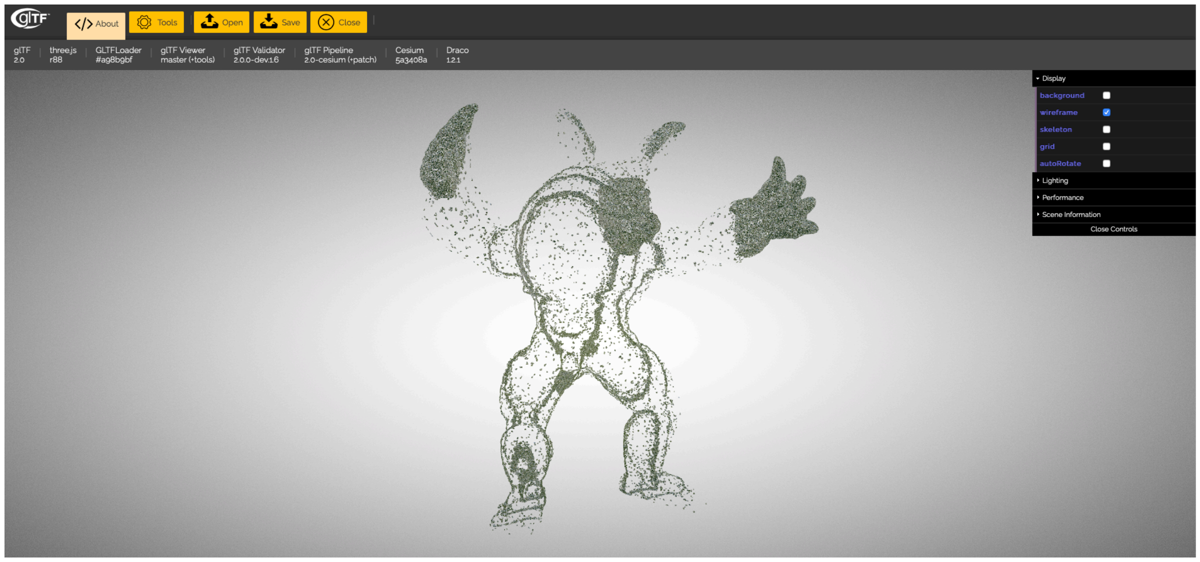
CS460 Fall 2020
Github Username: JamesEdMichaud
Due Date: 12/07/2020

Assignment 10: glTF!

We will load our favorite mesh from a file and then convert it to a valid glTF file. You can choose if you want to do this assignment in JavaScript or in Python. In class, we will use Python (see example colab `https://cs460.org/shortcuts/33/`).



Starter code for assignment 10. After pulling from upstream, there is the folder `10` in your fork. This folder contains an `index.html` file that uses JavaScript to make glTF JSON. This folder also contains a `gltf.py` script that you can run with `python gltf.py` to output the glTF JSON. As a start for this assignment, both versions create an identical valid glTF JSON structure holding a single triangle (see screenshot above).

Part 1 (1 points): Please decide which language you will use: JavaScript or Python. Python might be a bit easier to load and parse an existing file—with JavaScript we need to use Ajax to load the existing mesh and parse it (or as option 3: use a Three.js loader and grab the vertices/indices from there). For parsing files with Python look here: `https://tutorial.eyehunts.com/python/python-read-file-line-by-line-readlines/` For using Javascript and Ajax look here: `https://developer.mozilla.org/en-US/docs/Web/API/XMLHttpRequest/Using_XMLHttpRequest`.

Part 2 (15 points): Load a mesh from an external file. A .PLY or .OBJ file might be the easiest to parse.
I chose to write a .PLY parser, but nothing about it was easy. And yes, I know it's not technically done right, but I think I'm close enough to get partial / most credit (please!)?

Part 3 (20 points): Parse all vertices from the loaded mesh and create the VERTICES array and base64 code.
This part wasn't too bad. After a few hours of reading format specifications and numpy documentation I looked at several .ply files and found that each one has vertices in the same general structure (3 floats). I wrote a function called 'readNoListProperties', which will read 3 * rowCount floats using np.fromfile (binary) or np.loadtxt (ascii) and return the data that was read.
The 'NoList' part is a result of working through parsing the rest of the .ply file. Some elements have a 'list' property, which complicates things significantly (see next part).

Part 4 (20 points): Parse all indices from the loaded mesh and create the INDICES array and base 64 code.
This part was a bit frustrating. The .ply specification say that any element (vertex, face, etc) may be composed of a variable number of properties. One of those properties may be a 'list' of values. That list may vary in length from row to row, and the length of the list is not specified until the beginning of each row. This means each row needs to be parsed individually, significantly slowing down parsing speed. Properties may be a list, float, int, etc.
I used armadillo.ply to start building my parser, and the faces (indices) element is composed of two properties: an unsigned char representing intensity and a list of indices. Example row:
0 3 1 2 3
these numbers represent:
intensity, count of indices in this row, index1, index2, index3

To handle this I wrote two functions: 'readBinaryListProp' and 'readAsciiListProp'. These functions will be called any time there is an element that contains a 'list' property, and in all the ply files I looked through that means the 'faces' element, which contains the 'vertex_indices' property (which is a list). The functions read each property, and if it is a 'vertex_indices' property it goes into one array, and if it is any other kind of data it gets placed in a different array (both will be returned). Since we're not using any of the other data in this assignment it gets discarded after being returned.

Part 5 (10 points): Calculate all required fields for the glTF file (as we did in class) and generate the glTF JSON code. Store the glTF JSON code in a glTF file.
I wrote a bunch of functions to bring everything together, so apologies for the 200 lines of python code.

Part 6 (5 points): Please make sure the glTF file is valid using `http://github.khronos.org/glTF-Validator/`.
I found out that I needed to flatten my 2D array for validation to work properly. However, I still encountered some issues. The validator tells me "Declared minimum value for this component (0) does not match actual minimum (1).", which isn't true at all. I think some data may be be getting left out. I'm have a feeling it has something to do with the base64 encoding, but after several hours of debugging I'm not any closer to figuring it out. When I load the armadillo there are many triangles missing. The point lights I used in assignment 9 add a pretty cool effect though (see index.html). It's a very neat effect, but not at all what it should be.
I tried converting a cube file with only 8 vertices and got the message:
"Number of vertices or indices (8) is not compatible with used drawing mode ('TRIANGLES')."
I'm not sure how that's possible, considering there are exactly 8 vertices in a cube and 12 triangles (2 per face) that make it. Both files, along with a 'teapot' file are in my assignment 10 folder.

Part 7 (5 points): Visualize the glTF file using `https://gltf.insimo.com/`. You might have to choose the wireframe display option since the glTF file does not include material (Display -> Wireframe, in the dat.GUI). **Please replace the screenshot above.**

Part 8 (5 points): Add the glTF file to your fork.

Part 9 (10 points): Choose a final project—either an existing one from `https://cs460.org/assignments/final/` or a new one. Please list the project here and in the link. If working as a team, assemble your team and list the team members below and in the link.

Fix the local alignment bug in XTK! Awww, yeah! I was going to try to implement a molecular dynamics viewer, but as I was planning some of it out it began to get scary, given the amount of time we have left. I'd rather work on something that actually gets used!

Part 10 (9 points): Make sure this PDF and your glTF file are in your fork on github. Then, please send a pull request.

Bonus (33 points):

Part 1 (15 points): Please add any kind of material to the glTF file. For this, you would have to read the specs or google for examples :)
Sorry, no extra credit here for me this time.

Part 2 (18 points): Write THREE.js code that displays your glTF file using the `THREE.GLTFLoader`.
I did this one though! Check out index.html.