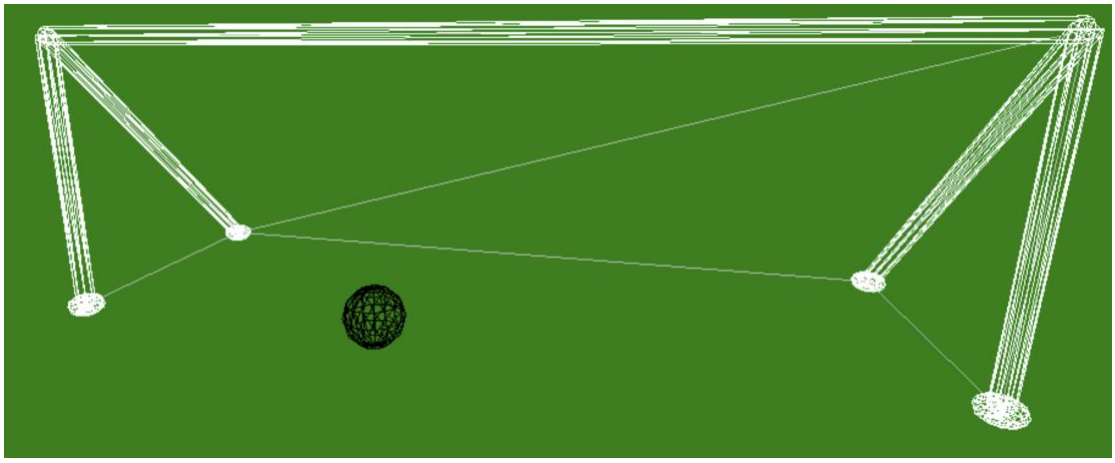
Reichman University: Computer Graphics Course, Spring ‘24
Instructor: Prof. Ariel Shamir
Submission due: 16th 2024 (23:59)

Exercise 5 – Modeling with WebGL



Overview

In this exercise, you will practice basic WebGL modeling, viewing, and projecting techniques. Your goal (😉) is to create an application allowing users to view a WebGL-rendered model. You must make a model of a football goal and a ball that moves near it.

The model you create here will also serve you for the next exercise.

Reichman University: Computer Graphics Course, Spring ‘24
Instructor: Prof. Ariel Shamir
Submission due: 16th 2024 (23:59)

Usage

- Pressing ‘w’ toggles wireframe view.
- Pressing ‘o’ toggles the Orbit Controls. (already implemented)
- Pressing ‘-’ / ‘+’ modifies the speed of the ball. (partially implemented)
- Pressing ‘1’ / ’2’ will toggle the ball's trajectory.
- Pressing ‘3’ will shrink the goal while preserving its proportions.

Modeling

There are multiple ways to model a football goal. Here, we require a minimum level of detail. You may, however, embellish the model with as many additional details as you see fit. Alternatively, you could build a 3D goal model that is at least as impressive as our model as long as it follows **the mode specifications**.

Building Blocks

The minimum set of primitives you need to use for modeling are:

- Cylinders for the two front goalposts, the two back supports, and the crossbar.
- Rings or toruses at the edges of the goalposts and back supports.
- Double-sided mesh planes for two side nets and the back net.
- A sphere for a ball.

Our model is built from these 3D primitives, along with the necessary affine transformations.

Structure

Your goal model must be modular and constructed as a group of primitives. Following is a list of the parts in our model. These are only suggestions, but you must use the building blocks mentioned above.

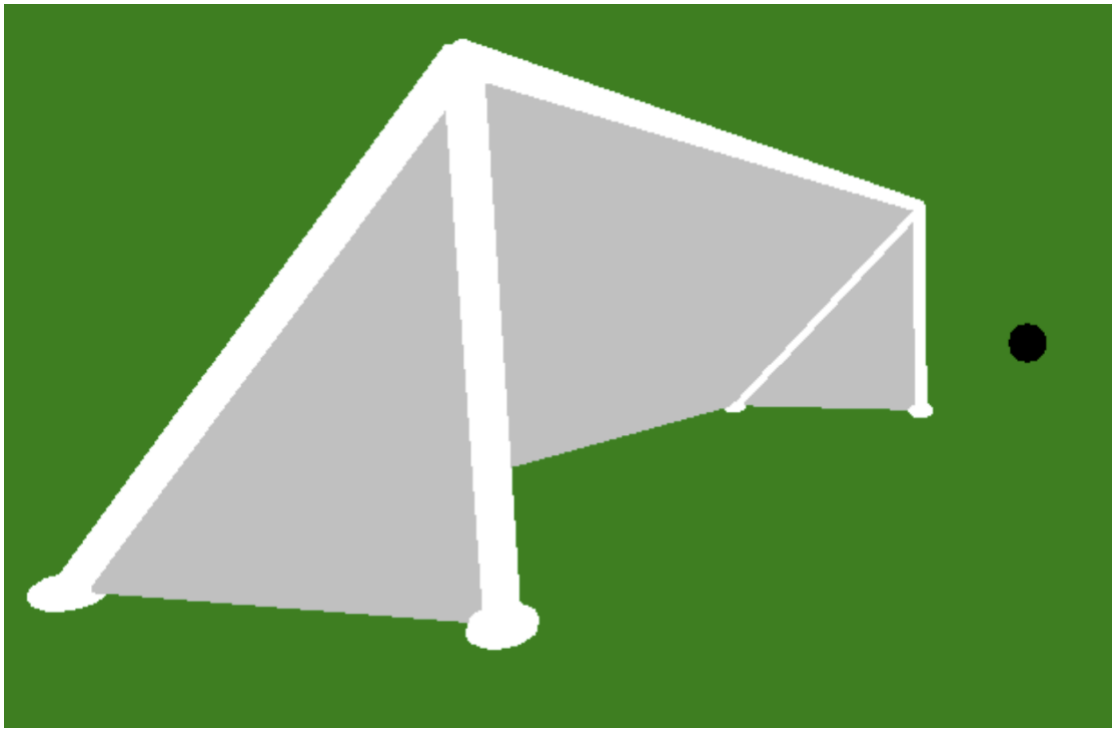
- Goal:
  - Skeleton: The main components of the goal. It must be white. Has these sub-components:
    - Crossbar: Long horizontal cylinder.
    - Goal posts: Two identical vertical cylinders next to each other. Each goalpost also has a torus or a ring at its edge.
    - Back supports: They are the same as goalposts, but notice that the cylinders are placed diagonally, yet the torus rings are perpendicular to the ground, just like the goalposts’ torus rings.
  - Nets: a rectangular in the back of the goal and two triangles at the sides of the goal. The nets must be light gray.
- Ball: A black sphere.

Reichman University: Computer Graphics Course, Spring '24
Instructor: Prof. Ariel Shamir
Submission due: 16th 2024 (23:59)

Model Specifications and Unit-measure

- **Ball:**
  - The ball must be a sphere object.
  - Ball to goal vertical scale ratio: 1:8 (eight balls on top of each other are of the same height as the goal)
  - Vertically, the goal, in its entirety (remember that it has a radius), must be somewhere between the top and bottom of the goal.
  - Depth-wise, the ball, in its entirety, must be far from the goal's entry by some distance of your choice.
- **Goal:**
  - **Skeleton:**
    - The width/height ratio of the net must be 3:1.
    - The angle between the goalposts and the back supports must be between 30° and 45°.
  - **Nets**:
    - The three nets must be wrapped between the skeleton parts.

If we do not define any limitation or specification, you can do so! We want you to have some creative freedom.



Reichman University: Computer Graphics Course, Spring '24
Instructor: Prof. Ariel Shamir
Submission due: 16th 2024 (23:59)

Positioning and rotating the objects

As we've learned in class and as you might also see online, there are many ways to manipulate a specific object pose. However, we require you to manipulate the pose only by applying matrices **to objects and multiplying matrices.**

This means you **cannot** use commands like *cube.position.x = 6,* etc.

However, you may use the already implemented basic Matrix4 commands, such as makeRotation and makeScale, and everything in the official documentation.

Viewing

The user should be able to rotate the model using the mouse (specifically, dragging it while clicking it). The viewing module involves projecting the positions of the mouse before and after the click on a sphere and then computing the rotation needed to transform between the two. This transformation is then performed on the model. This module is already implemented. You may read the implementation details here.

Surface Color

In this exercise, it is recommended that you use MeshPhongMaterial for your objects. In the next exercise, we will add light sources and define these materials, so it is important that you have the framework prepared for it.
Note: As you already know, working with Phong material requires a light source. If you wish to use the MeshPhongMaterial, you must add any kind of lighting (your choice). If you prefer to change it in HW6, you may use the **MeshBasicMaterial** for now.

Wireframe

As you saw in the recitation, we can change how the material works using the wireframe attribute set to **true** or **false**. You should implement a way to turn all the wireframe materials on and off simultaneously.
Hint - you can change any attribute during runtime using *material.attribute = value*.

Interactivity

You are required to 2 types of animations, each named after the keyboard key that toggles them on and off.
'1' - The ball will rotate in a single axis of your choice, and by that, it will enter the goal. Remember to consider the speedFactor variable.

'2' - The ball will enter the goal by doing a rotational movement in a **different** axis of your choice, not the same one from animation '1'. Remember to consider the speedFactor variable.

'Up arrow' - If animated, the speed of the ball increases (partially implemented).

'Down arrow' - If animated, the speed of the ball increases (partially implemented).

You are also required to implement this:

'3' - The goal, in its entirety, will shrink by 5% in each of its axes. Each click applies shrinking again (e.g., after two clicks, the goal's length in each dimension is 90.25% of its original length).

Note that animations 1 and 2 are not exclusive, meaning you should be able to toggle all of them at once (in that case, it's okay if the ball doesn't enter the goal).

Keyboard Trigger

To create keyboard triggers, we can use the **EventListener** implemented in JavaScript. You would want to use the "keydown" event type. You can see how to implement this logic in our framework for toggling the orbit controls. Note that the argument is a callback, so program accordingly!

Framework

You are given a framework that will set up a small server on your computer for you to work on your project. We made this so that you can work with multiple JavaScript files on the same webpage without invoking CORS errors.

To work with the server, you should download node.js (the LTS version is recommended). After installing, you need to set up your environment:

1. Use your terminal to navigate to the folder in which the environment is prepared.
2. Write *npm install* to install the server framework.
3. To run the server, write node index.js in your terminal. It will start hosting a server.
4. By default, your server will be at http://localhost:8000/

Improve the appearance or add animation to the Model (up to 8 points bonus)

In addition to the basic goal and ball models, you can design additional features (such as seats with fans, a trophy, or a flag of your favorite team) or anything else you see fit. There will be up to 8 points bonuses for creative additional features. To be eligible for the bonus points, you must add a README file describing what you did to your submission.

Reichman University: Computer Graphics Course, Spring ‘24
Instructor: Prof. Ariel Shamir
Submission due: 16th 2024 (23:59)

Grading

- Goal model - 50 points
    - Goal posts and crossbar - 15 points
    - Back supports - 20 points
    - Nets - 15 points
- Ball model - 5 points
- Wireframe mode - 10 points
- Animations - 20 points (2 x 10)
- Speed change - 10 points
- Goal size scale - 5 points
- Bonus - up to 8 points

Recommended Milestones

Design the 3D model in a bottom-up fashion. We suggest the following implementation milestones:

1. Start by adding only a single geometry and rendering it. Make sure that you understand how the orbit controls work. Note that we already created a box for you—you should remove it and replace it with a different geometry.
2. Implement the toggle for the wireframe, as it will help you with your designing and might be more complex to add later on when there are many materials.
3. Complete the goal’s skeleton:
    a. Start with the goalposts and crossbar.
    b. Continue with the diagonal back supports.
    c. Add the toruses\rings at the ends of the goalposts, back supports, and the crossbar.
4. Add the goal nets.
5. Add the ball and place it correctly beside the goal.
6. Start working on animations ‘1’ and ‘2’ together, as they are very similar. Make sure the speed is affected by the up and down arrows.
7. Add the goal scaling action ‘3’.

Tips

- Pay attention to the order in which you present the polygon's vertices.
- You can use the wireframe mode (by pressing “W”) to see how the provided jar file models different parts.

Reichman University: Computer Graphics Course, Spring ‘24
Instructor: Prof. Ariel Shamir
Submission due: 16th 2024 (23:59)

Submission

- Submission is in pairs
  - The zip file should include only a single script - hw5.js
  - A short README document if you add more elements to your model or add new animations explaining what you’ve implemented.
- The zip file should be submitted to Moodle.
  - Name it:
  - <Ex##> <FirstName1> <FamilyName1> <ID1> <FirstName2> <FamilyName2> <ID2>
- Before submission, be sure to check for updates on Moodle
- **There will be a point reduction of up to 10 points for submitting incorrectly.**