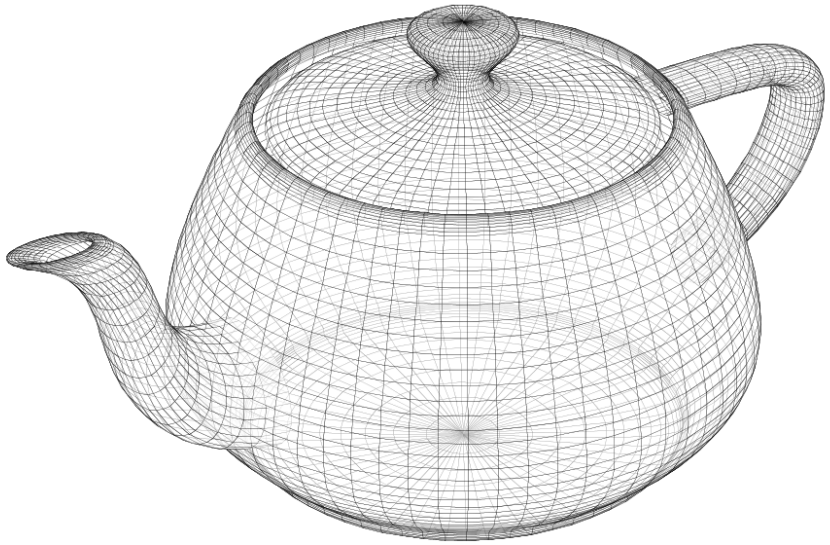
Reading an OBJ File



Interactive Computer Graphics

Professor Eric Shaffer

"All it takes is for the rendered image to look right."
—Jim Blinn

ILLINOIS

Goals

We will go over implement code to read a mesh from an OBJ file

Things you will need to do include:

- Learn how to asynchronously fetch a server side file
- Handle an asynchronous event
  - So you don’t try to draw before you have the mesh ready
- Parse the OBJ text file using JavaScript
  - So you can populate the vertex and face buffers for the mesh

All of this code will be used in MP3

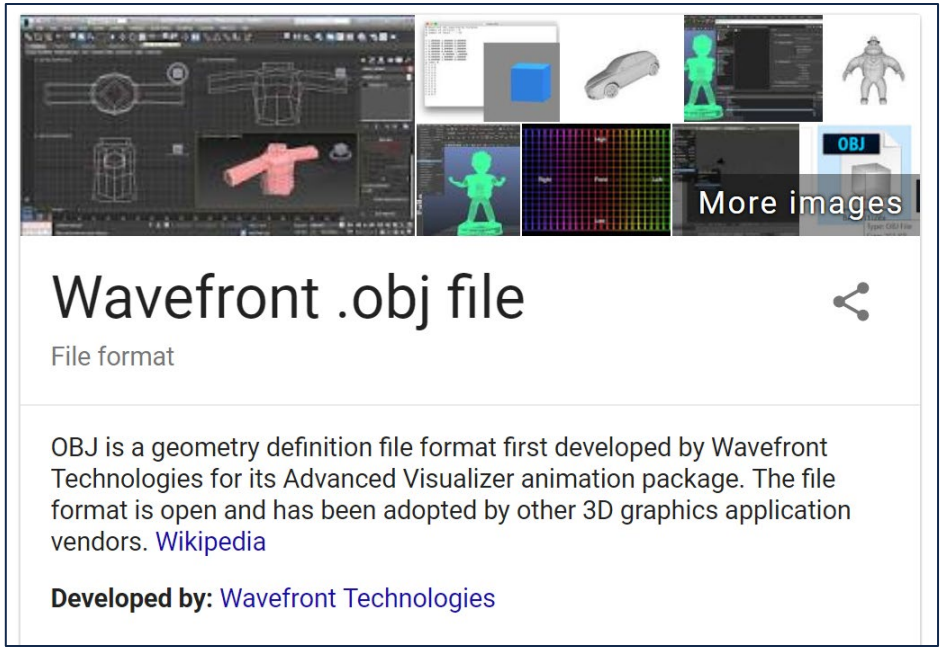
OBJ File Format

It is a text (ASCII) file format for 3D surface models

You will implement a parser for a subset of the format

…and render a cow

For details on the file format:
https://en.wikipedia.org/wiki/Wavefront_.obj_file



ILLINOIS

Asynchronously Read a File

It would probably be more useful to read and render a file on the client side…but for this MP, grading will be e easier if we read a server side file…

To start, we need to be able to read a text file

We will write code to fetch an obj file kept on the server side

The only slightly tricky part is that the fetch is done *asynchronously*

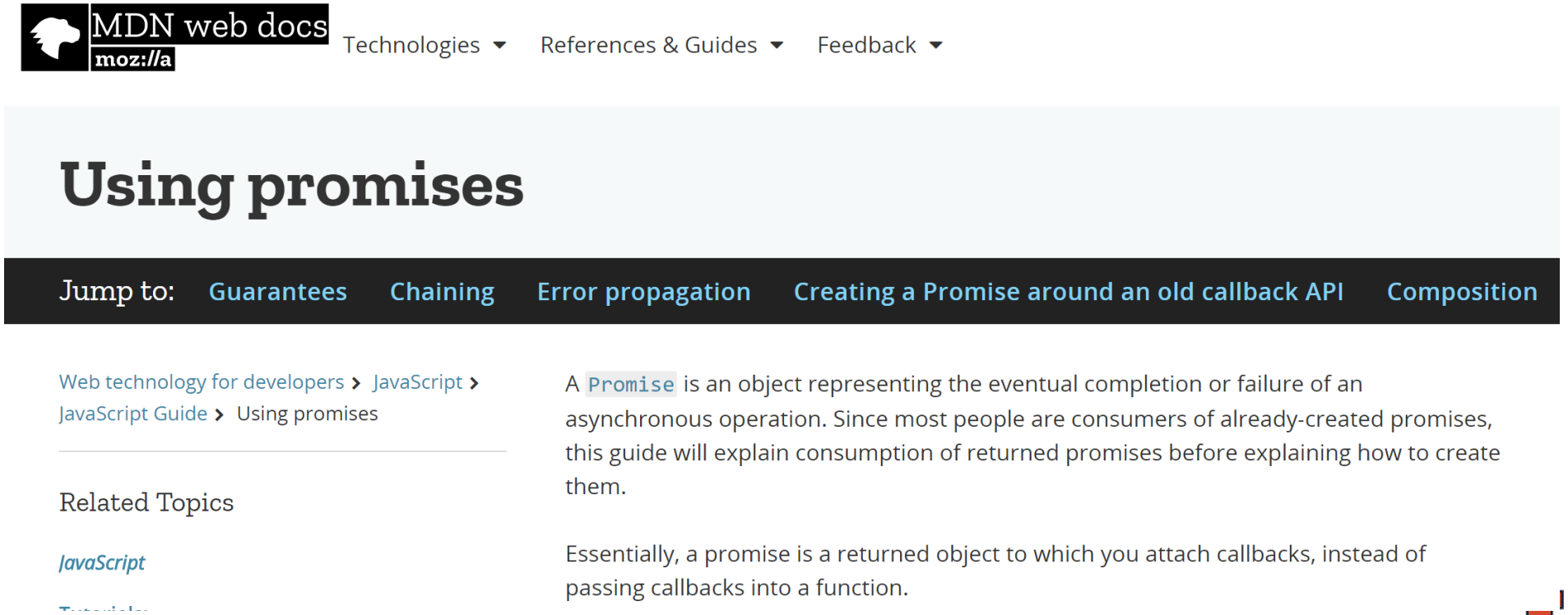
- Since the file read is asynchronous…
- We need to not try to draw the mesh before the data is ready

Promises

To get the text asynchronously from the server we will use a JavaScript promise

You can read about them:

https://developer.mozilla.org/en-US/docs/Web/JavaScript/Guide/Using_promises



ILLINOIS

Function setupMesh

Use this code in whatever your main file is the (the one that sets up buffers and renders).

Retrieve text from a function the reads a file and give it to Trimesh loadFromOBJ to be parsed

```
//--------------------------------------------------------------------------------
/**
 * Populate buffers with data
 */
function setupMesh(filename) {
    myMesh = new TriMesh();
    myPromise = asyncGetFile(filename);
    // We define what to do when the promise is resolved with the then() call,
    // and what to do when the promise is rejected with the catch() call
    myPromise.then((retrievedText) => {
        myMesh.loadFromOBJ(retrievedText);
        console.log("Yay! got the file");
    })
    .catch(
        // Log the rejection reason
       (reason) => {
            console.log('Handle rejected promise ('+reason+') here.');
        });
}
```

ILLINOIS

function asyncGetFile(url)

Here's the code to read a file….

```
//-----------------------------------------------------
/**
 * Asynchronously read a server-side text file
 */
function asyncGetFile(url) {
  console.log("Getting text file");
  return new Promise((resolve, reject) => {
    const xhr = new XMLHttpRequest();
    xhr.open("GET", url);
    xhr.onload = () => resolve(xhr.responseText);
    xhr.onerror = () => reject(xhr.statusText);
    xhr.send();
    console.log("Made promise");
  });
}
```

ILLINOIS

Don’t Draw the Mesh Before It Is Ready!

Stop `myMesh` from being drawn before it is ready

…add an if statement to prevent this in your draw function

Which TriMesh function will be useful?

loadFromOBJ(fileText) in TriMesh.js

Finally…write code to parse the text file….

```
/**
* Populate the JS arrays by parsing a string containing an OBJ file
* @param {string} text of an OBJ file
*/
loadFromOBJ(fileText)
{

    //Your code here









    //----------------
```

ILLINOIS

Parsing the OBJ File

You just need to parse a subset of the OBJ file format…

- Lines starting with # are comments…log these to the console
- Lines staring with v are vertex coordinates

```
v 0.123 0.234 0.345
```

- Lines starting with f are triangles

```
f 1 2 3
```

- **NOTE..obj vertex indices start at 1…you need to subtract one if your arrays will start at 0**

To make things easier, assume faces have only 3 numbers in each record
…not true for full OBJ format

Read those lines and fill this.vBuffer and this.fBuffer

ILLINOIS

Useful JS functions

- String method split()

  The `split()` method splits a `String` object into an array of strings by separating the string into substrings, using a specified separator string to determine where to make each split.

  https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/String/split

- parseFloat

  The `parseFloat()` function parses an argument and returns a floating point number.

  https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/parseFloat

- parseInt

  The `parseInt()` function parses a string argument and returns an integer of the specified radix (the base in mathematical numeral systems).

  https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/parseInt

Important MP3 Note

mp3 OBJ parsing

For mp3 OBJ parsing, splitting on a single whitespace char works for the cow but not the teapot which must have some lines in which multiple spaces are separators.

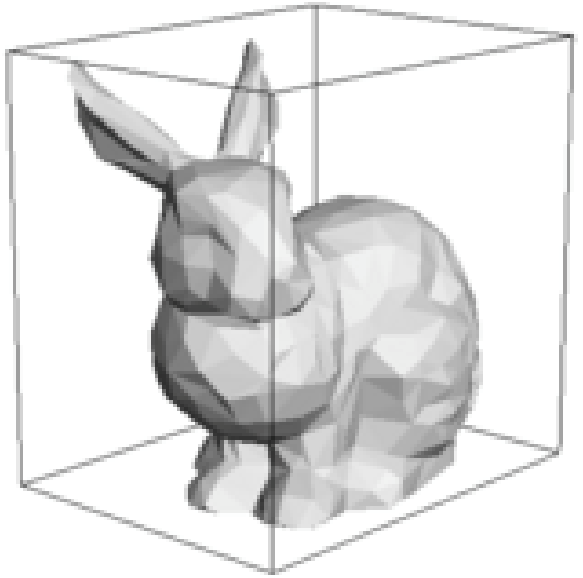
Try amending the parsing code in TriMesh.js to use

```
var tokens = lines[i].split(/\b\s+(?!$)/);
```

ILLINOIS

Axis Aligned Bounding Box

- Box is defined by
  - min point $p_0=(x_0,y_0,z_0)$
  - max point $p_1=(x_1,y_1,z_1)$
- Box is $[x_0, x_1] \times [y_0, y_1] \times [z_0, z_1]$
- How can we efficiently compute the box?
  - Imagine you are given a bunch of triangles
  - What is the bounding box for all those triangles?



Model Size and Position

The cow mesh AABB:

```
TriMesh: Loaded  5804  triangles.
TriMesh: Loaded  2904  vertices.
AABB:
-0.5221959948539734,-0.3198379874229431,-0.17014099657535553
0.5221959948539734,0.3198379874229431,0.17014099657535553
```

The view and projection matrices are set up to see that geometry

Other meshes might not naturally be located in your view volume

- May need to be translated and scaled

In TriMesh.js complete the functions shown here

…you can use that information to determine how a mesh should be transformed

ILLINOIS

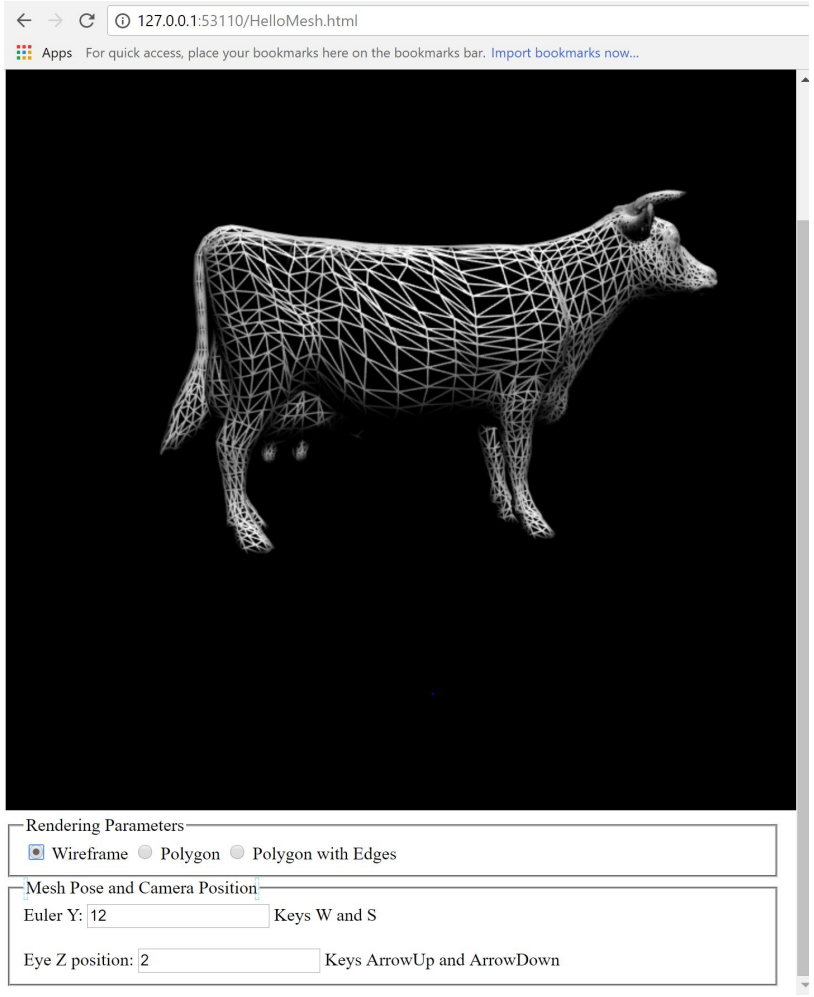
TriMesh.js:

Complete `computeAABB()` and `getAABB(minXYZ,maxXYZ)`

- You can then use these functions in code that
  - Determines if the geometry should be scaled
  - Determines if the geometry should be translated
  - You would apply those tranformations to the MVMatrix before drawing

ILLINOIS

Results…



ILLINOIS