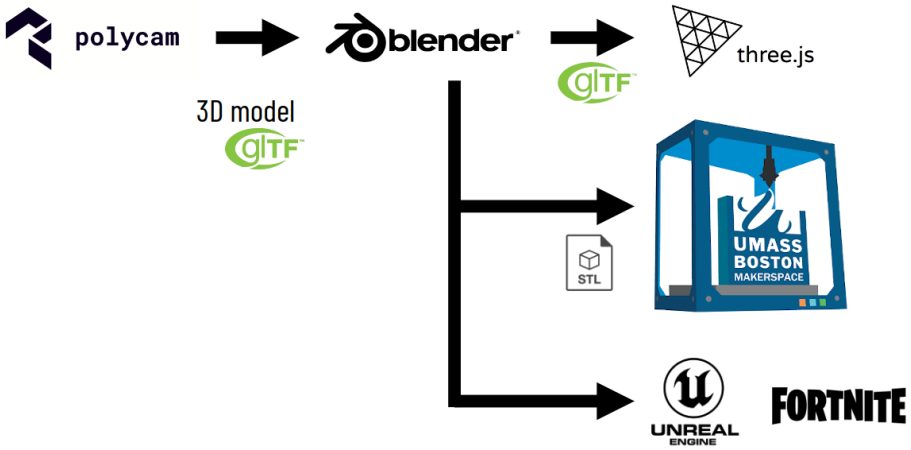
In partnership with CreateAccess Empowering Futures, One 3D Creation at a Time

Assignment 5: CreateAccess.org meets CS460 - From Realworld to 3D!

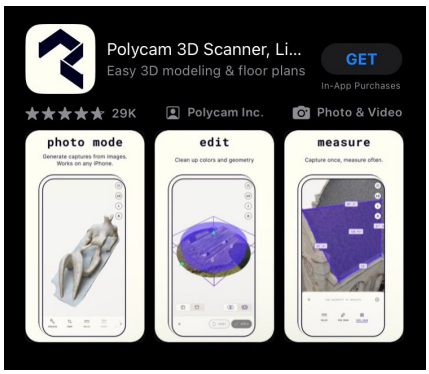
We will use PolyCam + Blender + Three.js to render a realworld object and then 3D print it!



In class, Patrick and Liz introduced their work on democratizing computer graphics education - a very important effort! As part of this assignment, we will expand their CreateAccess.org Photogrammetry microcourse! First, we will use the PolyCam app to create a digital version of a real world object. We will then use the popular 3D modeling software Blender to edit the scanned geometry. After, we will use Three.js to render both the original and then edited version. Finally, we will leverage the excellent UMB MakerSpace to 3D print the edited geometry and, if you are up for it (Bonus!), add it to a Fortnite island using the Unreal Engine.

Part 1: Create a 3D model using PolyCam! (30 points)
Please follow the CreateAccess guide on scanning a real world object with the free version of PolyCam (like I did with the croc!).

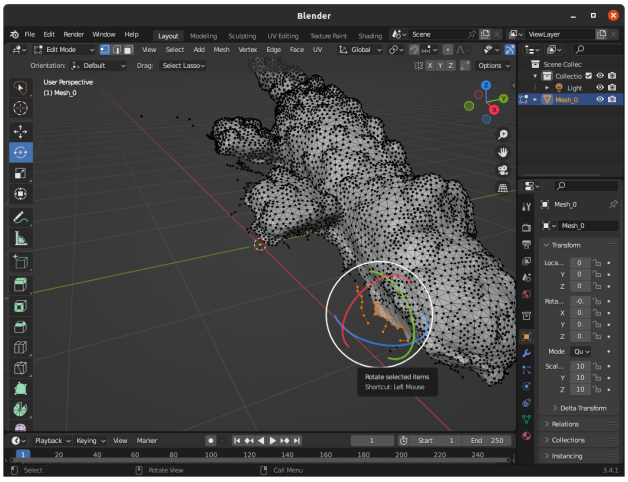
YouTube: `https://www.youtube.com/watch?v=nUb5bacXnRs`
PDF: `https://drive.google.com/file/d/1RZiJXwQnU0qRfGNm_a6sTaVCk2ZVQVP1/view`

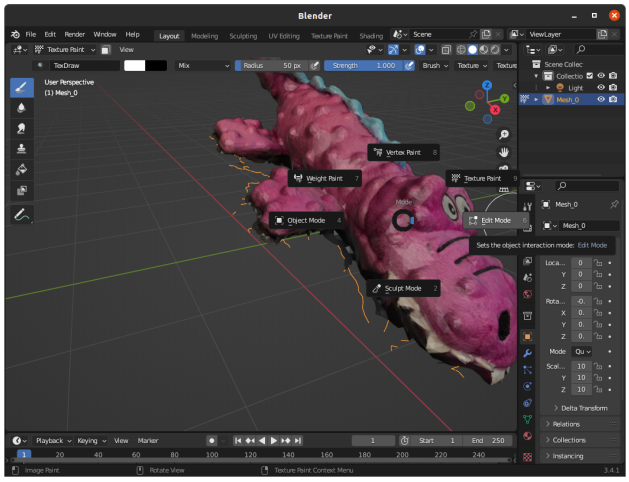


Part 2: Edit the 3D model using Blender! (60 points)
First, we will need to download and install Blender which is available for all platforms at `https://blender.org`! Then, please follow the CreateAccess guide on editing the model that you scanned in Part 1. I used Dropbox to transfer the `poly.glb` file from my phone to my computer but any other way is fine too.

YouTube: `https://www.youtube.com/watch?v=RCRzv0_LMHc`
PDF: `https://drive.google.com/file/d/1pUoiRGyunjLEaZHJutlSXadjaisrpQtk/view`

This part involves switching between the Object Mode and Edit Mode in Blender to rotate the mesh and then to remove vertices (the guide above describes using the Lasso, Box, and Circle tools—all are fine to use). **Hint:** CTRL+Tab allows to switch between these modes quickly and also allows to visualize the texture of the model in Texture Paint mode (screenshot right). This can be helpful to see which areas should be edited to remove the background/floor of the model—goal is to isolate the object as much as you can. But for the editing, you will need to switch back to Edit mode (screenshot left).





Important: Once finished, please make sure to **export the changed model twice**: 1) as a glTF binary file (`.glb`) and 2) as a Stl (`.stl`) file. **Also, please save the Blender scene (`.blend`).**

Part 3: Three.js visualization! (60 points)
Please copy 4 files to your `05/` folder in your `cs460student` fork:

1. `poly.glb` (the original mesh acquired with Polycam)
2. the `.glb` and `.stl` files from part 2.
3. the `.blend` file from part 2.

Now, create an empty `index.html` in that folder and follow these steps—**no starter code this time!**

1. Setup a Three.js boilerplate scene like we did in Assignment 3. You will need to setup the Three.js imports and include addons as well because we will use the `OrbitControls` and `AnaglyphEffect` again but also others :) Please don't forget the CSS to stretch the `<canvas>` across the whole window with no margin or padding and a black background. After this step, you should have the `window.onload` method and the `animate` method set up including camera, renderer, and scene—but nothing shows up yet (hopefully also no console.log errors!).

2. Next, please configure one directional light and one ambient light (again, like assignment 3). Still nothing shows up..

3. Now, we will use the `GLTFLoader` to load the `poly.glb` file. After adding a new import...

```
import { GLTFLoader } from 'three/addons/loaders/GLTFLoader.js';
```

...we can use the following snippet to load a binary glTF file and add it to the scene:

```
loader = new GLTFLoader();
```

```
loader.load( 'poly.glb', function ( gltf ) {

  scene.add( gltf.scene );

} );
```

4. But.. you might not see it yet. Please play around with the camera position and the scaling of the mesh in the loader callback to visualize it properly. For me, a `zNear` of 0.1, `zFar` of 10000 and a `camera.position.set(0,10,0);` helped but only adding this code to the loader callback did the trick:

```
var poly = gltf.scenes[0].children[0];

poly.scale.x = 10;
poly.scale.y = 10;
poly.scale.z = 10;
```

At this stage, your unedited PolyCam mesh should appear!

5. So let's add the edited version next to it. Copy another GLTFLoader call with callback but this time, please load your edited Mesh from Blender. If you scanned a pink crocodile, you might have called it `croc.glb`. And now, you might notice that this mesh is strangely placed and has different scaling than the PolyCam mesh and might be also overlapping. This is because the `.glb` file from Blender includes all rotations of the camera and the mesh itself when exporting. Let's fix that..

6. We should overwrite the quaternion for both meshes with the identity and set the scaling for both to factor 10 like above. This is especially important if you followed the Blender editing instructions carefully and changed the scale there for easier editing. For the quaternion, you can use this snippet in the callback but replace the ? with the correct values for an identity quaternion to remove the rotation for both loaded meshes.

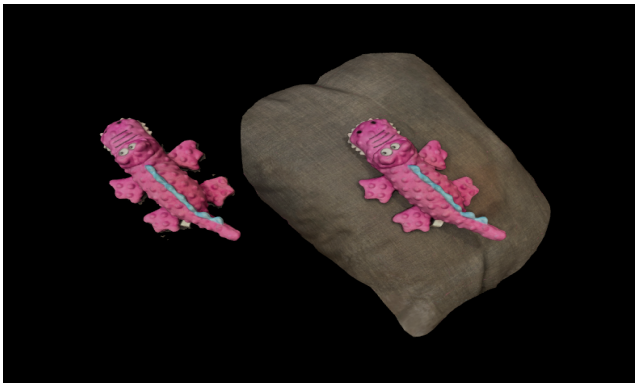
```
var poly = gltf.scenes[0].children[0];

// ... scaling like in 4. above

poly.quaternion.w = ?;
poly.quaternion.x = ?;
poly.quaternion.y = ?;
poly.quaternion.z = ?;
```

I also had to add a small translation to the first mesh to avoid any overlap: `poly.translateX(5);`—please play around with the translation to make it look good :)

7. If the PolyCam mesh and the edited mesh from Blender now appear next to each other, you are golden and can move on :) If not, please figure out good values for the transformation and camera to make this happen.



At this stage, the PolyCam mesh and the edited Blender mesh should appear next to each other!

8. Let's add some user interface with `dat.gui`!

TODO COMING SOON

9. Scene toggles and parameters: Anaglyph, Light X/Y/Z and intensity, and the AmbientLight color.

TODO COMING SOON

10. The PolyCam Mesh panel with wireframe setting and our 180 degree rotation.

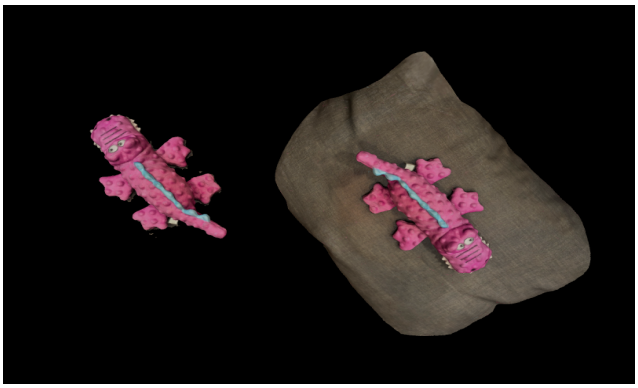
TODO COMING SOON

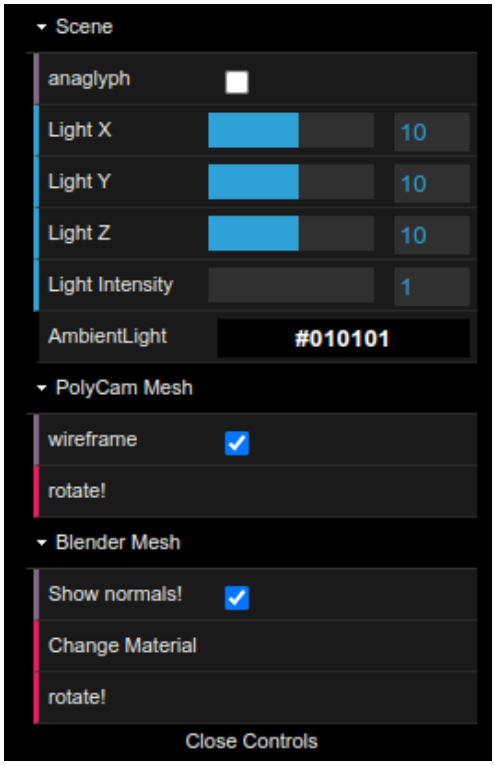
11. The Blender Mesh panel with normal visualization and the MeshNormalMaterial and of course, another 180 degree rotation!

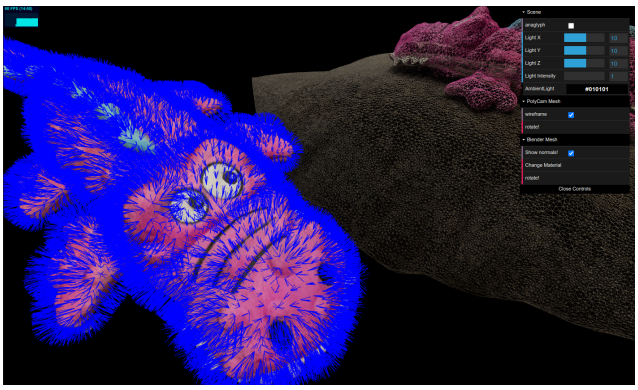
TODO COMING SOON

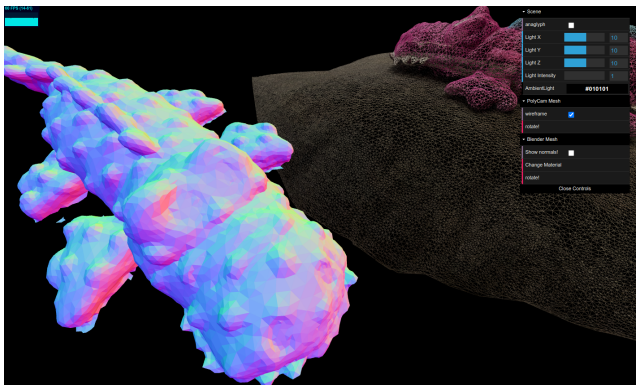
12. The `stats.js` rendering performance indicator.

TODO COMING SOON









Part 4: 3D Printing in the UMB Makerspace! (40 points)
Warning: This part will take around 48 hours so please plan ahead :)

	Monday			Tuesday			Wednesday			Thursday			Friday		
8:00 AM															
8:30 AM															
9:00 AM					OM					OM					
9:30 AM					OM					OM					
10:00 AM	CL	OM		CL	OM		CL	OM		OM			OM		
10:30 AM	CL	OM		CL	OM		CL	OM		OM			OM		
11:00 AM	CL	OM	SB	CL		SB	CL	OM	SB		SB		OM	SB	
11:30 AM	CL	OM	SB	CL			CL	OM	SB				OM	SB	
12:00 PM	CL	OM		CL			CL	OM					OM		
12:30 PM	CL	OM		CL	OM		CL	OM		OM			OM		
1:00 PM	CL		SB	CL	OM		CL		SB	OM				SB	
1:30 PM	CL		SB	CL	OM		CL		SB	OM				SB	
2:00 PM	CL		SB	CL	OM		CL			OM				SB	
2:30 PM					OM					OM					
3:00 PM					OM					OM					
3:30 PM					OM					OM					
4:00 PM															
4:30 PM						SB			SB		SB				
5:00 PM						SB			SB		SB				
5:30 PM															
6:00 PM															
6:30 PM															
7:00 PM															
7:30 PM															
8:00 PM															
8:30 PM															

Last Updated: 2023-09-11

NOTE: This is our official schedule, however we are not locked into these hours. We may need to leave early or come in late for one reason or another

Feel free to email us at makerspace-staff@umb.edu or call at 617.287.7098 to see if we're here!

CL: Christian Lopes
OM: Olivia Moos
SB: Sydney Bailey

The schedule above shows the open hours of the MakerSpace. As a CS460 student, you can drop in and bring your `.stl` file from Part 2 either on a USB stick or **download it through your github pages link** right on the MakerSpace workstations. Tell staff that you are part of this course and would like to print a 3D model from an `.stl` file—they will be happy to help you. But, if the MakerSpace is too busy at the time of your visit, you might be better off coming another time or schedule with them via email. **Once printed, please take a picture of your model!**

Part 5: Submit the Form! (10 points)
Finally, please push everything to your fork if you have not done it yet! As always, please double-check Github Pages to see if the Part 3 loads! Then, please send a pull request and fill in the form at `https://cs460.org/assignments/05/` including screenshots and pictures but don't submit it yet if you plan on getting some bonus points!

Bonus: Import your model to Fortnite Islands with the Unreal Editor for Fortnite! (66 points)
For the bonus, please follow the CreateAccess guide on how to import the 3D model into Fortnite with the Unreal Editor for Fortnite (UEFN):

YouTube: `https://www.youtube.com/watch?v=RkEEHn253yg`
PDF: `https://drive.google.com/file/d/1U3PGz2MW_7NRe7DJAtvcWi8OSTBvbfUB/view`

By the way, instead of editing the model with Blender, we could have done that with UEFN as well. Patrick and Liz have a guide for that on CreateAccess.org! And UEFN is extremely powerful, see `https://dev.epicgames.com/documentation/en-us/uefn/unreal-editor-for-fortnite-documentation` and `https://dev.epicgames.com/documentation/en-us/fortnite-creative/fortnite-creative-video-tutorials`. Please don't forget the screenshot once you are done and then the form submission (Part 5 :).