Stevens Institute of Technology
CS 537 – Interactive Computer Graphics
Dr. Chloe LeGendre

Assignment 2

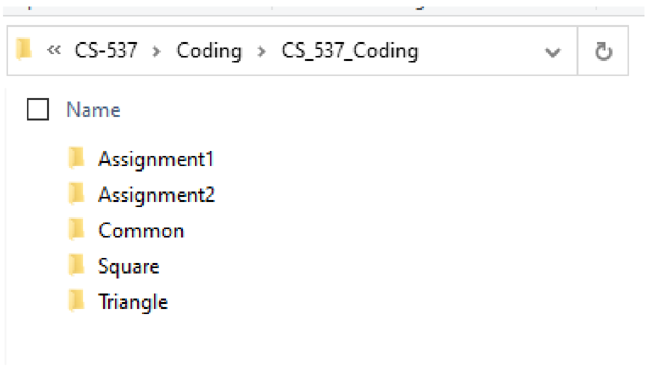
Date posted to Canvas: Feb 28, 2021
Due date: March 14, 2021; 11:59 PM

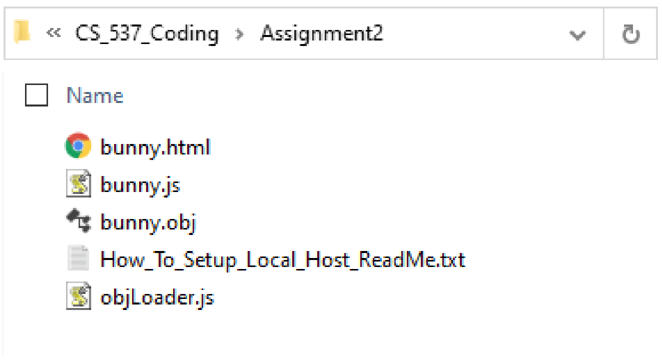
Overview

In this assignment, you'll be asked to render a more complex geometric shape to the screen, and you'll be asked to write shaders that will adjust the position, orientation, and scale of the model with respect to the camera. You'll also be working with user input and interaction, and several buttons and key presses will trigger different events to be displayed. This will also be the first time that we discuss how to load assets from outside the html and JavaScript files that will be needed for rendering, so we'll walk through how to set this up. The content of this assignment will be covered predominantly in the readings and notes of Week 04 and Week 05. As such, it will be due at the end of Week 06. **This is an individual assignment and you should not work in a group or share code with your peers.** Discussion with peers is fine, but it should be higher-level concepts rather than code snippets.

Starter Code

As with all assignments in this course, you will be given "starter code" that compiles and runs in a browser. This code requires the "Common" folder that you were asked to download as a part of the exercises for Week 02. You should copy the provided code files for Assignment2 into one folder labeled Assignment2. You should place this Assignment2 folder in the same directory as the "Common," "Assignment1," and the other folders. So, your final directory structure should look like this:




« CS_537_Coding › Assignment2
Name
bunny.html
bunny.js
bunny.obj
How_To_Setup_Local_Host_ReadMe.txt
objLoader.js

The starter code will render a black square to the canvas. You will be asked to add various user interaction/input functionalities to the web page, render the bunny provided in the file "bunny.obj" in a particular way, and adjust the bunny's orientation with respect to the camera, along with its size. Your code will also involve animation. Specific goals will be outlined in the section of this document "Goals" (below).

How to turn in your Assignment:

Upload a compressed .zip file to Canvas named as follows: Assignment2_yourusername.zip. The zip archive should contain your Assignment2 folder. This naming system helps us make sure the right assignment is paired with the right individual.

Loading an OBJ File

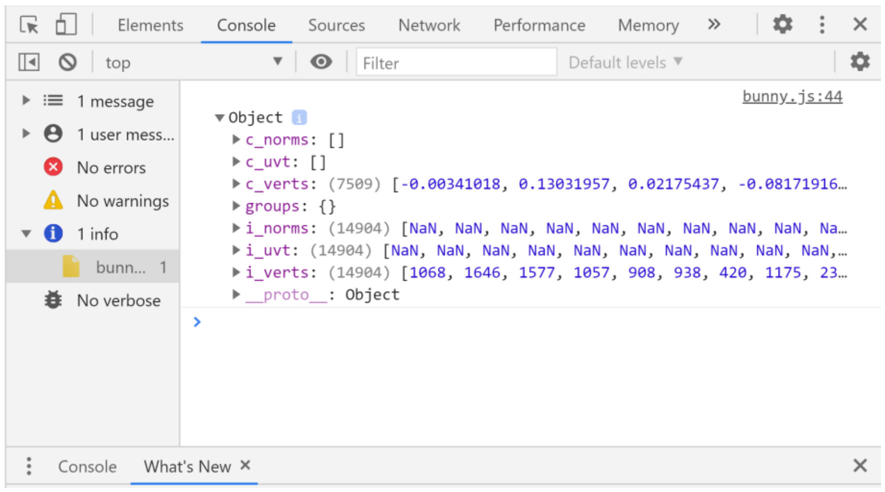
In the Assignment2 directory, you'll find a file called "bunny.obj" that contains the geometry we'll use for this assignment. We covered this file format in Week 03. You can open it in a text editor to explore the layout of this geometry file format. To simplify the assignment for you so that you don't need to parse this OBJ file yourself, you've been provided with an OBJ loader function in the file objLoader.js. Do not write your own OBJ loader; just use the code in this file. **Important**: The starter code that you have been provided already loads the OBJ file contents into memory in the JS application – you shouldn't need to call the OBJ loader functions yourself.

Setting up a Local HTTP Server

In the Assignment2 directory, you'll fine a text file called "How_To_Setup_Local_Host_ReadMe.txt." In order to use assets in your application that come from outside the HTML and JavaScript files, for instance image files or our "bunny.obj" file, you'll need to setup and run a local HTTP server and access your "bunny.html" file this way as you are programming. Follow the instructions in the ReadMe file to setup your local HTTP server (or use any other preferred method of setting up for a different operating system). When we grade the assignments, we'll also be using a local server to facilitate loading of assets/files outside the core HTML and JavaScript files. **Important**: you will not be able to load the "bunny.obj" file unless you first set up the local HTTP Server and access the html file through your localhost address.

Tip: Debugging JavaScript Code

Tested in Google Chrome browser. In your browser, you can right click and select "Inspect" or alternatively type "Ctrl-Shift-I" (Windows OS). In the "Console" tab of the window that pops up, you can find any information that you've logged from your JavaScript application. This example shows the console log of the bunny object after the OBJ loading, from the line "console.log(m);" on line 44 of the starter code JavaScript file. The "c_verts" array contains the xyz coordinates or position for each vertex, and the "i_verts" array contains a list of vertex indices that create faces. In the example shown, the first triangle is formed from vertices 1068, 1646, and 1577. You might note that these are "off by one" compared to the OBJ file; this is because of 0 vs 1 indexing in WebGL vs OBJ files. Our objLoader functions correct the indexing for you, so you can assume these arrays are 0-indexed.



```
                                                          bunny.js:44
▼Object
  ▶c_norms: []
  ▶c_uvt: []
  ▶c_verts: (7509) [-0.00341018, 0.13031957, 0.02175437, -0.08171916…
  ▶groups: {}
  ▶i_norms: (14904) [NaN, NaN, NaN, NaN, NaN, NaN, NaN, NaN, NaN, Na…
  ▶i_uvt: (14904) [NaN, NaN, NaN, NaN, NaN, NaN, NaN, NaN, NaN, NaN,…
  ▶i_verts: (14904) [1068, 1646, 1577, 1057, 908, 938, 420, 1175, 23…
  ▶__proto__: Object
>
```

Goals

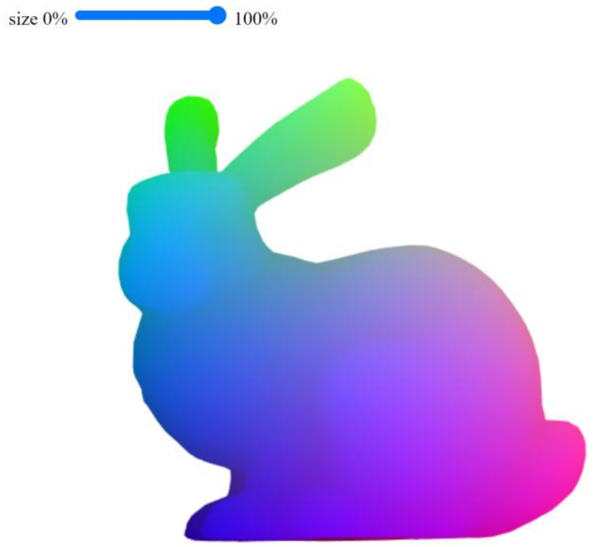
Your goal for this assignment is to adjust the JavaScript file and the vertex shader in the HTML file to generate a program with the following features:

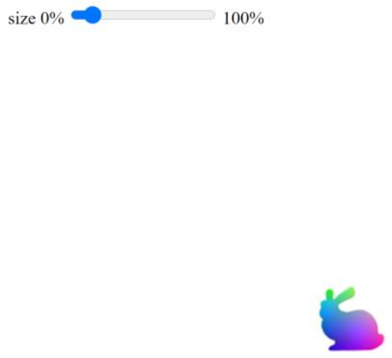
1. **Render the bunny geometry (triangle mesh) from the OBJ file to the screen.** For this, you should investigate and use the function "gl.drawElements" instead of "gl.drawArrays," as gl.drawElements allows you to use an vertex index list as provided in the OBJ file.

2. **Render the bunny using a color-coding system where its positional coordinates (xyz) are mapped to the RGB color cube.** For instance, the largest value in "y" could be mapped to pure green, while the smallest value in "y" could be mapped to "no green." The largest value in "x" could be mapped to pure red, and the largest value in "z" could be mapped to pure blue. Variants of this color-coding scheme will receive

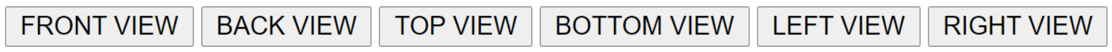
full credit, so long as the color of each vertex is clearly a function of the position. You could compute this positional color-coding scheme in the application or in a shader; the implementation is up to you.

3. **Scaling with a Slider Bar.** Add a slider bar to the interface and implement a feature where the adjustment of the slider changes the scale of the bunny rendered to the display. Your implementation must use the vertex shader to make this adjustment. At both ends of the scale (0%) and (100%), the bunny should mostly fit within the canvas. As you change the slider bar, the bunny's size should automatically adjust.

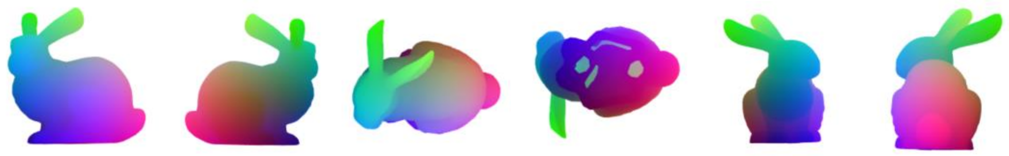




4. **Rotations with Buttons.** Add six buttons that allow you to rotate the bunny so that a different part of the model is facing the virtual camera. Imagine that you have placed the bunny inside a cube, and you want to see the equivalent of the front, back, top, bottom, left, and right face of the cube towards the camera. Each time you click a button, the view should change. Your implementation must use the vertex shader to make the adjustment. (Note: The "Bottom" view might have some hole artifacts as shown below – this is normal)
Buttons:

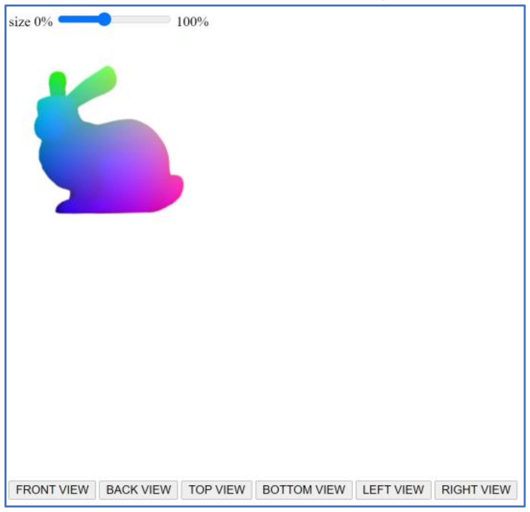


Views:

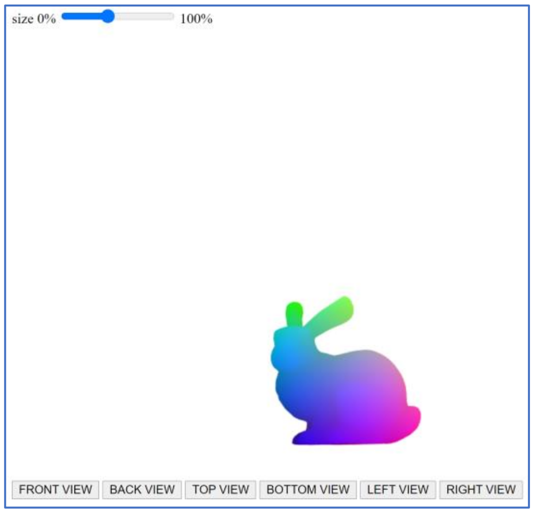


Front Back Top Bottom Left Right

5. **Mouse clicking for Positional Input.** Add a feature so that when you click within the canvas, the bunny moves/is translated to be centered at that new location on the screen. Your implementation must use the vertex shader to make the adjustment.



After clicking in upper left



After clicking in lower right

6. **Keyboard presses for Rotation Animation.** Add the feature so that when you press the “left” and “right” keyboard arrows, the bunny continuously rotates in the appropriate direction around the y axis. Add the feature so that when you press the “up” and “down” keyboard arrows, the bunny appropriately rotates around the x axis. Add the feature so that when you press the “spacebar” key, the rotation stops. (The starter code already includes the event keycodes for these keys.) [See video for demonstration]

7. **Extra credit: (5%) Mouse clicking for position input is “slowly animated” rather than “immediate” or instantaneous.** You can implement functionality that allows the bunny to gradually move in the direction towards the mouse click, and then stop when it has arrived at the mouse click position. [See video for demonstration]

How-To Guide:

There are several functions that we suggest you edit, and these are all marked with //TODO comments in the JavaScript file. You also will need to edit the vertex shader in the .html file. However, you should note that there are several ways to achieve these visual results. This means that you do not necessarily need to follow the //TODO comments; they are merely included as a suggested place to begin. **Note**: there are many helpful/useful functions in Common/MV.js that you are encouraged to use for matrix transformations.

Grading:

Basic Bunny Rendering	**20**
Passes bunny vertices to shader correctly	5
Renders bunny vertices using gl.drawElements	5
Renders bunny with positional color-coding scheme, mapping xyz to RGB	10
Bunny Scaling (Size)	**20**
Adds slider bar to page to adjust bunny size with scaling	3
Adjusts vertex shader to implement scaling adjustment using the slider bar	12
Implements scaling function that maps 0% and 100% of the slider bar to sizes that fit within the window when the bunny is centered in the frame	5
Bunny Rotation Angles (Buttons)	**20**
Adds buttons to page to adjust bunny rotation	4
Adjusts vertex shader to implement rotations using the buttons, yielding visually correct appearance (Partial credit awarded if only some views are correct)	16
Bunny Position (Clicking)	**20**
Adds mouse clicking function to JS code	4
Adjusts vertex shader to implement translation using the mouse position on the canvas, yielding visually correct appearance	16
Bunny Animated Rotation (Keyboard presses)	**20**
Adjusts keyboard event listeners in JS code	5
Adjusts shader/application code so left and right keyboard presses animate rotation around the y-axis, yielding visually correct appearance (partial credit if only left or only right works)	6
Adjusts shader/application code so up and down keyboard presses animate rotation around the x-axis, yielding visually correct appearance (partial credit if only up or only down works)	6
Implements functionality to stop rotation using the spacebar keyboard press	3
Extra Credit	**5**
Bunny moves slowly from starting position to “clicked” position upon a mouse click, adding animation functionality to the “bunny position” input	5
TOTAL	**105**

Partial credit may be given if some parts of code are completed but the visual result is incorrect, at the graders’ discretion.