

Search Topics

Overview

Bookmarks

Course Schedule

Table of Contents

Syllabus

Policies

Assignment 1

Assignment 2

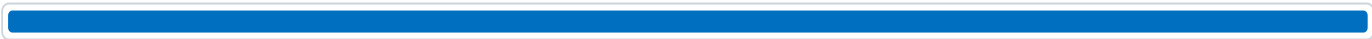
Assignment 3

Study/Practice Material

Midterm Exam


Final Exam

Assignment 2

 Due Nov 7, 2022 11:59 PM 100 % 1 of 1 topics complete

## Assignment 2

Assignment

 Due Nov 10, 2022 11:59 PM

This assignment tests your understanding of hierarchical transformations, the camera, textures, and shaders.

Write a program in JavaScript/WebGL that draws an animated scene with the requirements given below. Template code is provided to handle many of the aspects necessary to get basic objects rendered with textures and moving in a scene. Use the template code found on the course webpage.

**There are important notes about the template code regarding textures below! You will very likely have to address these, so make sure to read them carefully. There is a lot of information here, it is for your benefit. Make sure you read the assignment text from beginning to end. If you have questions consult this assignment text first.**

**In Lab 5 we modified the assignment basecode to support textures in multiple ways. I suggest using this code to get started.**

### Marking Scheme

**Total Marks: 49**

1. **[4 Marks]** At least one two-level hierarchical object (e.g. human arm).
2. **[6 Marks]** Make use of at least two textures either procedural or mapped. You must map them to a(n) object(s) in a meaningful way. Using the textures from the Lab modified assignment base code does not count toward the two.
3. **[5 Marks]** Convert the ADS shader in the assignment base code from a vertex shader to a fragment shader.
4. **[2 Marks]** Convert the Phong to Blinn-Phong in the new assignment base code fragment shader created in step 3.
5. **[5 Marks]** At least one shader edited or designed from scratch to perform a clearly visible effect. Each line of your shader code must be commented clearly explaining exactly what the following line does and why. You must clearly identify the purpose and effect the shader produces in the submitted README.

6. **[4 Marks]** 360 degrees camera fly around using *lookAt()* and *setMV()*.
7. **[4 Marks]** Connection to real-time. You should make sure that your scene runs in real-time on fast enough machines. Real-time means that one simulated second corresponds roughly to one real second.
8. **[2 Marks]** You should display the frame rate of your program in the console window or the graphics window once every 2 seconds.
9. **[5 Marks]** *Complexity*: scene setup and design, movement of animated elements, and programming.
10. **[5 Marks]** *Creativity*: storytelling, scene design, object appearance and other artistic elements.
11. **[5 Marks]** *Quality*: Attention to detail, modelling quality, rendering quality, motion control.
12. **[2 Marks]** Programming style.
13. **[-2 Marks if not]** Make and submit a movie of your animation. The movie should be at least 512x512 resolution and in a standard format, such as mp4. Include a cover image (png or jpg) of at least 512x512 pixels. You may use any screen capture program that is available (e.g. ShareX).
14. **[-4 Marks if not]** Provide a readme.txt that describes what you have done, what you have omitted, and any other information that will help the grader evaluate your work, including what is stated below.

## Requirements/Policies

### Collaboration

None. If you discuss this assignment with others, you should submit their names along with the assignment material.

### Original Work

The assignment must be done from scratch. Apart from the template provided, you should not use code from any other source, including the previous offering of the class.

Note there are many shader sources on the internet. Please develop your own first! You can seek inspiration but **do not** use others' code.

### Zero Mark

If the code does not run, or no objects appear in the window, or only the template code is running properly, no partial marks will be given.

### Clarifications

- Note that creativity and complexity are both important for this assignment.

- Make sure you rotate the body parts around the correct point, i.e. where they touch the parent body parts, so bodies do not appear to break apart.
- Start your code in `main.js:render()` and feel free to write additional functions
- You must do the assignment from scratch. If you use existing code (e.g. A1) then you must state that clearly in your `readme.txt` file. Using any piece of code from any source (including previous offerings of the course, A1, the web etc) without attribution will be considered plagiarism.
- You can use libraries as stated below. You will not get credit for the code you have not written, however, it might help with the visual complexity of your scene. For example, if you use an existing model for required element 1, then you will not get credit for it. See below for more details.

### Project Submission Instructions

1. Make a backup copy of your project folder first to avoid accidentally deleting your work.
2. Rename the parent project folder to include your name
3. Remove unnecessary files from the project.
4. Archive the project folder in a zip folder and call it `csc305_assignment_2.zip`.
5. Test the archive
  - a. Unzip the archive in a different location.
  - b. Make sure everything works as expected.
6. Submit the zip file on brightspaces.

We want to be able to open your zip file, run, and see your project right away. Also, note that you can add as many files as you want to the project as well as modify any settings that you need to (see below). However, it would be useful if you stated unusual settings and additional files in the `readme.txt`.

### Movie Submission Instructions

1. There are lots of screen capture software that simplify screen recording. On Windows, ShareX is very good.
2. Call your movie file `<first>_<last>_movie.[mpg,mp4]`, replacing `<first>` and `<last>` with your first name and last name.
3. Call your image file `<first>_<last>_image.jpg`.
4. Zip the two files in an archive called `<first>_<last>_bundle.zip`.
5. Submit the file on brightspaces.

### Security Issues with Textures

If the textures do not work, check the console for an error message related to "Cross-origin image". To use textures in WebGL we have to bypass a security issue that is present with most browsers.

The easiest way to handle this is with the python HTTP server approach we discussed in Lab.

1. Open explorer/file manager and navigate to the code folder.
2. Now shift-click in the file list and open a command prompt or powershell here
  1. Alternatively, you can open a command prompt or power shell from the windows menus and then use `cd [folder]` to navigate to the correct folder.
3. Now run the following:  
`python -m http.server 8080`
4. Now you should be able to open a browser and type `localhost:8080`  
In the URL bar
5. You'll see your files being hosted on a simple local server, you can now click your html file.

## Firefox

1. Go to: `about:config`
2. Find: `security.fileuri.strict_origin_policy` parameter
3. Set it to false

## Chrome

Close all running chrome instances first. Then start Chrome executable with a command line flag (strongly suggest making this a desktop icon for easy reuse):

`chrome --allow-file-access-from-files`

For more details and other methods see:

<https://github.com/mrdoob/three.js/wiki/How-to-run-things-locally>

## FAQ

1. Are there any size limitations to the project? i.e. number of classes, project size, memory, number of textures, texture resolution etc?

**A:** In terms of texture resolution, number of textures etc, there are:

- a. Hardware and software limitations. There is a maximum number of textures you can have active at the same time, and there is a maximum size for textures.
  - b. Practical limitations. Brightspaces has a 1gb limit on what you can upload and a **recommended 40mb maximum**. You also want your project to run in real-time.
2. What should be the duration of the movie?  
**A:** However long to capture it generally, if it loops you may want to capture two or three loops. Often these are between 20-60 seconds.

### 3. Can we use images from the web?

**A:** As long as the images are free for public use, sure. Do not use assets that are copyrighted. Note that the filters on google images are rarely correct w.r.t usage rights. Check out resources like:

a. Wiki Commons

i. [https://commons.wikimedia.org/wiki/Main\\_Page](https://commons.wikimedia.org/wiki/Main_Page)

b. PARIS MUSÉES

i. <https://www.parismuseescollections.paris.fr/en>

c. Smithsonian Open Access

i. <https://www.si.edu/OpenAccess>

### 4. Can we use external libraries?

**A:** As long as they (a) are free for public use, (b) do not make the required elements unfairly easier for you (do not use an engine, tool, package, module, or any resource that solves a required element for you), and (c) the TA can run your project, you can use additional libraries. However, if the TA cannot run your program because of missing dependencies etc. you will have a problem. Generally, that means they need to be freely available and you need to submit the complete assignment. No setup or downloading on our part should be required to get it to run.

### 5. Can we use audio?

**A:** You can, however 3) also applies here.

### 6. How ambitious can we be?

**A:** I encourage you to take this opportunity and explore your programming (additional libraries, effects etc.) and your creative interests. However, I advise you not to aim too high, unless you are confident about what you are doing. Above all, FIRST make sure that you have covered the required elements of the assignment.

## Hints

- Use the timestamp or dt variables to synchronize your animation elements.
- Use setColor(r,g,b) to set the desired colors.
- For the motions, you may want to use functions such as
  1.  $x(\text{timestamp}) = A * \text{Math.cos}(w * \text{timestamp} + h)$ 
    - Where timestamp is real-time, A is amplitude, Math.cos is cosine function, w is the angular frequency, h is the phase

1.  $x(t + dt) = x(t) + x'(t)dt$

- Next position in time (  $x(t + dt)$  ) is just the current position (  $x(t)$  ) plus velocity (  $x'(t)$  ) scaled by the change in time (dt).
- Wave functions
- Euler integration for movement
- Common mistakes with motion
  1.  $\text{eye}[0] = \text{eye}[0] * \text{Math.cos}(\text{theta}) + \text{eye}[2] * \text{Math.Sin}(\text{theta}) ;$
  2.  $\text{eye}[2] = -\text{eye}[0] * \text{Math.sin}(\text{theta}) + \text{eye}[2] * \text{Math.cos}(\text{theta}) ;$  // use of the new  $\text{eye}[0]$  instead of the original  $\text{eye}[0]$
- Camera Motion