callbacks are asynchronous, if you ever need to access a variable defined within a callback function from somewhere outside of the callback's scope.
Hint 3: You may use the `clock` variable for implementing the dog's periodic waving.

2.2 Part 2: Creative License (Optional)

You have many opportunities to unleash your creativity in computer graphics! In this **optional** section, and you are invited to extend the assignment in fun and creative ways. We'll highlight some of the best work in class. A number of exceptional contributions may be awarded bonus points. Here are a couple of ideas we can think of:

- Create more complex motions for the dog, e.g. making its head move around, while also making the eyes and the laser beams move along with the head.
- Make the dog dance!
- Implement an inverse kinematic solver for some of the dog.

3 Submission Instructions

3.1 Directory Structure

You must submit your final code and write a clear `README` file which includes your name, student number, and the core idea/changed code/explanation/screenshot of each question. The TAs strongly encourage you to read the template of answers in template.md and use a similar template to submit your answers.
Your README file can be in PDF, Word, or Markdown format. It must be placed in the root directory and will serve as the basis for grading your work. **Missing a README file will result in a 30-point deduction.**

3.2 Submission Methods

Please compress everything under the root directory of your assignment into `a3.zip` and submit it on Canvas. You can make multiple submissions, but we will grade only the last one.

4 Grading

4.1 Point Allocation

Each assignment has 100 points for Part 1. Part 2 is optional and you can get bonus points (0-10 points) at the instruction team's discretion. The max score for each assignment is 110 points. **Missing a README file will result in a 30-point deduction. Please make sure the README file includes the screenshot of each question.**
TAs will carefully review your README file and evaluate based on idea/code/explanation/screenshot. If the screenshots show correct rendering effects, full points will be graded.

4.2 Anti-AI/Plagiarism Measures

To prevent plagiarism/AI cheating, 10 students will be randomly selected. These students are required to present their final submitted code to TAs during office hours and point out the locations of core code for the corresponding problems. TAs will typically notify selected students via email with meeting links, offline addresses, and a schedule for students to book appointments. Each student will have 5 minutes to answer questions about their code.

4.3 Penalties

Aside from penalties from incorrect solution or plagiarism, we may apply the following penalties to each assignment:
Late penalty. A deduction of 10 points will be applied for each late day. Note that

1. We check the time of your last submission to determine if you are late or not;
2. We do not consider Part 1 and Part 2 submissions separately. Say if you submitted Part 1 on time but updated your submission for Part 2 one day after the deadline, that counts one late day.

AI/Plagiarism penalty. If a student cannot point out where their code is located for solving specific problems (e.g., which lines of GLSL code correspond to the answer for a particular problem) and cannot clearly articulate their approach to solving the problem, this will result in a complete deduction of points for the corresponding assignment.