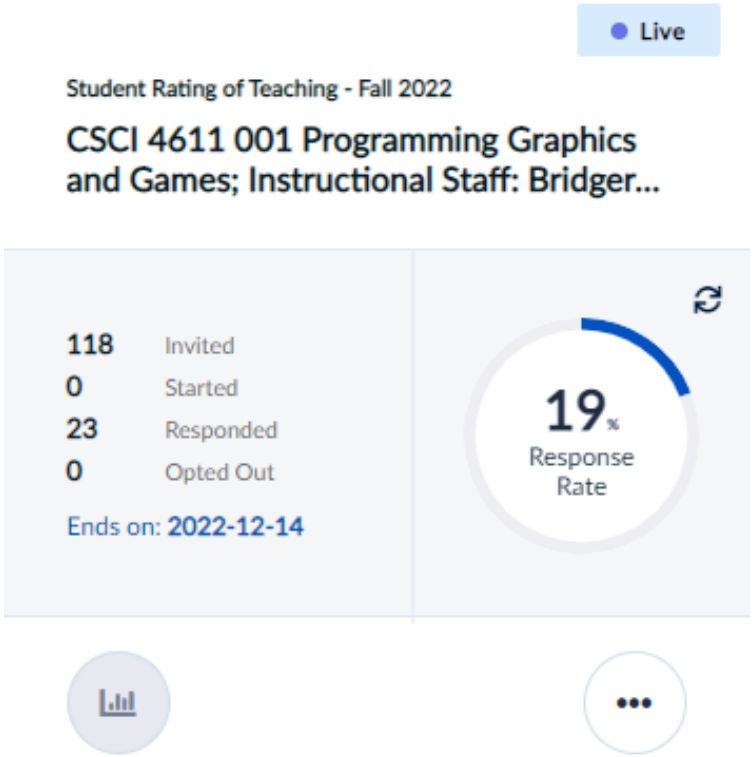
UNIVERSITY OF MINNESOTA
Driven to Discover®

Wrap Up

CSCI 4611: Programming Interactive Computer Graphics and Games

Evan Suma Rosenberg | CSCI 4611 | Fall 2022

Reminder: Complete the SRT!



The SRT closes **tomorrow**.

Bonus Quiz

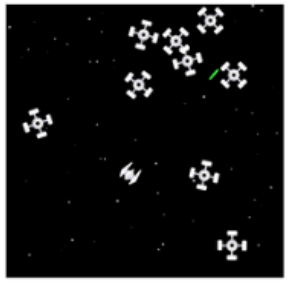
The optional bonus quiz is now available on Canvas.

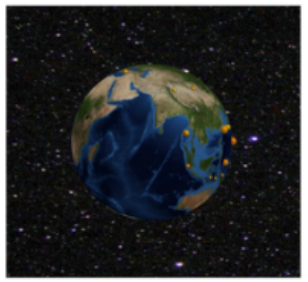
It is due on **Tuesday, December 20 at 6pm**.

Everyone has been issued a 0 on the bonus quiz by default.
Canvas has been set to automatically drop your lowest quiz grade.

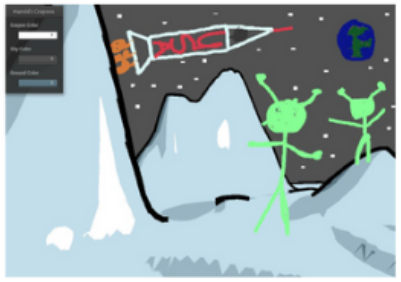
Recap: First Half of Semester

- Structure of interactive graphics programs
- Points and vectors
- 2D/3D transformations
- Polygonal modeling and mesh data structures
- Texture mapping



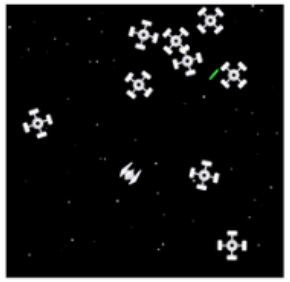


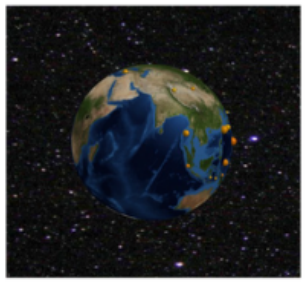




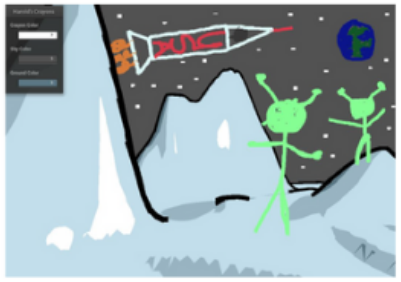
Recap: Second Half of Semester

- Hierarchical transformations and animated characters
- Lighting models
- Shader programs
- Projection and the virtual camera
- Ray casting for 3D user interfaces









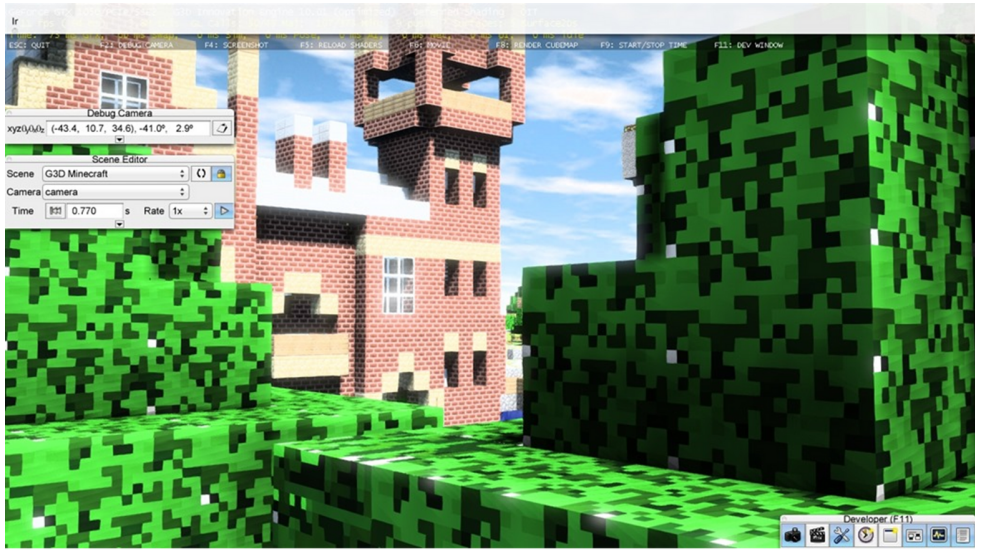
Computer Graphics Toolkits

- In this class, you gained experience using a web-based computer graphics toolkit that was created specifically for this course.
- You will find a very similar structure in almost all graphics libraries and game engines.
- If you know the core concepts (math library, meshes, shaders, textures) you can pick up almost any graphics toolkit and find the corresponding functions.
- Also, since the shaders use GLSL, you can write custom shaders for pretty much any toolkit that uses OpenGL.

What else is out there? (Graphics APIs)

- G3D
- Irrlicht Engine
- OGRE
- Hoarde3D
- OpenSceneGraph (OSG)
- OpenSG Visualization Toolkit (VTK)
- Three.js

  Babylon.js



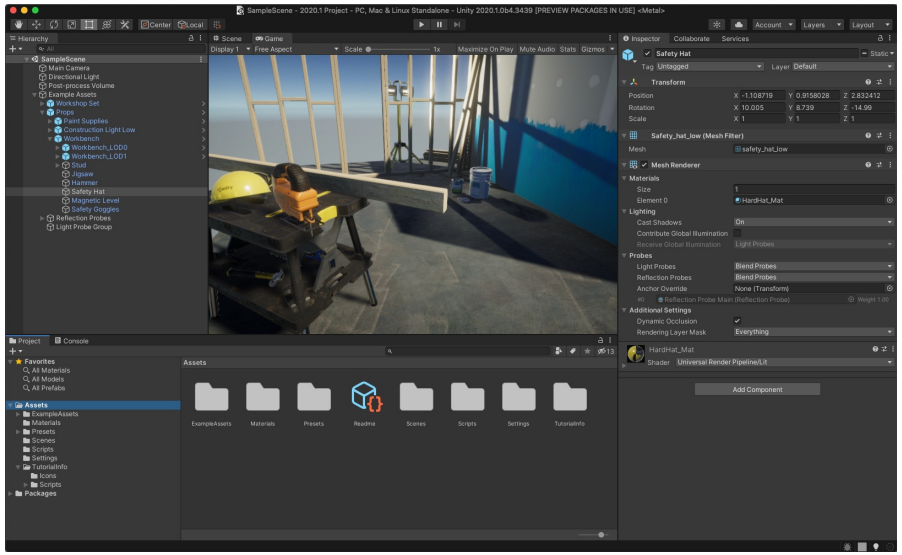
https://casual-effects.com/g3d

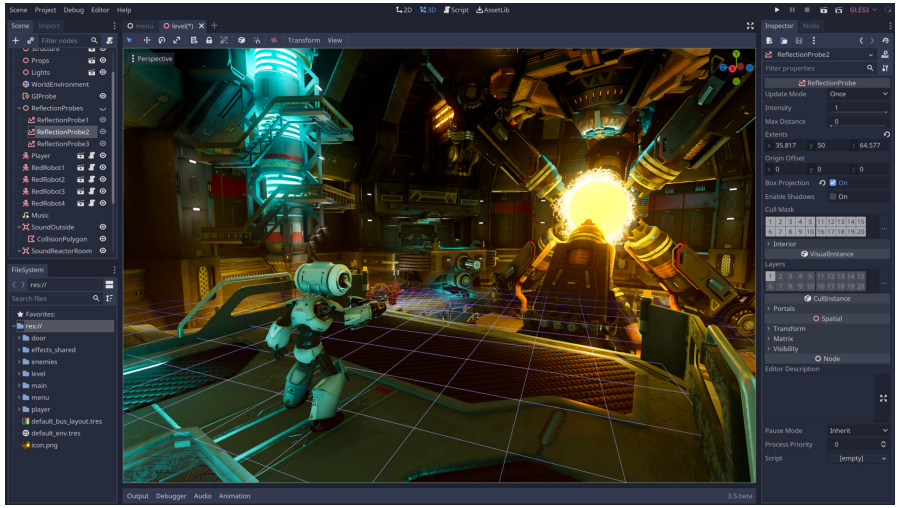
What else is out there? (IDE + Scripting)

This style of game engine is very popular. There are tons of tutorial videos and examples to help developers get started.

- Unity
- Unreal
- CryEngine

  Godot



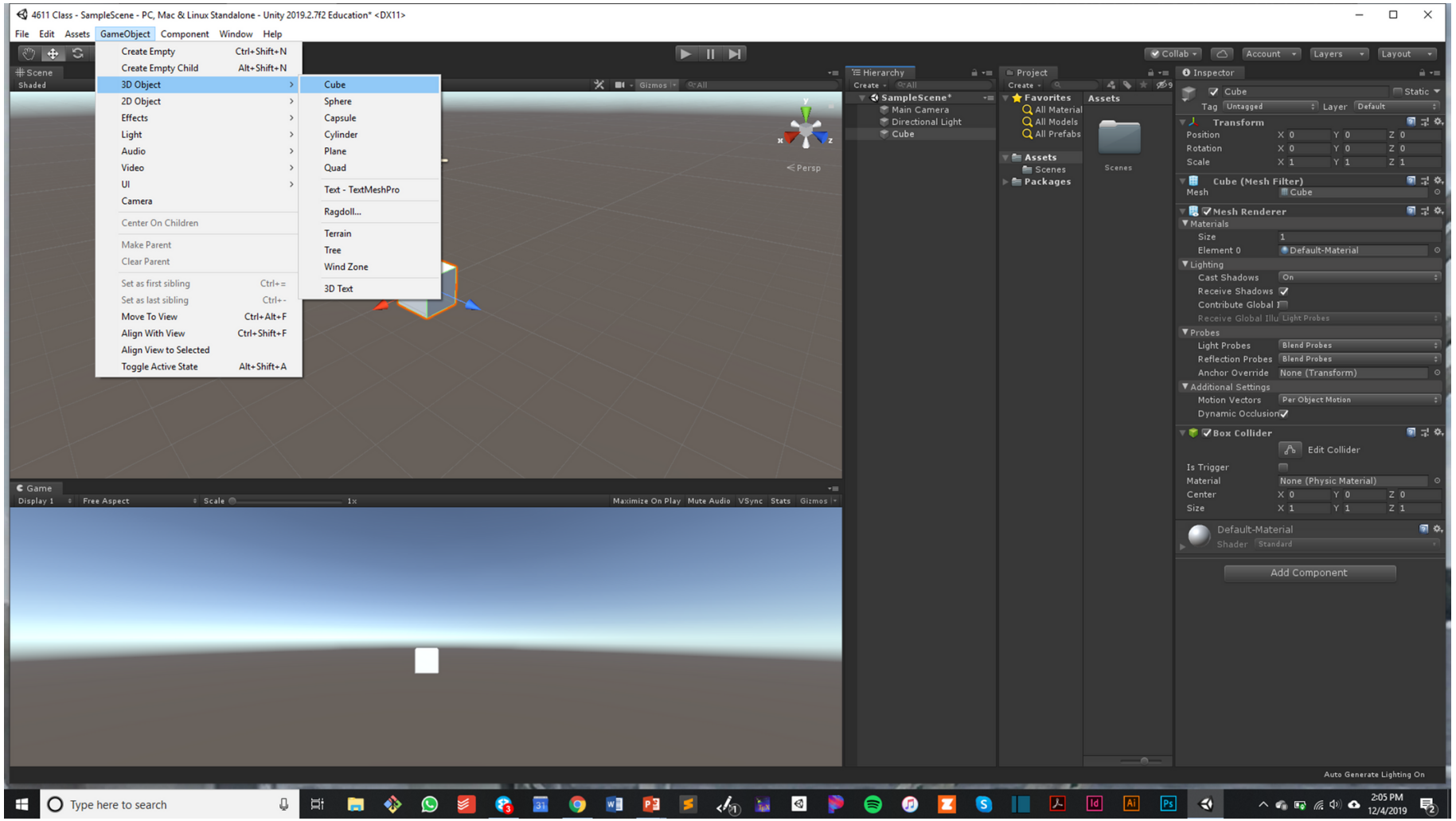


Let's take a little tour of Unity.

FYI - you can download it free for personal use.

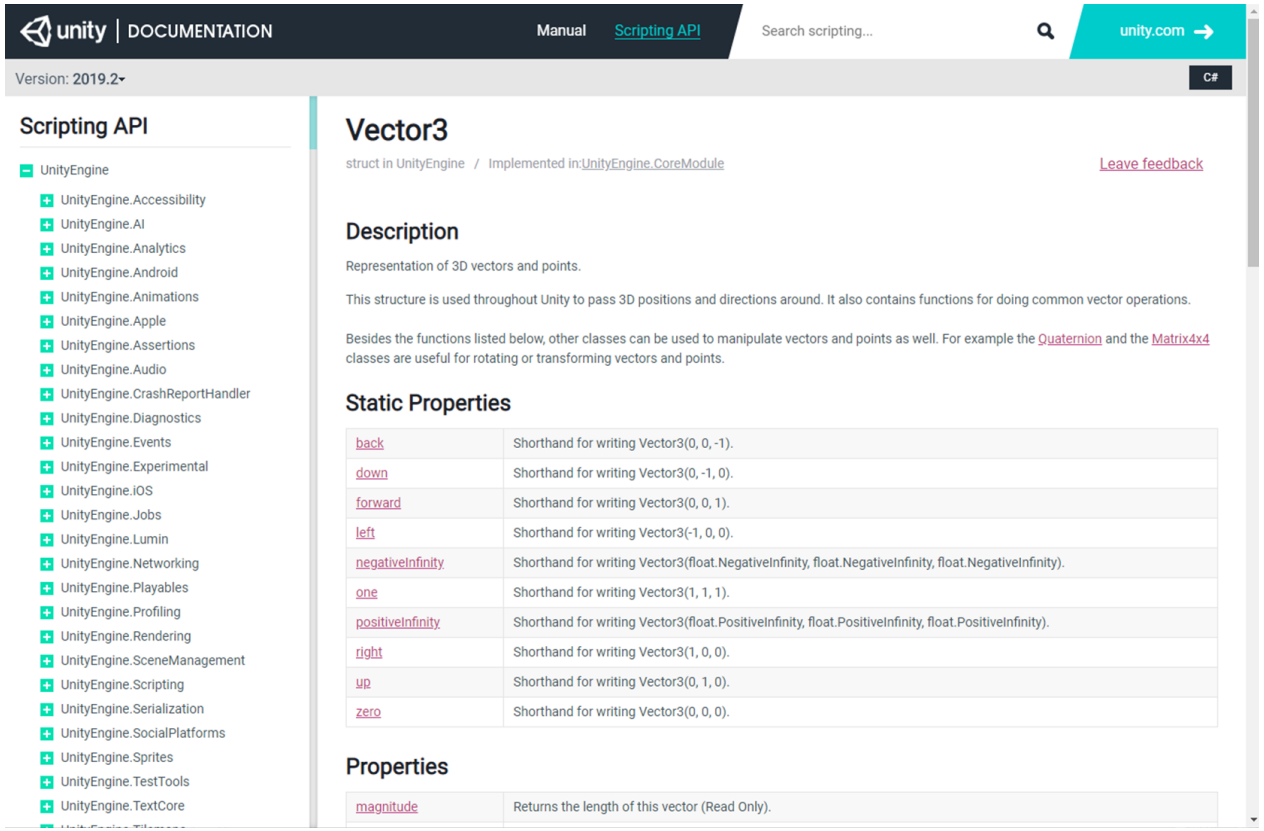
Transitioning to a new toolkit or game engine

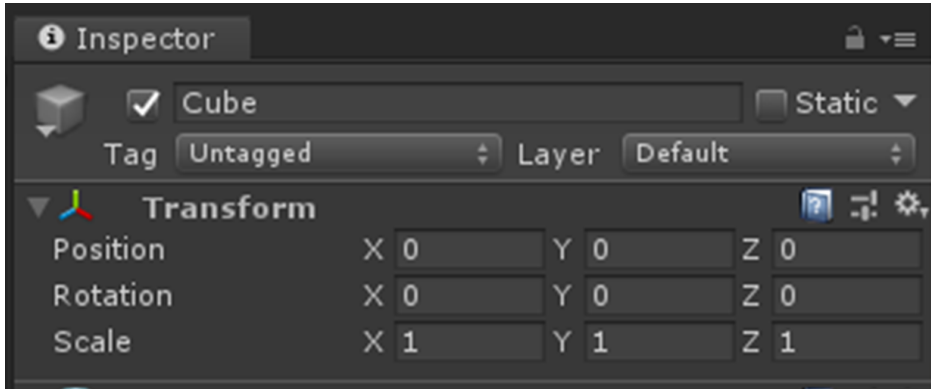
1. Find the 3D graphics primitives (cube, sphere, cylinder, etc.)



Transitioning to a new toolkit or game engine

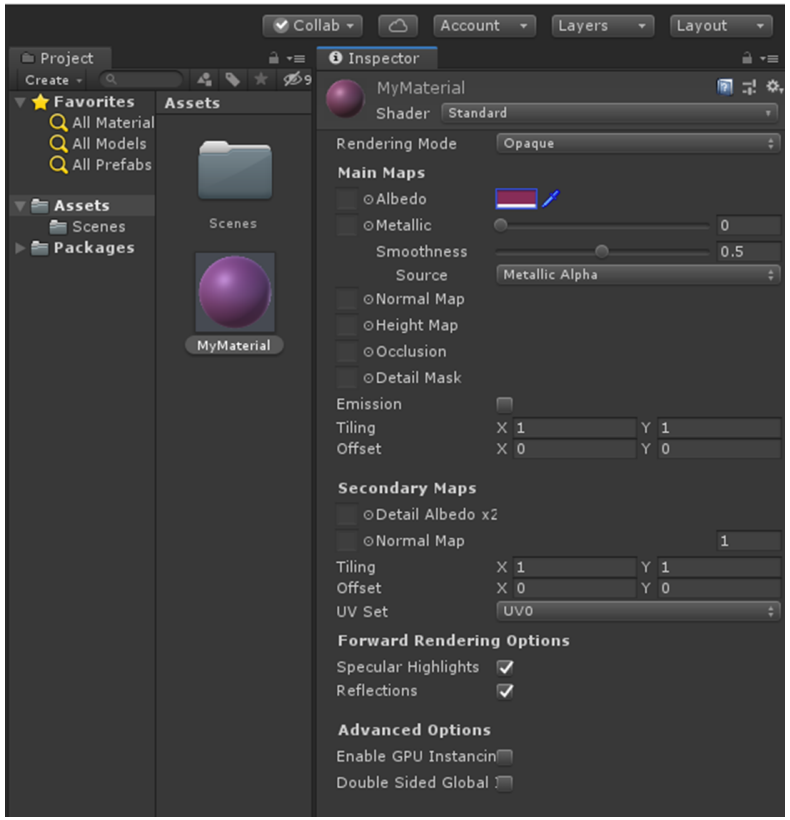
2. Find the Math library (Vector3, Matrix4, Transform, Ray, etc.)

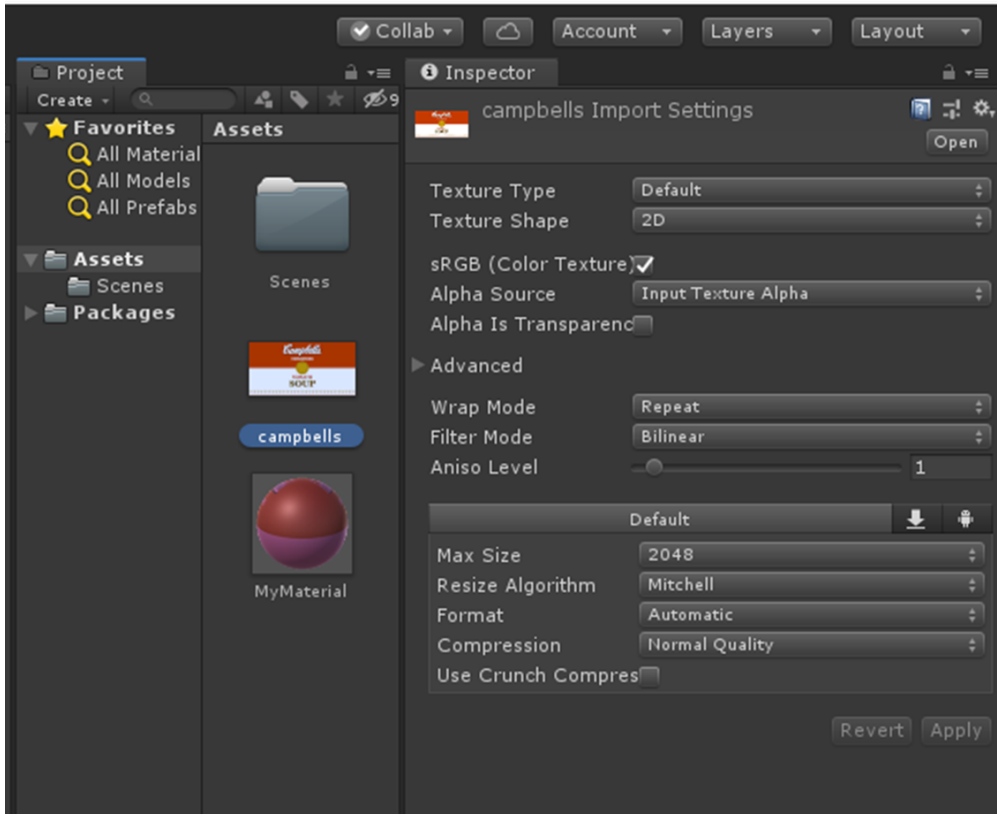




Transitioning to a new toolkit or game engine

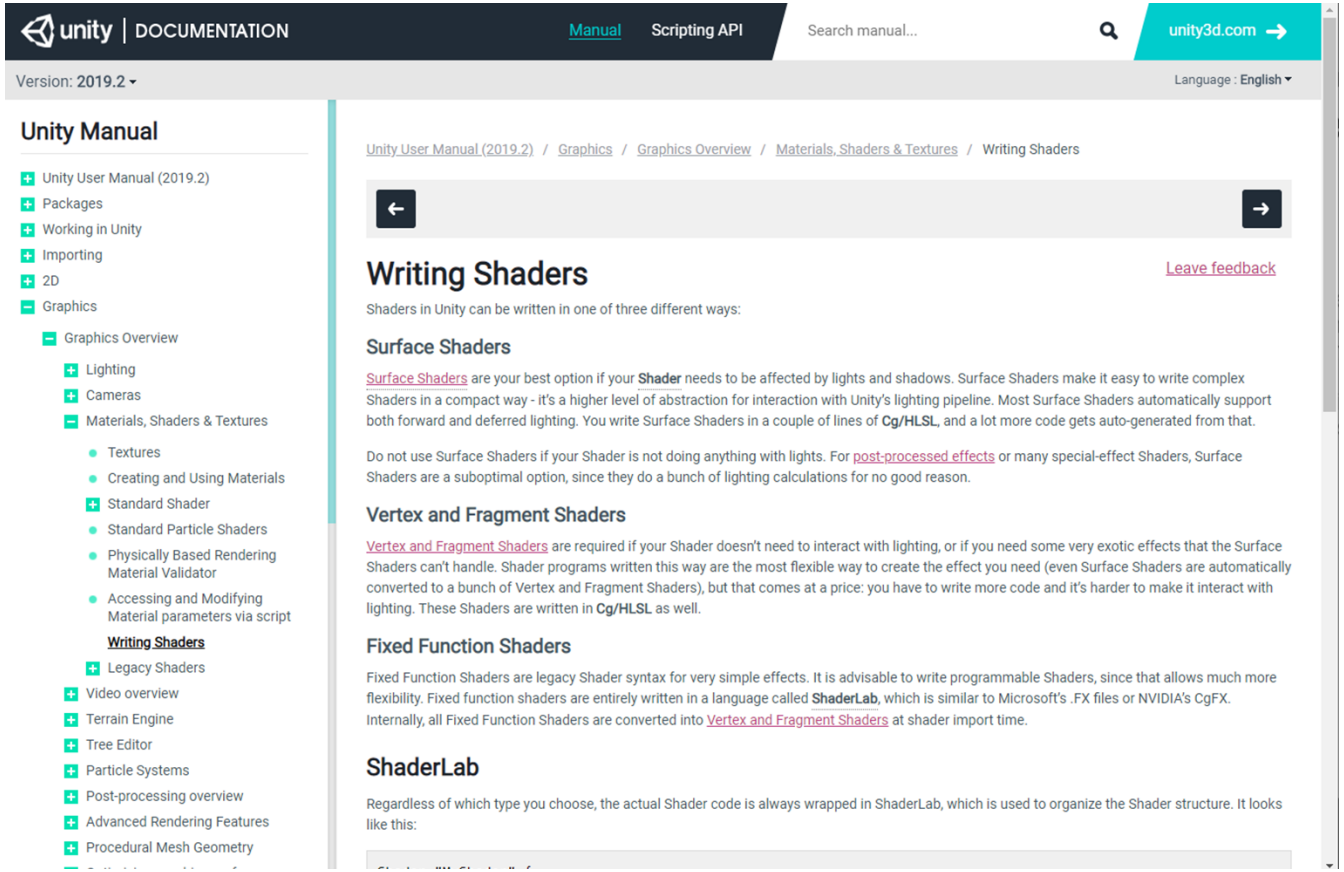
3. Find the material properties (colors, images, textures, etc.).

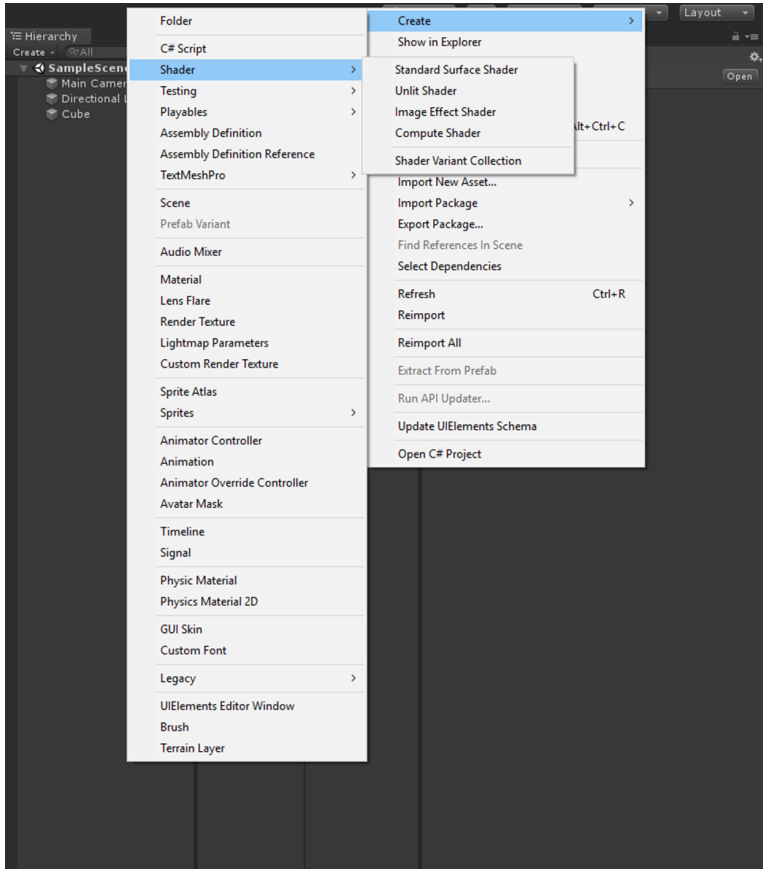




Transitioning to a new toolkit or game engine

4. Find the shaders.





Notes about writing custom shaders

- Game engines will come with a variety built-in shaders, and you could potentially write an entire game without writing a line of shader code.

- However, many games will implement custom shaders to implement visual effects or achieve a specific aesthetic.

- Writing your shaders from scratch is often easier because modifying the built-in shaders, because the professional toolkits will often have very complex shader code that include normal mapping, reflections, transparency, and other effects integrated into one unified shader.

  Shaders will generally be written in GLSL (OpenGL), HLSL (DirectX), or sometimes a custom format that is a high-level wrapper around one of the two.

GLSL vs. HLSL

HLSL (Unity, Unreal)

```
//Vertex Shader
void MainVertexShader(
    float4 InPosition : ATTRIBUTE0,
    float2 InUV : ATTRIBUTE1,
    out float2 OutUV : TEXCOORD0,
    out float4 OutPosition : SV_POSITION
    )
{
    OutPosition = InPosition;
    OutUV = InUV;
}

//Fragment Shader
void MainPixelShader(
    in float2 UV : TEXCOORD0,
    out float4 OutColor : SV_Target0
    )
{
    OutColor = float4(UV, 1.0, 1.0);
}
```

GLSL (Three.js, Godot, G3D, countless others)

```
//Vertex Shader
in vec4 InPosition;
in vec2 InUV;

out vec2 UV;

void main()
{
    gl_Position = InPosition;
    UV = InUV;
}

//Fragment Shader
in vec2 UV;

void main()
{
    gl_FragColor = vec4(UV, 1.0, 1.0);
}
```

Transitioning to a new toolkit or game engine

5. Find the higher-level things we have learned in the assignments:

- **Initialization, update, and draw methods:** You will always find methods where you can initialize the graphics, create objects in the scene, and implement code that needs to execute once per frame as needed for animation, etc.
- **Physics simulation:** You can always do it by hand using points, vectors, etc. Some game engines will also have a built-in physics engine to handle collision detection, gravity, bouncing, etc.
- **Hierarchical transformations:** Every toolkit will have some ability to organize models in the scene in a hierarchy. Unity has a graph of GameObjects, where each has a parent and a list of children.
- **Character animation:** – Many toolkits will support some form of character animation using hierarchical transformations, like we've done. You will need to consider the same concepts we have in class (blending between mocap motions, etc.).
- **Camera and user interface:** There is often a mode for drawing 2D text and lines on top of the screen, like the 2D strokes in the final assignment. Sometimes this includes pre-defined routines for drawing buttons, sliders, etc. You will always find a Camera class or something similar that allows you to create 3D pick rays from the mouse position to create 3D user interfaces.

Some additional game engine features

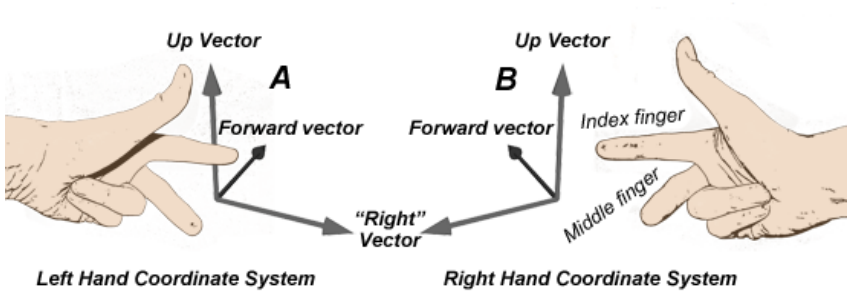
Fancy real-time rendering effects: Most are just extensions of what you have learned about using textures inside shaders and adjusting vertex positions and lighting calculations. Examples include normal mapping, bump mapping, reflections, "moving textures" like water, transparency effects, non-photorealistic effects, and more.

AI: Crowd simulation, flocking behaviors, natural language processing, terrain generation, etc.

VR support: The standards for VR/AR/XR are still actively being developed in many engines. Honestly Unity's API for this is still not that great and often changes between major versions, but it is really cool that if you are not doing anything too fancy in terms of the UI, it is actually quite easy to deploy to a PC-based VR system or mobile VR headset.

Let's discuss two final graphics math concepts that I have seen confuse students when working with game engines such as Unity.

Watch out for Left-handed vs. Right-handed



- If I had to guess, I would estimate that 90% of graphics toolkits use a right-handed coordinate system.
- Unity happens to be in the 10% that picked left handed. *Ugh.*
- Thankfully, all the math really works out the same either way. Just switch to using the "left hand rule" rather than the "right hand rule" for cross products.
- Also be careful when importing models from 3D modeling software or connecting to other systems outside of Unity, like VR tracking systems or your phone's inertial tracker. Almost everybody else uses right handed coordinates.
- You can convert points and vectors from right-handed to left-handed by negating the one of the coordinates. (When Unity loads 3D models stored in right-handed format it usually automatically negates the X coordinate.)

  Converting transformation matrices that include rotations is a bit trickier but can also be done.

Watch out for Rotations

Different game engines will also adopt different conventions for how to represent rotations.

3D Rotations can be represented in several different ways:

- Euler angles (easiest to talk about, but several problems)
- Rotation matrices
- Quaternions (best for interpolation)

Watch out for Rotations

Unity uses all three representations for rotations:

- **Euler angles** are first shown in the editor.
- Each frame, they are converted to **quaternions** to store in a GameObject.
- Programmers can also convert them to **rotation matrices,** apply a series of transformations in a script, and then convert it back to a quaternion.

Some additional opportunities for those interested in computer graphics and games.

(unless you are imminently about to graduate,
in which case congratulations!)

Courses that can build on what you have already learned

- **CSCI 5607: Computer Graphics 1**

  Typically offered once per year in the Spring (taught by Victoria Interrante).

  This course is more theoretical/mathematical than this class.

- **CSCI 5611: Animation and Planning in Games**

  Typically offered once per year in the Fall (taught by Stephen Guy).

- **CSCI 5609: Visualization**

  Typically offered once per year in the Spring (taught by Dan Keefe).

- **CSCI 5619: Virtual Reality and 3D Interaction**

  Typically offered once per year in the Fall (usually taught by me).

- **CSCI 8980: Special Topics Courses**

  These are special one-time courses. Advanced undergraduates can take these classes with instructor permission!

Getting involved in a research lab

- **Research is an opportunity to more deeply explore topic areas that interest you.**

  You don't need to be a graduate student to take advantage of this!

- **Building a mentorship connection with a professor.**

  It's also a fantastic way to get very strong **recommendation letters**!

- **Explore whether graduate school is right for you.**

- **You never know what you might discover!**

  This was my origin story... being a computer science professor was not on my radar until I had a summer research opportunity.

How do I get started?

- **It's never to early to talk to a professor in a research area that interests you!**

  Undergraduates are often active participants in my lab, and I am currently advising several undergraduate research projects.

- **This can often lead to funding from the Undergraduate Research Opportunities Program (UROP).**

  You can get a scholarship stipend to support a research project.

  https://ugresearch.umn.edu/opportunities/urop

Final Thoughts

I had a lot of fun teaching this course, and I hope you had fun taking it too!

If you are interested in chatting about opportunities to get more experience with computer graphics, virtual reality, and/or research, please feel free to reach out!

Final Thoughts

I had a lot of fun teaching this course, and I hope you had fun taking it too!

If you are interested in chatting about opportunities to get more experience with computer graphics, virtual reality, and/or research, please feel free to reach out!

For those who are wondering about my video game plans over winter break!