Hololens Documentation

Objective

The purpose of this document is to provide an in-depth report on the steps we took to create our Hololens app. This document should help students who are planning on taking over this project.

URL Link to the Project Website:

https://sites.google.com/illinois.edu/cs-468-hololens/home

Project Description

I. Background

This project consisted of four key members: Charis Ryu, Dhruv Bansal, Kunal Shah, and Shruti Goli. All four of us have a background in Computer Science and were exploring how we could use the Microsoft Hololens to support key advertising techniques.

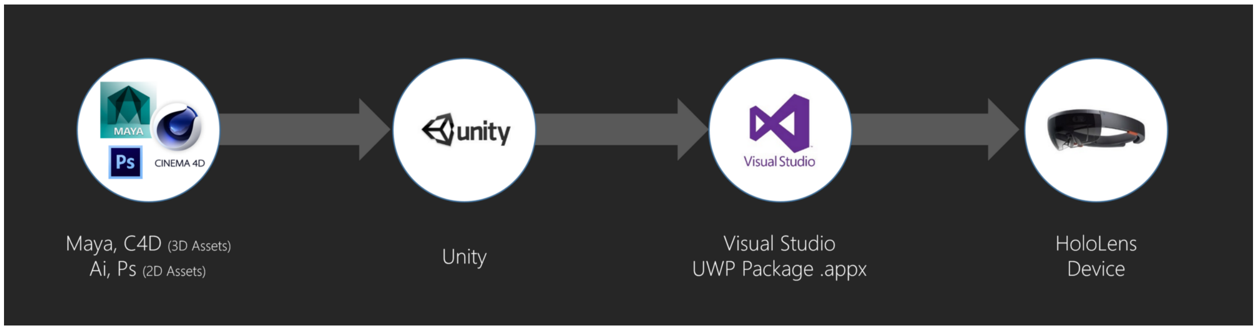
II. Project idea & Goals

The general goal of this project was to explore the Microsoft Hololens development environment by creating an application of our own to run on the device. The idea of the project was influenced by everyone's unequivocal love of food. Because of this, we decided to show nutrition facts for various toppings on a pizza as well as facts about the pizza itself. In addition to making this app, we also wanted to thoroughly document our development experience on the Microsoft Hololens and the steps that we took.

Hololens App Setup and Development

I. Setting up the Environment

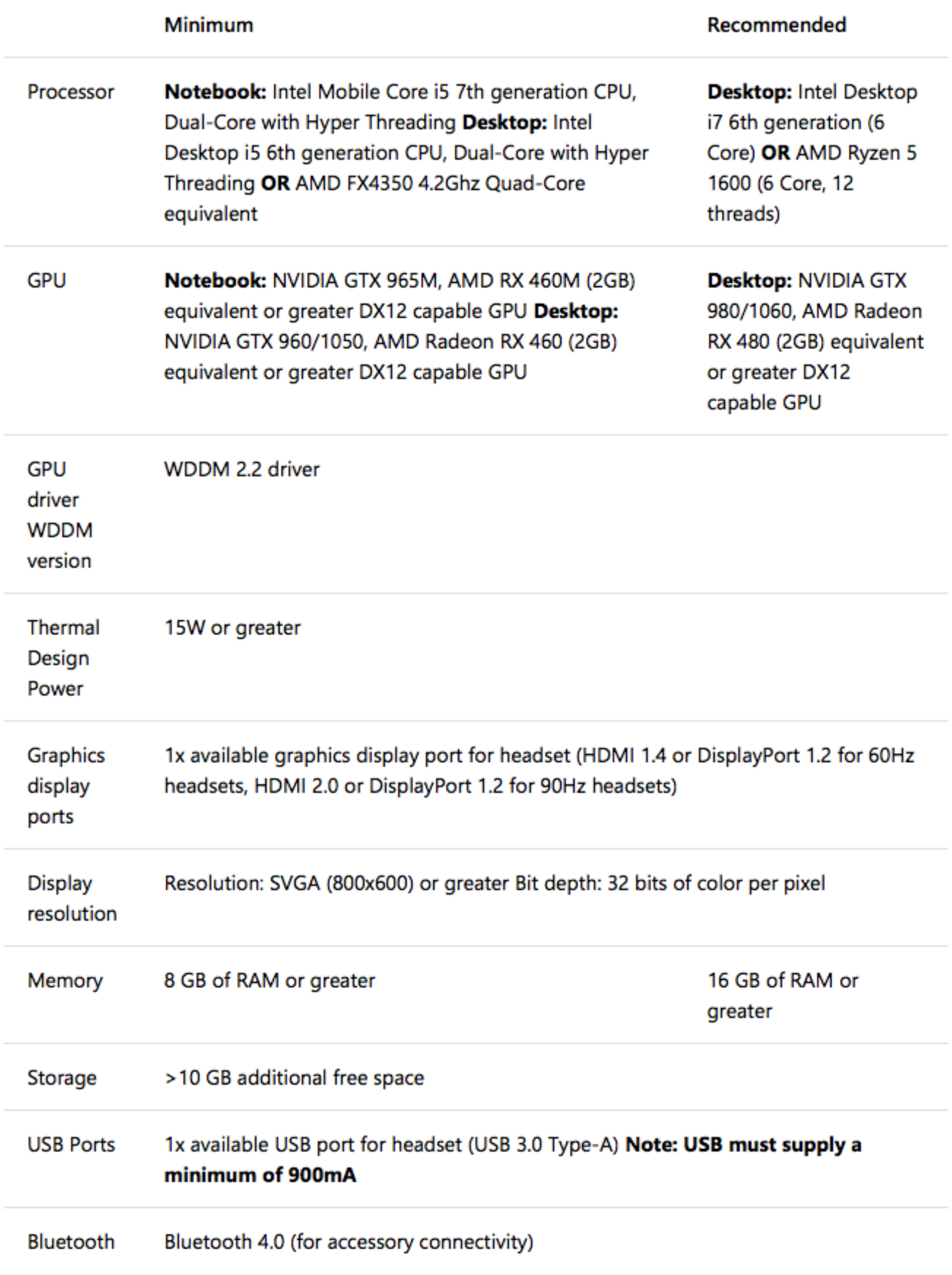
As all of the team members were new to AR/VR devices and softwares, we first did a lot of research on the steps in 3D development, especially Hololens app development as well as the tools and engines to use for our final project. First, we found various 3D development engines that were widely used by the 3D developers. We then narrowed down the options to the tools that are appropriate for Hololens app development, and decided to use Unity and Visual Studio for our development and deployment process. Below is the general deployment process of an Hololens app. We used Photoshop to create our 3D Assets, which will be discussed more deeply later in



this section.

retrieved from: https://medium.com/microsoft-design/holosketch-spatial-layout-and-ux-sketching-app-for-hololens-cb5e17237108

Preparing for Development



	Minimum	Recommended
Processor	**Notebook:** Intel Mobile Core i5 7th generation CPU, Dual-Core with Hyper Threading **Desktop:** Intel Desktop i5 6th generation CPU, Dual-Core with Hyper Threading **OR** AMD FX4350 4.2Ghz Quad-Core equivalent	**Desktop:** Intel Desktop i7 6th generation (6 Core) **OR** AMD Ryzen 5 1600 (6 Core, 12 threads)
GPU	**Notebook:** NVIDIA GTX 965M, AMD RX 460M (2GB) equivalent or greater DX12 capable GPU **Desktop:** NVIDIA GTX 960/1050, AMD Radeon RX 460 (2GB) equivalent or greater DX12 capable GPU	**Desktop:** NVIDIA GTX 980/1060, AMD Radeon RX 480 (2GB) equivalent or greater DX12 capable GPU
GPU driver WDDM version	WDDM 2.2 driver	
Thermal Design Power	15W or greater	
Graphics display ports	1x available graphics display port for headset (HDMI 1.4 or DisplayPort 1.2 for 60Hz headsets, HDMI 2.0 or DisplayPort 1.2 for 90Hz headsets)	
Display resolution	Resolution: SVGA (800x600) or greater Bit depth: 32 bits of color per pixel	
Memory	8 GB of RAM or greater	16 GB of RAM or greater
Storage	>10 GB additional free space	
USB Ports	1x available USB port for headset (USB 3.0 Type-A) **Note: USB must supply a minimum of 900mA**	
Bluetooth	Bluetooth 4.0 (for accessory connectivity)	

1. Ensure that you have the right hardware setup

2. Install required software

- Visual Studio 2017
    - *Please see the additional installation section for additional components that must be installed while installing Visual Studio*
- Unity 2017
    - Make a Unity account
    - Install Unity from https://unity3d.com/get-unity/download
    - In Choose Components screen, select "Universal Windows Platform"
    - In Player Settings, check Virtual Reality Supported
- Windows 10 Fall Creators Update
- Hololens Emulator and Holographic Templates
    - (**Optional)** *Please note this is only needed if you do NOT have a device*

3. Additional installation

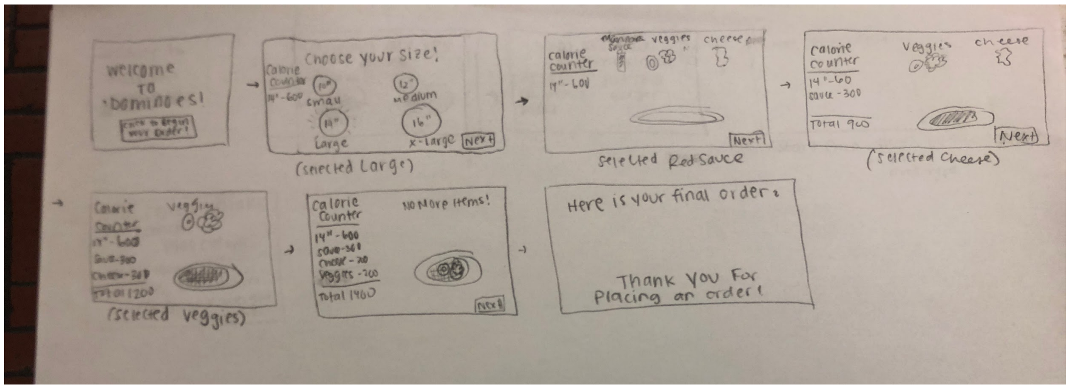
- Enable **Developer mode**
    - Go to settings → Update & security → For developers
- **Visual Studio Workloads**
    - Check the respective box for the following workloads to install
        - Universal Windows Platform development
        - Desktop development with C++
        - Game development with Unity
- Update **graphics drivers**
    - Start→ Windows Settings→ Update and Security→ Check for Updates
    - Check for latest optional driver updates using Device manager
    - Device manager can be accessed by right clicking Start→ Device Manager→ expand Display Adapters → right click your machine's graphic card and choose Update Driver → Search automatically for updated driver software
    - Clone the MixedRealityCompanionKit

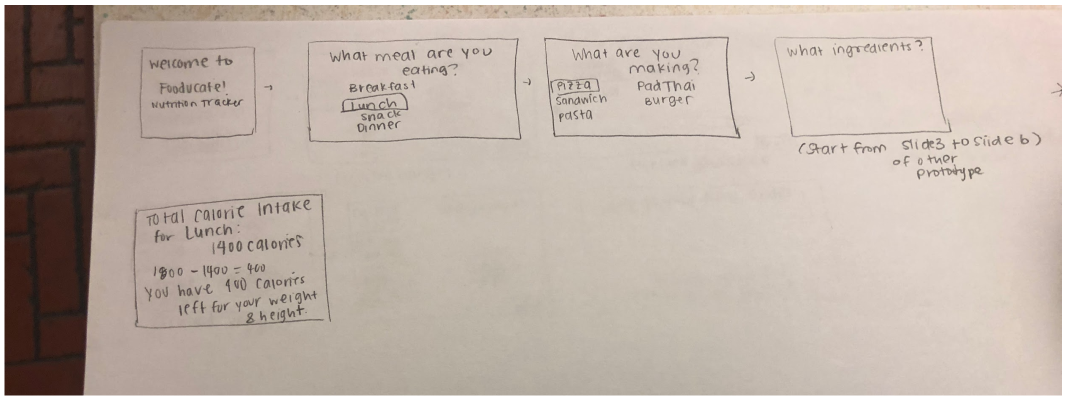
4. To see how to deploy to the Hololens, go to the **Deploying to Hololens** section below

II. Development Process

Prototyping

After completely forming an idea for a product, it is important to prototype these ideas. In terms of prototyping this, we started off by doing a paper prototype. We spent a lot of time discussing the layout we would ideally want for our pizza. Additionally, we watched a few helpful videos on how Microsoft Hololens is being applied. We identified important user scenarios such as being able to pull the items apart, put them together, and creating a product or idea with various features. We came up with a couple basic prototypes and thought about user testing during this process. We wanted to create a pizza, where the different ingredients could easily be pulled apart or put together.





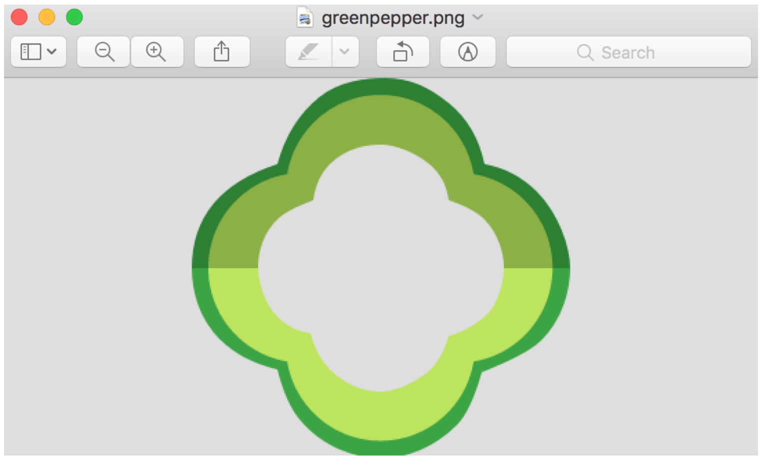
Creating 2D/3D Assets

The next step after creating the prototypes and discussing the interactive features and the functionality of the app was to start developing our app. We followed the general deployment process of “Creating assets -> Developing on Unity & Visual Studio -> Displaying on Hololens”. After creating these assets, we dragged the MixedRealityCompanionKit assets into our Unity project

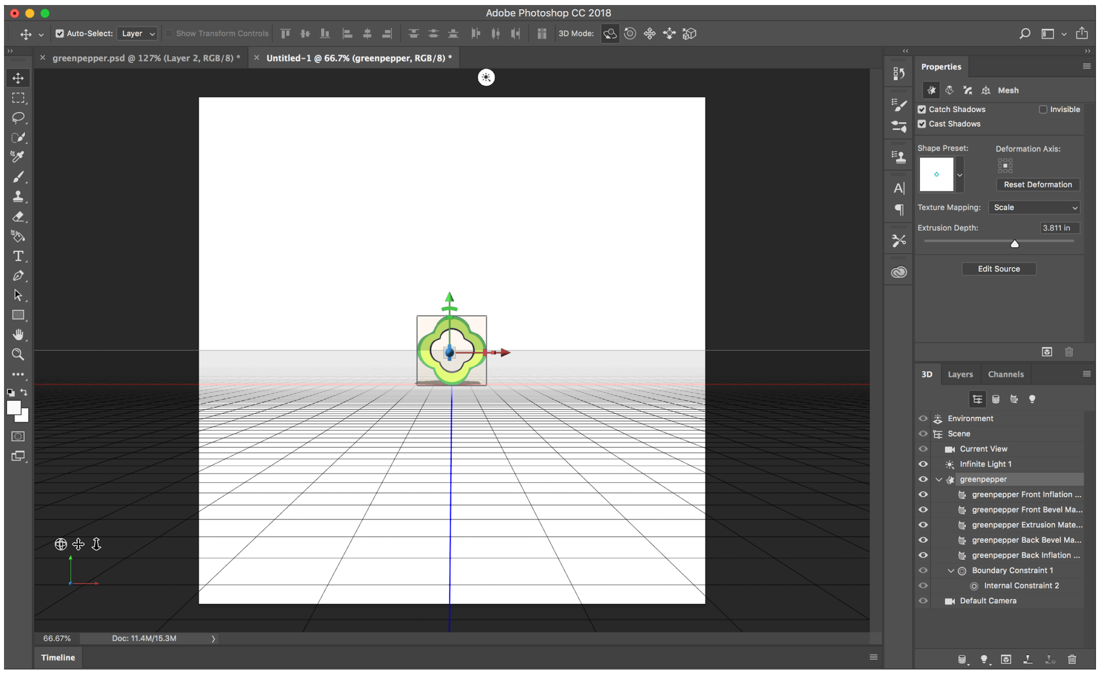
First, we designed 3D assets with Adobe Photoshop CC. Various 2D and 3D assets are available on Unity Asset Store (https://assetstore.unity.com), however we could not find the appropriate assets that we need for our project, the models of pizza ingredients. We decided to create our own 3D models with Photoshop. Professional 3D designers would use other softwares such as Blender, Maya, or Autodesk 3ds Max, however we chose Photoshop as Photoshop can build simple 3D models out of 2D images very quickly. Below are the steps of our simple 3D modeling process.

3D Modeling

- Create 2D images



- Import the image files to Photoshop
- Select **3D -> New 3D Extrusion from Selected Layer**


Adobe Photoshop CC 2018

greenpepper.psd @ 127% (Layer 2, RGB/8) *

Untitled-1 @ 66.7% (greenpepper, RGB/8) *

Properties — Mesh

[✓] Catch Shadows [ ] Invisible

[✓] Cast Shadows

Texture Mapping: Scale

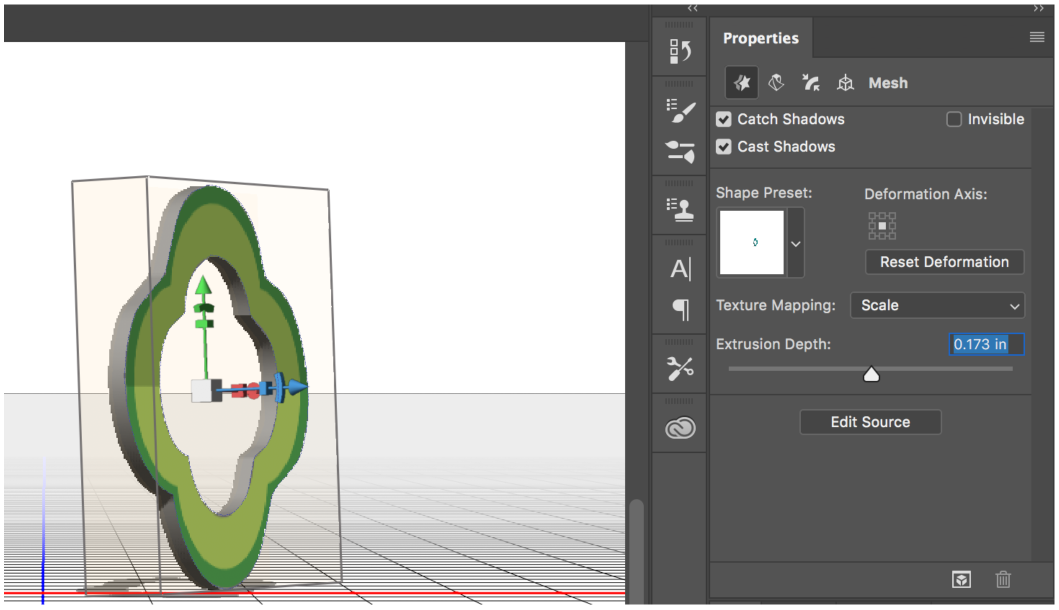
Extrusion Depth: 3.811 in

3D

- Environment
- Scene
  - Current View
  - Infinite Light 1
  - greenpepper
    - greenpepper Front Inflation ...
    - greenpepper Front Bevel Ma...
    - greenpepper Extrusion Mate...
    - greenpepper Back Bevel Ma...
    - greenpepper Back Inflation ...
    - Boundary Constraint 1
      - Internal Constraint 2
  - Default Camera

66.67% Doc: 11.4M/15.3M

- Edit the properties to make the model into the desired 3D shape
  - We only had to change the “Extrusion Depth” and “Shape Preset” for most of our 3d models as they were in very simple shapes.


Properties — Mesh

[✓] Catch Shadows [ ] Invisible

[✓] Cast Shadows

Texture Mapping: Scale

Extrusion Depth: 0.173 in

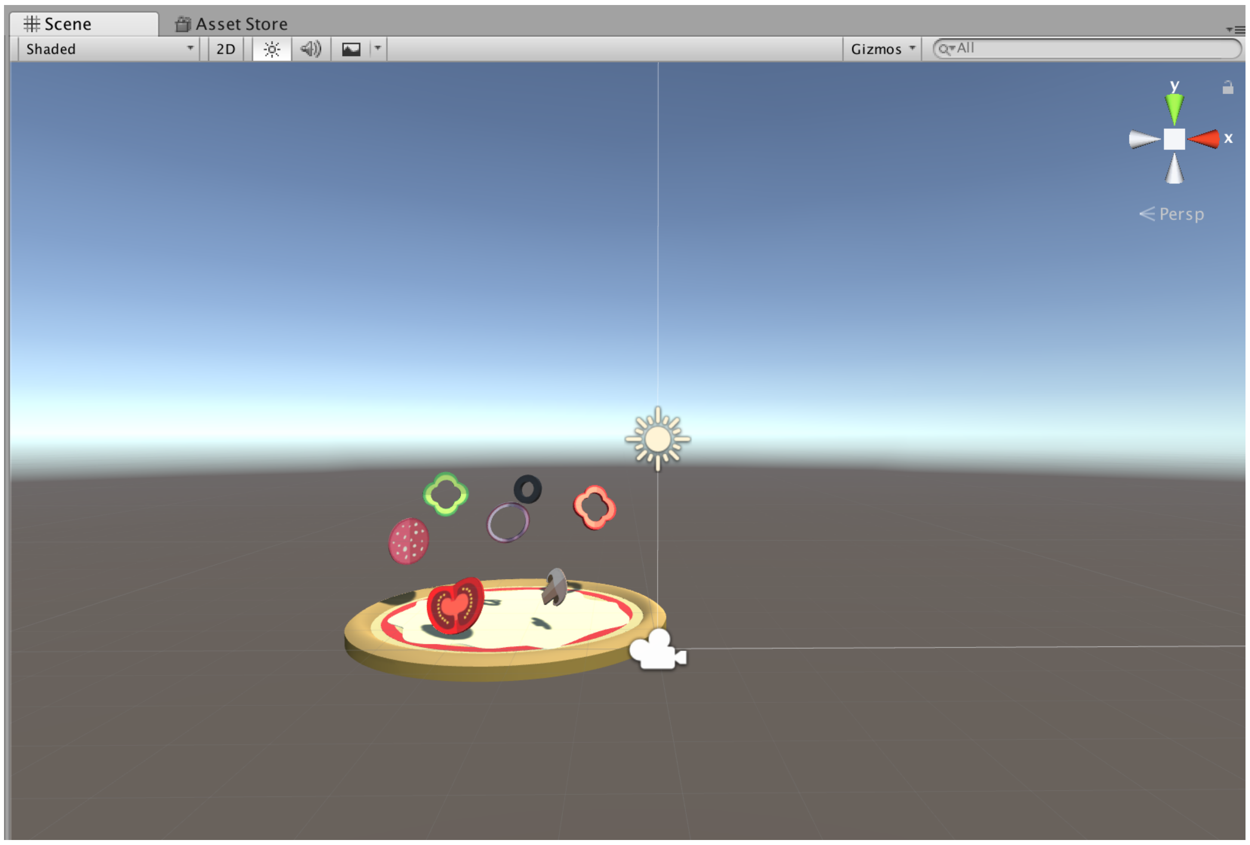
- Export 3D model by selecting **3D -> Export 3D layer** and change the Export Properties (We exported our models as Collada DAE format)
- Import them in Unity for use

2D assets we needed were nutrition facts charts for the toppings that the users can view when they select each topping on Hololens. We created the charts

Creating a Unity Project

The next step after creating the 3D models was setting up a scene in our Unity Project. As we were not developing a multi-scene software like VR game, we only needed one scene, which will display every asset we have.

- Open a new project and create a new scene
- Change the Camera and Light Settings
- Import 3D assets

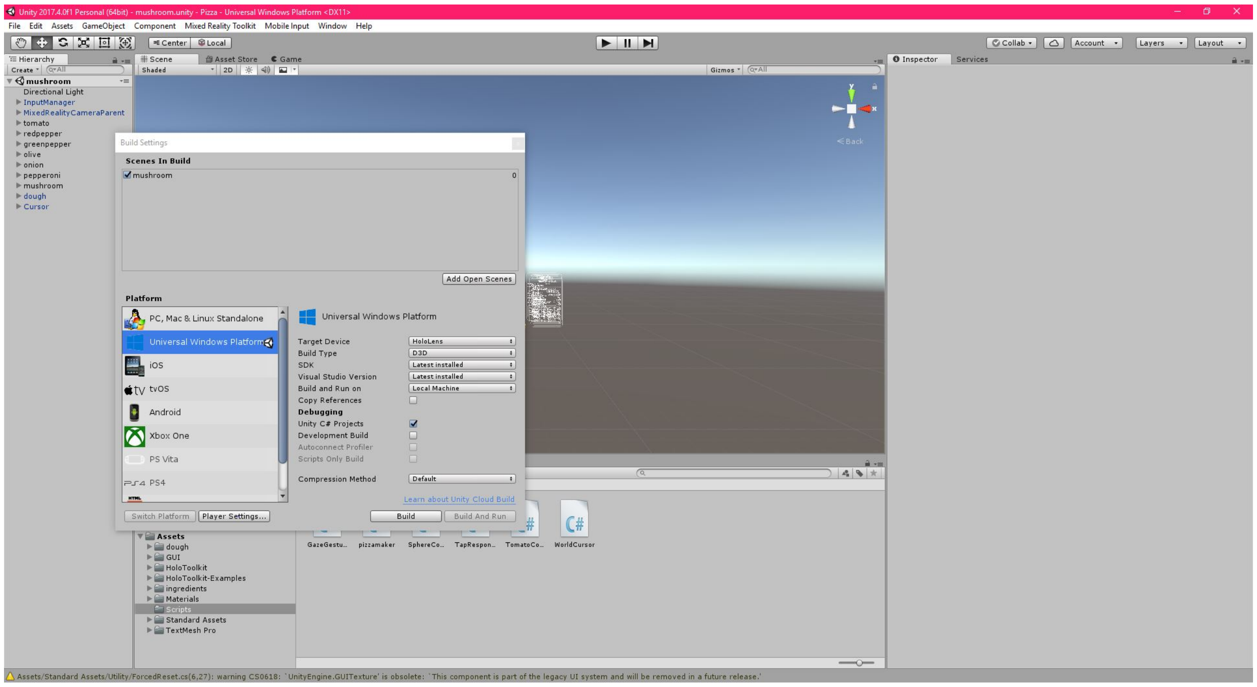


Above was the initial scene that we created with the 3D assets we imported from Photoshop. Then, we built the project in Unity in order to load it into Visual Studio 2017. Once the build process was completed, our main project folder popped up. We opened up the “FinalProjectName.sln” file to load our project into Visual Studio.

Deploying to Hololens

To run the project and see it on the Hololens, do the following:

1) Open unity and click on Build Settings under File
2) Make sure you are on Universal Windows Platform and change **Target Device** to Hololens and **Build Type** to D3D.
3) Make sure Unity C# Projects is checked

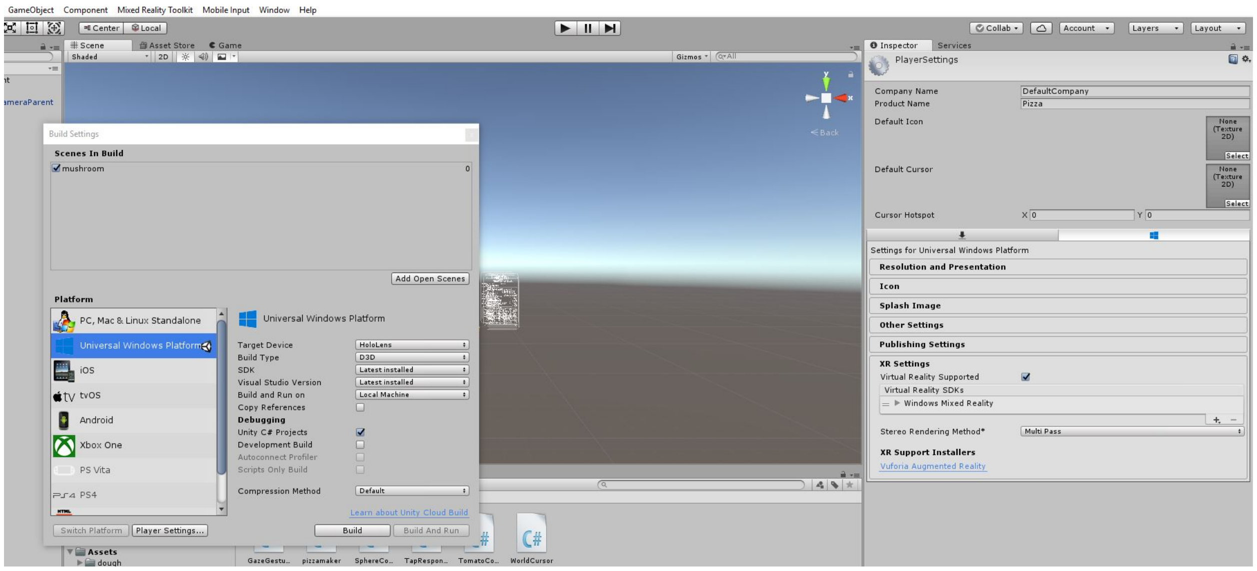


Universal Windows Platform

Target Device	HoloLens
Build Type	D3D
SDK	Latest installed
Visual Studio Version	Latest installed
Build and Run on	Local Machine
Copy References	☐
Debugging	
Unity C# Projects	✓
Development Build	☐
Autoconnect Profiler	☐
Scripts Only Build	☐
Compression Method	Default

Assets/Standard Assets/Utility/ForcedReset.cs(6,27): warning CS0618: `UnityEngine.GUITexture' is obsolete: `This component is part of the legacy UI system and will be removed in a future release.'

4) Click on **Player Settings** to open the Inspector View. In the Inspector View, go to the **XR Settings** and make sure Virtual Reality Supported is checked.


Build Settings

Scenes In Build
☑ mushroom 0

Platform: Universal Windows Platform

Target Device	HoloLens
Build Type	D3D
SDK	Latest installed
Visual Studio Version	Latest installed
Build and Run on	Local Machine
Copy References	☐
Debugging	
Unity C# Projects	☑
Development Build	☐
Autoconnect Profiler	☐
Scripts Only Build	☐
Compression Method	Default

PlayerSettings

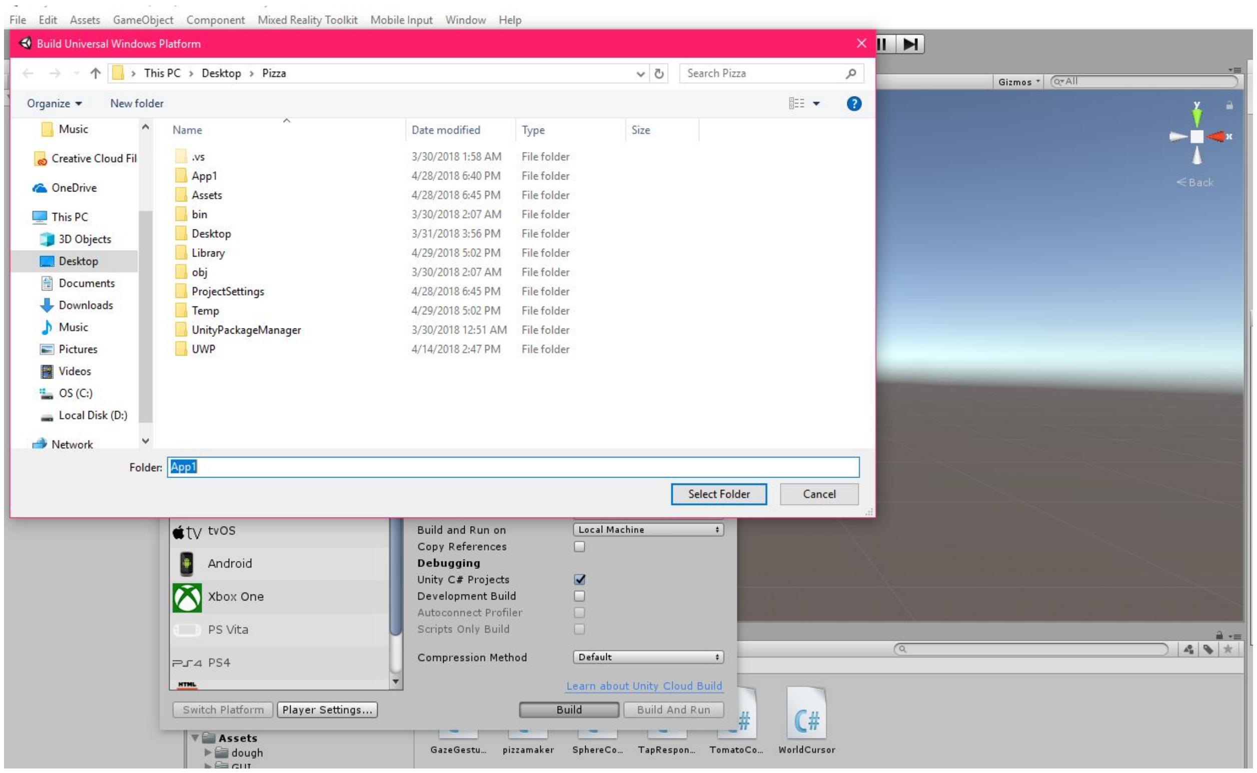
Company Name	DefaultCompany
Product Name	Pizza
Cursor Hotspot	X 0 Y 0

Settings for Universal Windows Platform

XR Settings

Virtual Reality Supported	☑
Virtual Reality SDKs	Windows Mixed Reality
Stereo Rendering Method*	Multi Pass

5) After doing the steps above, click Build and File Explorer should open up. Title the folder App and continue. **Note:** If the folder already exists, when you continue, the folder should update automatically. If it for some reason does not, delete the folder and try again.


Build Universal Windows Platform

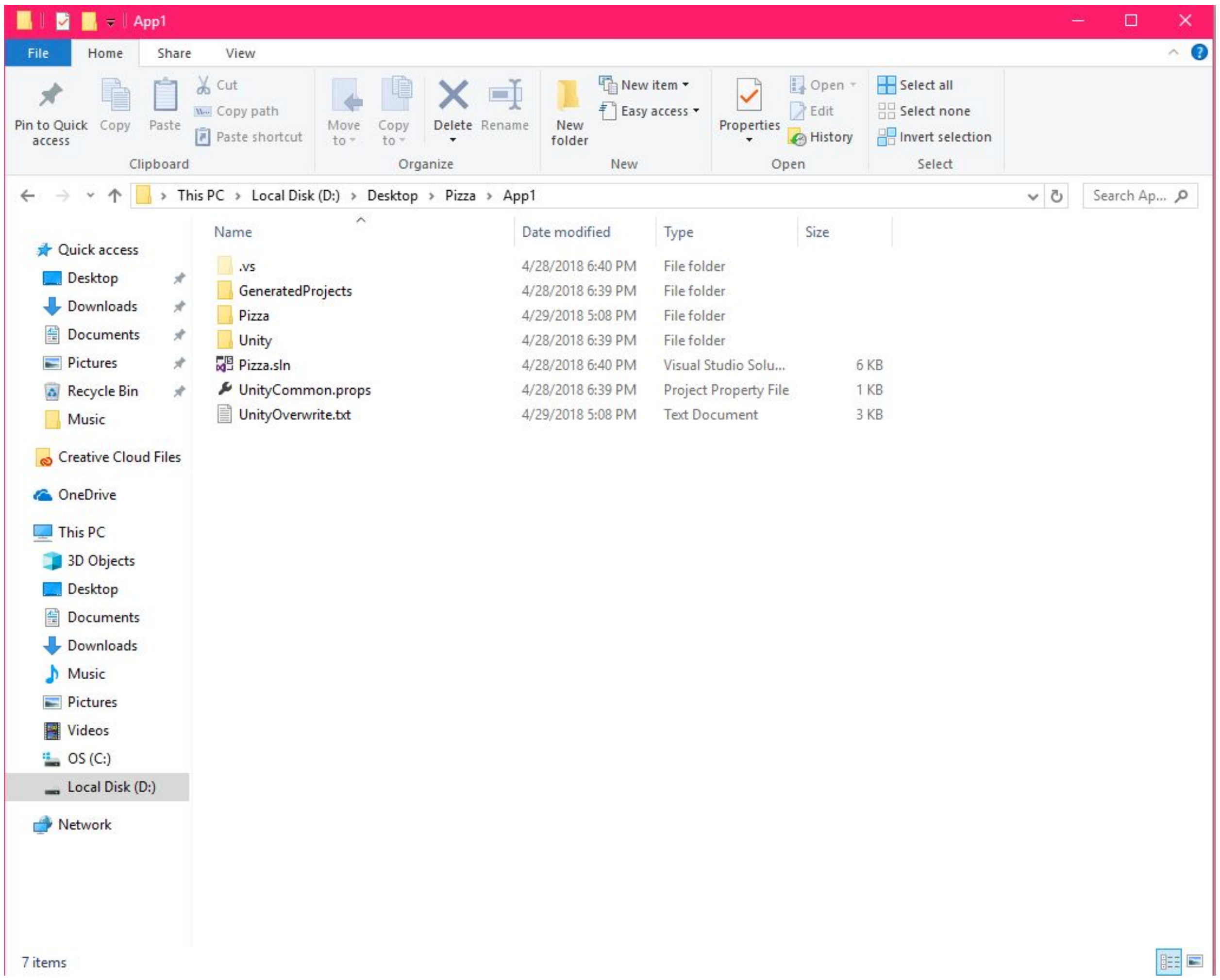
This PC › Desktop › Pizza

Name	Date modified	Type	Size
.vs	3/30/2018 1:58 AM	File folder	
App1	4/28/2018 6:40 PM	File folder	
Assets	4/28/2018 6:45 PM	File folder	
bin	3/30/2018 2:07 AM	File folder	
Desktop	3/31/2018 3:56 PM	File folder	
Library	4/29/2018 5:02 PM	File folder	
obj	3/30/2018 2:07 AM	File folder	
ProjectSettings	4/28/2018 6:45 PM	File folder	
Temp	4/29/2018 5:02 PM	File folder	
UnityPackageManager	3/30/2018 12:51 AM	File folder	
UWP	4/14/2018 2:47 PM	File folder	

Folder: App1

Select Folder | Cancel

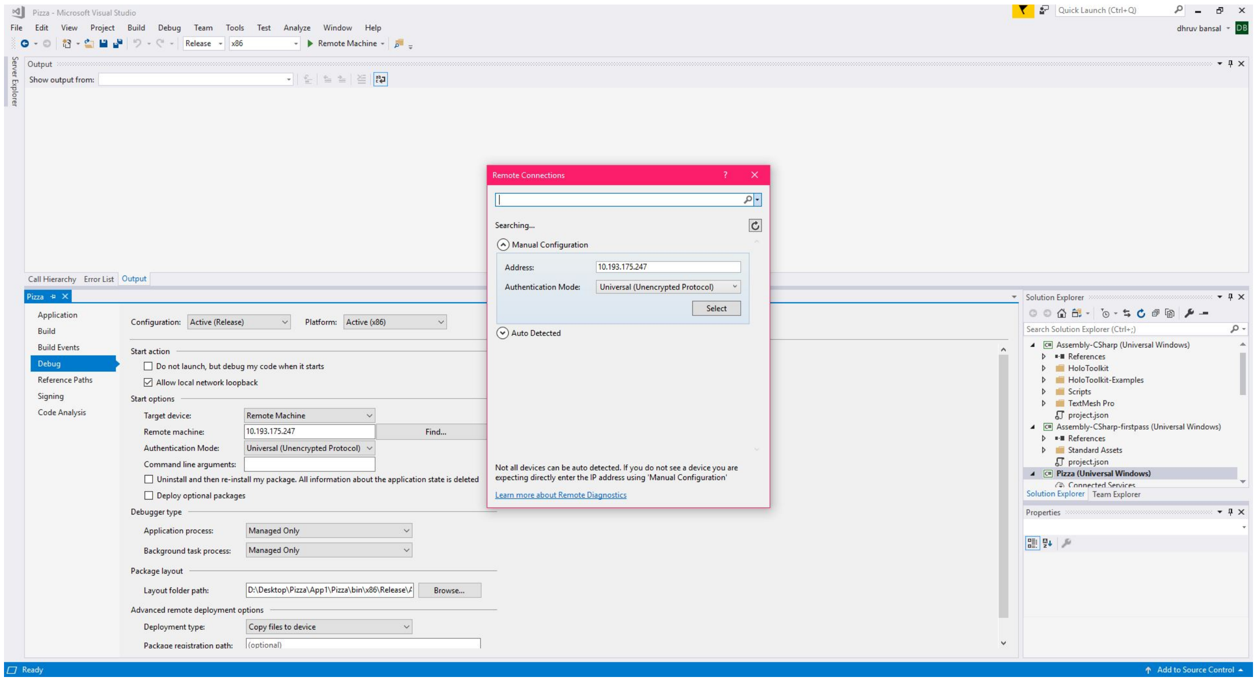
6) Once the unity project is finished building, go to the folder you created and click on Pizza.sln (or whatever the project name is). That should open up Visual Studio.



Name	Date modified	Type	Size
.vs	4/28/2018 6:40 PM	File folder	
GeneratedProjects	4/28/2018 6:39 PM	File folder	
Pizza	4/29/2018 5:08 PM	File folder	
Unity	4/28/2018 6:39 PM	File folder	
Pizza.sln	4/28/2018 6:40 PM	Visual Studio Solu...	6 KB
UnityCommon.props	4/28/2018 6:39 PM	Project Property File	1 KB
UnityOverwrite.txt	4/29/2018 5:08 PM	Text Document	3 KB

7) In Visual Studio, change **Debug** to **Release** and **ARM** to **x86**.
8) Additionally, change Local Machine to Remote Machine. That will open up a new menu called Remote Connections. **Note: If the Remote connections menu does not open go to *Debug* > Pizza Properties (or appname-properties) and then change Remote Machine to the IPv4 and press find. That should open the Remote Connections menu and then continue to step 9.**
9) In **Remote Connections,** change **Authentication Mode** to **Universal (Uncrypted Protocol).** For **Address** put the IPv4 from Hololens. To get the IPv4, go on the Hololens and go to Setttings > Network. In Network there should be an option called **Advanced Options**. Tap on that and it should have the IPv4 address. If Advanced Options is not available, make sure Developer Mode is on in the Hololens (check the installation section above for that or refer to

https://docs.microsoft.com/en-us/windows/mixed-reality/connecting-to-wi-fi-on-hololens). **Note: We connected to the Hololens over wifi, there is an option to connect over USB. To connect over USB, refer to https://docs.microsoft.com/en-us/windows/mixed-reality/using-the-window)s-device-portal**)



Remote Connections

Searching...

Manual Configuration

Address: 10.193.175.247

Authentication Mode: Universal (Unencrypted Protocol)

Auto Detected

Not all devices can be auto detected. If you do not see a device you are expecting directly enter the IP address using 'Manual Configuration'

Learn more about Remote Diagnostics

Configuration: Active (Release) Platform: Active (x86)

Start action

- [ ] Do not launch, but debug my code when it starts
- [x] Allow local network loopback

Start options

Target device: Remote Machine

Remote machine: 10.193.175.247

Authentication Mode: Universal (Unencrypted Protocol)

Command line arguments:

- [ ] Uninstall and then re-install my package. All information about the application state is deleted
- [ ] Deploy optional packages

Debugger type

Application process: Managed Only

Background task process: Managed Only

Package layout

Layout folder path: D:\Desktop\Pizza\App1\Pizza\bin\x86\Release\

Advanced remote deployment options

Deployment type: Copy files to device

Package registration path: (optional)

10) Once Remote Connections are set, go to Debug and click on **Start without Debugging**

11) If done correctly and once the code compiles successfully, on the Hololens, a screen saying **Made With Unity** will pop up automatically and the app should open automatically.

C# Coding in Visual Studio

Once the scene was successfully displayed on Hololens, we started to implement the functionalities of the app. We created a C# file in our Unity Project Folder, which was automatically opened in Visual Studio.

https://docs.unity3d.com/Manual/VisualStudioIntegration.html

We implemented two main functions for our app -- viewing the nutrition facts charts for each pizza topping and drag&dropping the toppings on the pizza dough.