
TECH 421 - Future of Digital Media
TECH 3706 - AR/VR in Architectural Environments

Due: 10/24

(This Tuesday)

All work on logic/art of your programs is done. We will spend the day troubleshooting the build process.

10/26
In Class Showcase + Discussion

All Materials due 10/31:

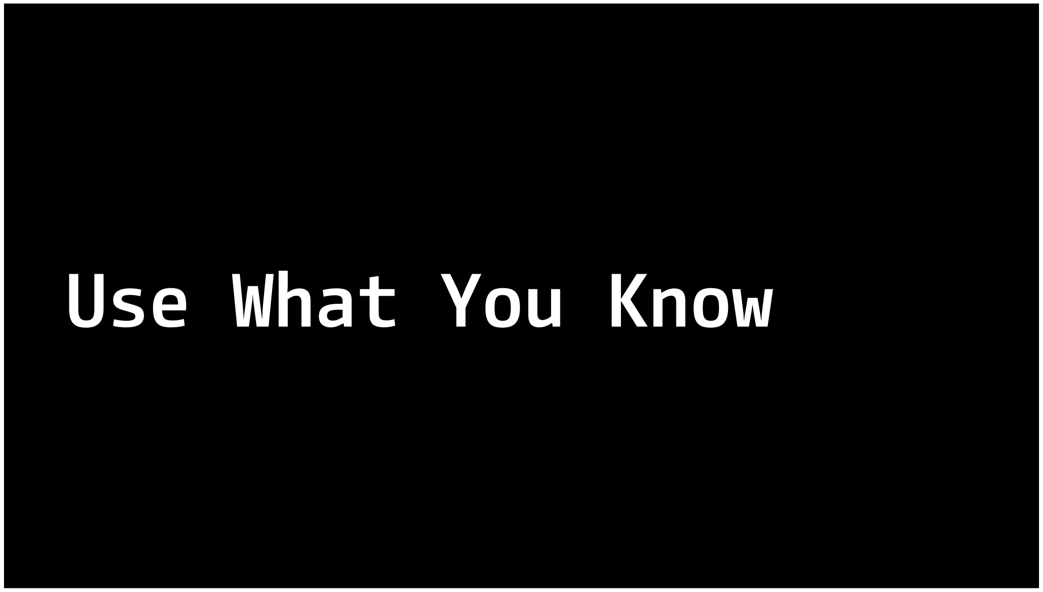
Documentation

- Video of the program
- Planning documentation

Unity Project File

Finished (Compiled) Program

Home Stretch!



- Break problem into smaller pieces
- ‘Hack-y’ solutions are OK!

Don’t go down rabbit holes…

A beautiful 3D Model is great, but not if your project doesn’t do anything.

Define your problem

What is success for your project?

Don’t forget the boring stuff

- Organizing your program helps you make changes
- Keep track of the **state** of your program
  - For example:

```
bool isReady = false;
```

- Use if/else statements to check the state and change behavior accordingly:

```
if (isReady == true) {
        // Do something here
} else {
        // Do something different here
}
```

Don’t worry! Here’s a puppy!

Some more Hololens Topics:

- RayCasting
- Spatial Mapping
- Global Event Listeners

RayCasting

RayCasting is when we shoot an invisible line into our scene to see if we hit something in that direction.

To understand RayCasting, you must understand **Vectors**.

Vectors

In programming terms, you can think of Vectors as a way to store 2, 3, or 4 values in one easy-to-use package:

```
Vector2 someNumbers = new Vector2(1.0, 2.2);
Vector3 someOtherNumbers = new Vector3(5.3, 2.6, 12.0);
Vector4 evenMoreNumbers = new Vector4(7.4, 2.1, 12.0, 9.8);
```

Vectors

We can use vectors to:

- Store multiple numbers in one variable
- Describe the position of something in our world
  - For example: (2.1, 8.9, 7.4) represents the point in space 2.1 units along the X-axis, 8.9 units along the Y-axis, and 7.4 units along the Z-axis.

Vectors

We can use vectors to:

- Describe a direction
  - For example: (0.0, 1.0, 0.0) represents a point 1 unit directly above (along Y) the origin.
  - If we a drew an arrow from the origin to this point, it would point straight up.
  - It doesn't matter how long the Vector is:
    - (0.0, 1.0, 0.0) and (0.0, 5.2, 0.0) are different points, but they both describe the same *direction* (straight up).

Vectors

Unity has some built-in direction shorthands:

```
Vector3 example = Vector3.up;
```

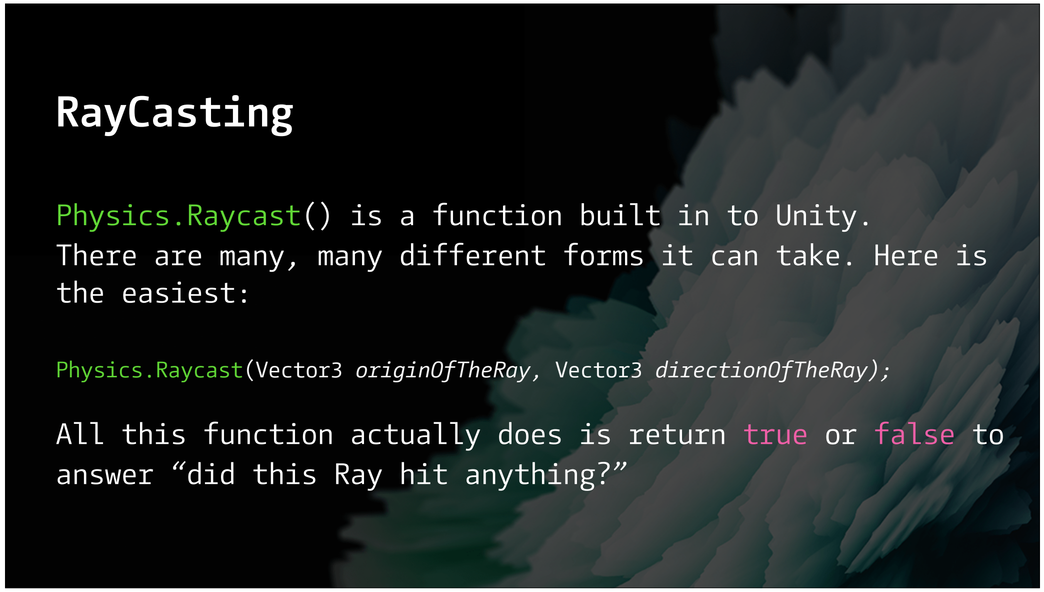
is the same as:

```
Vector3 example = new Vector3(0.0, 1.0, 0.0);
```

Vectors

Other shorthands:

```
Vector3.up (pointing along Y-axis)
Vector3.forward (pointing along Z-axis)
Vector3.right (pointing along X-axis)
Vector3.one (Equal to (1.0, 1.0, 1.0)
```



See the example script on the github for more details:
https://github.com/ivaylopg/Tech421Tech3706/blob/master/Session17/ScriptExamples/RayCaster.cs

RayCasting

To store information about *what* was hit, and more importantly *where* the hit is in space, we have to do two things:

1. Declare a variable of the type RaycastHit to store the information about the hit point.
2. Use a slightly different version of Physics.Raycast() to pass the hit info out of it:

```
RaycastHit hitInfoVariable
Physics.Raycast(Vector3 originOfTheRay, Vector3 directionOfTheRay, out hitInfoVariable)
```

See the example script on the github for more details:
https://github.com/ivaylopg/Tech421Tech3706/blob/master/Session17/ScriptExamples/RayCaster.cs

RayCasting

So if wanted to Raycast from a GameObject (for example a Vive tracker or the user's headset POV):

We want to shoot a ray from:
gameObject.transform.position

in the direction of:
gameObject.transform.forward

(**gameObject.transform.forward** is the **local Z-axis** of the *object*, which may be different from the *world* Z-axis, which is Vector3.forward)

See the example script on the github for more details:
https://github.com/ivaylopg/Tech421Tech3706/blob/master/Session17/ScriptExamples/RayCaster.cs

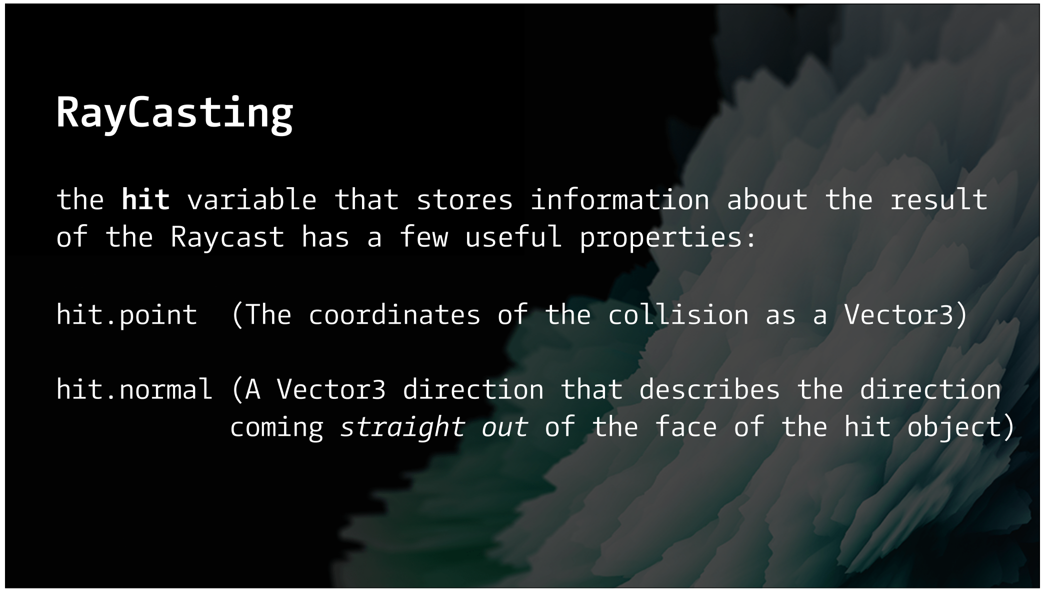
RayCasting

```
void Update() {
    RaycastHit hit;
    if ( Physics.Raycast(gameObject.transform.position, gameObject.transform.forward, out hit) ) {

        Debug.DrawLine(gameObject.transform.position, hit.point, Color.red);
        Debug.DrawRay(hit.point, hit.normal, Color.green);

    } else {
        hitMarker.SetActive(false);
    }
}
```

See the example script on the github for more details:
https://github.com/ivaylopg/Tech421Tech3706/blob/master/Session17/ScriptExamples/RayCaster.cs



See the example script on the github for more details:
https://github.com/ivaylopg/Tech421Tech3706/blob/master/Session17/ScriptExamples/RayCaster.cs

RayCasting

These visual Debug functions help you see what’s going on. They will draw lines in your *Editor*, but never in the actual *Game* view:

```
Debug.DrawLine(Vector3 lineStartCoordinate, Vector3 lineEndCoordinate, Color color);

Debug.DrawRay(Vector3 lineStartCoordinate, Vector3 lineDirection, Color color);
```

See the example script on the github for more details:
https://github.com/ivaylopg/Tech421Tech3706/blob/master/Session17/ScriptExamples/RayCaster.cs

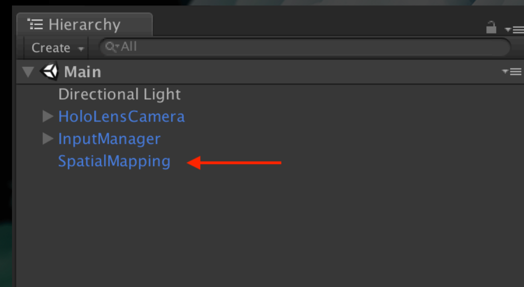
Spatial Mapping

As soon as you start a program on Hololens, it begins to scan its surroundings and make this data available to you through the **Spatial Mapping** API.

Spatial Mapping

HoloToolkit makes it super easy to use this info in our programs!

Drag the **SpatialMapping** prefab into the root of your scene

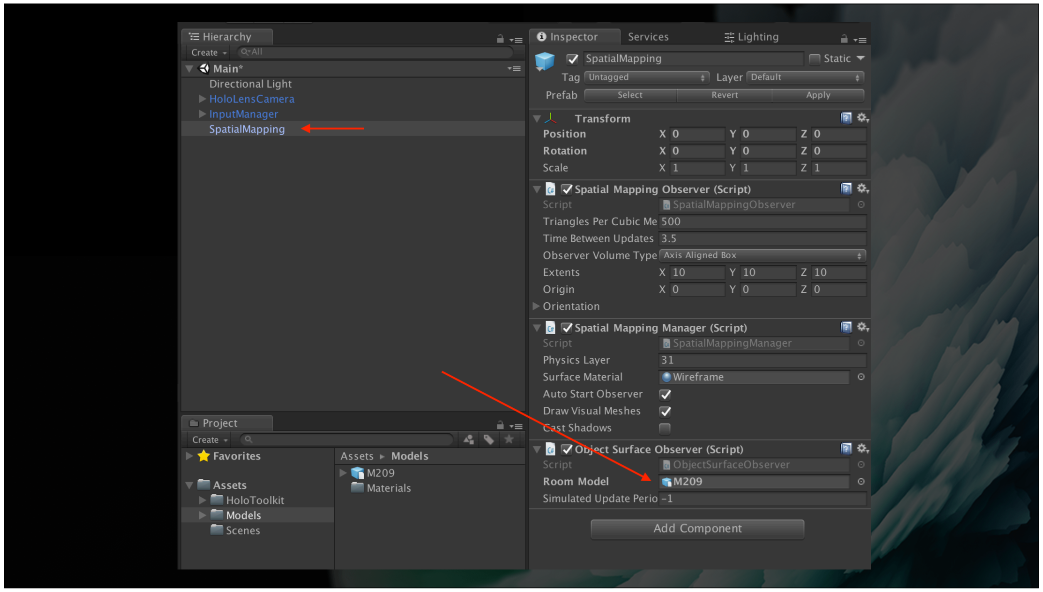


Spatial Mapping

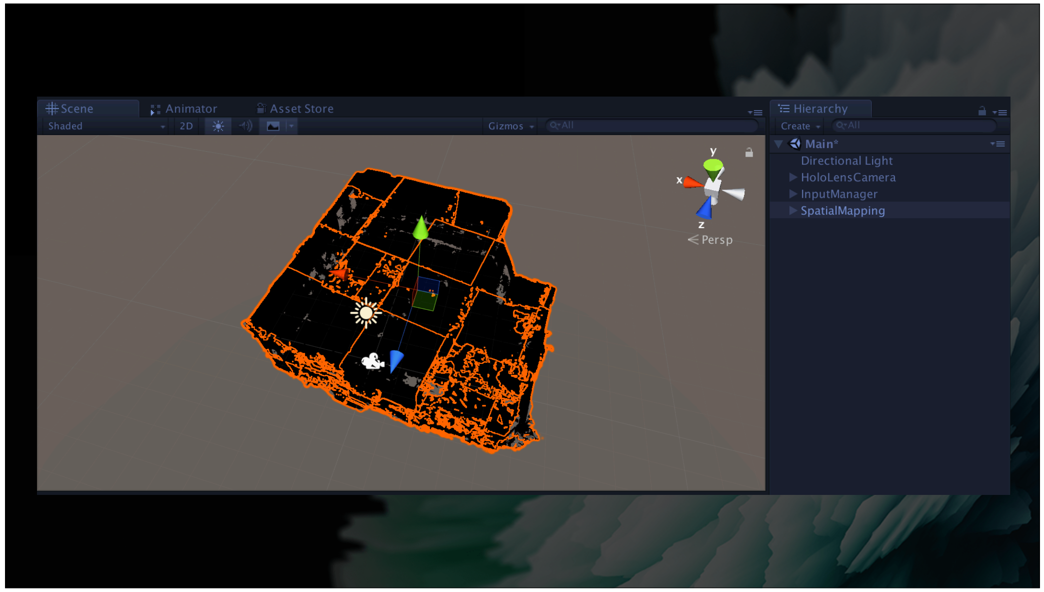
On an actual device, you'd be good to go. Unity will start seeing the spatial information as it comes in.

To use this in our *simulator* though, we have to provide a room model. Any OBJ model will work.

An OBJ of M209 is on the github, pulled directly from one of our Hololenses (so it is *exactly* how it would see our room)



Hierarchy
Main*
Directional Light
HoloLensCamera
InputManager
SpatialMapping
Project
Favorites
Assets
HoloToolkit
Models
Scenes
Assets ▸ Models
M209
Materials
Inspector
Services
Lighting
SpatialMapping
Static
Tag Untagged
Layer Default
Prefab Select Revert Apply
Transform
Position X 0 Y 0 Z 0
Rotation X 0 Y 0 Z 0
Scale X 1 Y 1 Z 1
Spatial Mapping Observer (Script)
Script SpatialMappingObserver
Triangles Per Cubic Me 500
Time Between Updates 3.5
Observer Volume Type Axis Aligned Box
Extents X 10 Y 10 Z 10
Origin X 0 Y 0 Z 0
Orientation
Spatial Mapping Manager (Script)
Script SpatialMappingManager
Physics Layer 31
Surface Material Wireframe
Auto Start Observer
Draw Visual Meshes
Cast Shadows
Object Surface Observer (Script)
Script ObjectSurfaceObserver
Room Model M209
Simulated Update Perio -1
Add Component

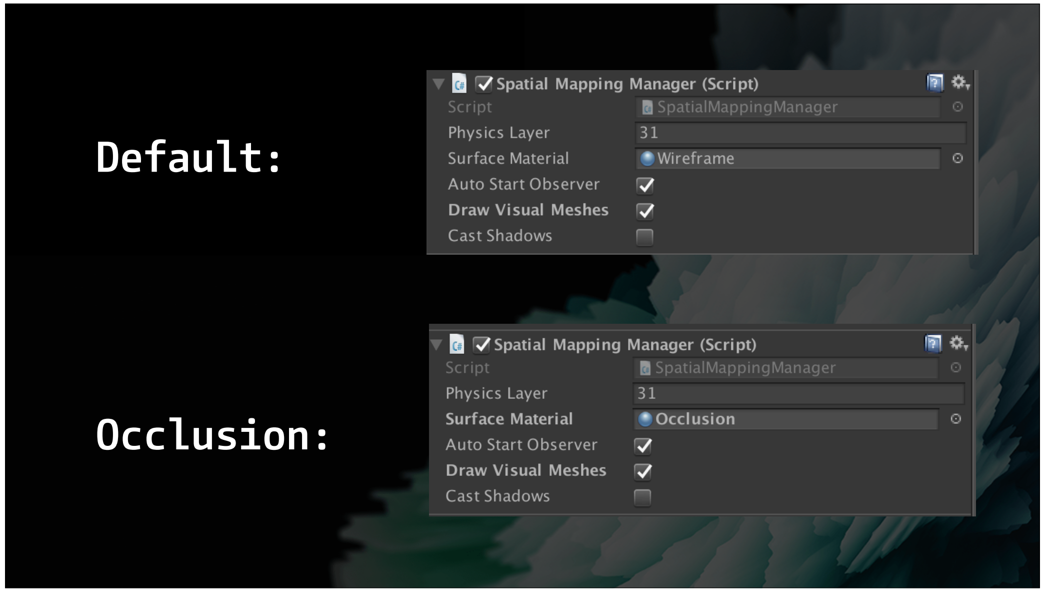


Scene
Animator
Asset Store
Shaded
2D
Gizmos
Hierarchy
Create
Main*
Directional Light
HoloLensCamera
InputManager
SpatialMapping
Persp

Spatial Mapping

By Default, **SpatialMapping** uses a black-and-white wireframe material, but we usually don't want to see this.

HoloToolkit comes with a material called **Occlusion** which is *invisible* <u>but still obscures things behind it</u>. Most of the time you want to tell **Spatial Mapping** to use this material.



To see how to make this change while the program is running, take a look at the example on github.

Global Event Listeners

Last time we talked about how HoloLens uses **events** to "listen" for events instead of "asking" 60x per second if an interaction has happened.

We went over how to add create your own air-tap "listener" and detect if the user tapped while they were looking at something.

Global Event Listeners

But what if we want to “listen” for an interaction *globally*?

That is, we want to react to a tap or a hold when we are *not* looking at a specific object.

Global Event Listeners

The script is the ***exact same*** as the normal event listener with one extra line:

```
void Start() {
  // This line registers this script as the “fallback” event-
  // handler for events of this type not bound to another object
  InputManager.Instance.PushFallbackInputHandler(gameObject);
  // Your other Start stuff goes here...
}
```

You can find the whole script on the github page


TECH 421 - Future of Digital Media
TECH 3706 - AR/VR in Architectural Environments