AS125688

Join the party in Social VR!

Iffat Mai
Perkins + Will

Simon Chavez
Perkins + Will

Learning Objectives

- Learn how to prepare a Revit model for VR export
- Learn how to bake materials and lighting in 3ds Max
- Learn how to setup rendered model in Unity game engine
- Learn how to setup social VR using Photon VR

Description

Virtual Reality can be quite isolating if only one person can experience it all alone. Meanwhile, Social VR brings people together virtually and provides a unique opportunity for collaboration that is both immersive and interactive. Social VR collaboration is a perfect solution for the distributed project team, where team members could be physically located in different cities, states, or even countries. Using Social VR, a project team could meet while immersed in the VR model. They can review the project model while taking a virtual tour together. This session will show you how to turn your isolating VR model into a Social VR model, and create a Social VR experience where multiple participants can chat with each other, review the project virtually and take some selfie all while immersed in Virtual Reality. During the class, we will go from Revit, to 3D Max, to Unity with Photon VR for network, and create some avatar prefabs and experience the VR using Vive and Rift VR system.

Speaker(s)

Iffat Mai is the Firm wide Design Application Development Manager of Perkins and Will. During her twenty plus years of working in the AEC Technology field, Ms. Mai has shown leadership in making strategic technology decisions, developing innovative solutions and integrating cutting-edge technologies into AEC design workflow. Her recent focus has been in weaving Virtual Reality, Augmented Reality and Mixed Reality with Building Information Modeling into professional Architectural practice. Ms. Mai leads an innovation team in developing immersive AR and VR experience that enables design team to engage project stakeholders and expedite design review process on projects. As an authority on Innovative Technology and BIM custom development, Iffat has presented at various technical conferences on topics related to Computational Design, Custom Revit API and Python development. Ms. Mai holds a Bachelor of Science degree in Architecture from Massachusetts Institute of Technology.

Iffat.mai@perkinswill.com

Simon Chavez is a Visualization designer at the Perkins+Will Seattle Office. I graduated with a degree in Virtual Technology and Design in 2015. Render engines and game engines are my passion and I enjoy making the virtual that much more real.

Simon.Chavez@perkinswill.com

VR in Design

Virtual Reality (VR) has been around for a long time. Designers have always been trying to enhance their design communication capability to their project team and client. Thanks to the new VR hardware and software, VR is finally affordable enough for everyday Designers to use for all their projects.

Traditionally, designers use two dimensional plans, elevations and sections to explain their design, with the assumption that the others are capable to accurately interpret the various drawings and imagine what the 3D spatial geometry would look like. Such assumption often result in misinterpretation, which leads to requests for changes that causes delay in the project and ultimately a dissatisfied client. If a picture is worth a thousand words, then VR Experience is worth a million drawings. Virtual Reality is the perfect method for Designers to communicate and share their designs with their project team and their clients.

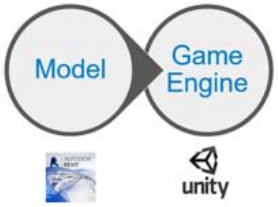
At Perkins and Will, Project teams are using VR in all phases of the project. For example, our designers used VR to help create a lobby entrance experience. As they put on the headset and walk through the entrance lobby, they can adjust the size, height and shape of the folding canopy while in VR. Designers are now able to put themselves in the shoes of the future occupants of the building they are designing, and change their design based on how an end user walking through the space would see and feel the space. VR provides the possibility of designing at different scales and different viewpoints. Virtual mockup is another great application for VR. Instead of building expensive physical mockup, our teams are letting end users experience the space in VR to verify the equipment and furniture layout. We invited teachers to visit their future classrooms and nurses to visit the nurse's station and ICU bedrooms in VR and use their feedback to improve our design.

What is Social VR

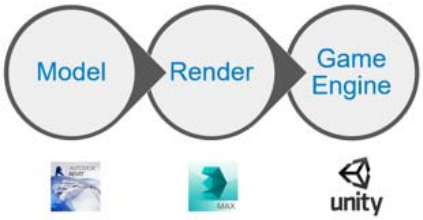
Social VR, also known as multiplayer VR or networked VR, means a Virtual Reality Experience that promotes social interaction between multiple users in the same virtual space. Typically, only one-person at a time can be in VR and that really hinders the process of collaboration. Since what the person sees in VR is quite different from what the others, who are not in VR, sees. It is also quite strange to be talking to people that are not virtually in the same space that you are in, and assume that they are seeing what you are seeing. Social VR brings all the party into the same reality, so everyone can be in the same room so to speak, and share the same experience as they explore the space around them. We wanted to add the ability for all our VR experiences to have multiple users beam in from any corner of the world, and join the party in exploring the space that we are designing together.

Workflow

The workflow for taking our design model and turn it into a social VR experience is simple. Export the 3D BIM model from the modeling software and take it into the gaming engine where we set it up for the Social VR experience. If you want a higher rendering quality for the scene, then add an extra step of rendering touch up in a rendering software.

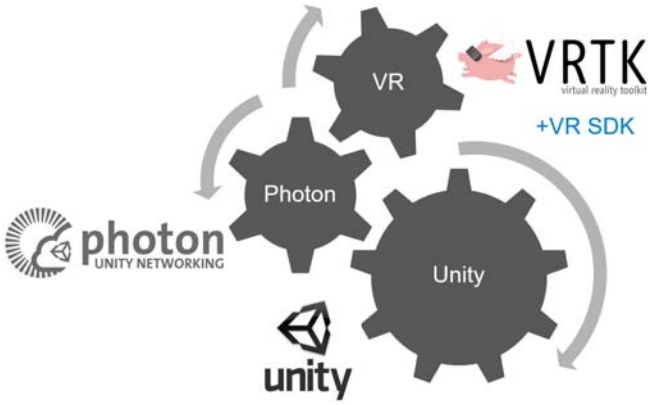


Basic Workflow



Rendered Workflow

Clearly, the most critical part of this workflow is in the game engine. We will introduce a few plugins that simplifies the process and make it easier to accomplish VR setup and networking. The two plug-ins are Photon VR and VRTK.



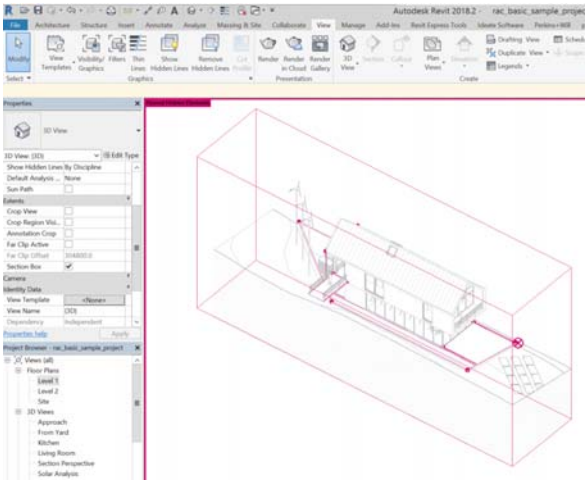
Game Engine components

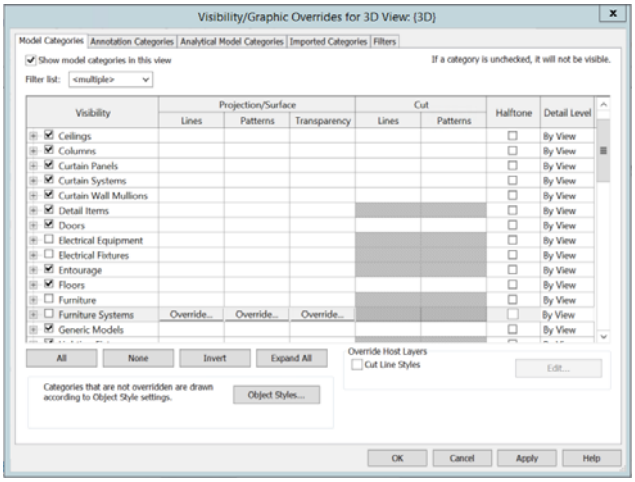
Prepare a Revit model for FBX export

The first step of creating a social VR application is to have the 3D model exported from your modeling software. Though the 3D model is typically a detailed BIM models created in Autodesk Revit, it can also be a 3D model coming from Rhino, SketchUp or 3DS Max. We will use Revit for this example. Before exporting the 3D model to FBX format, it is best to prepare the model with the following steps:

- Create a detached copy of your Revit model
- Delete anything you don't need to see
- Use Visibility/ Graphics Settings to turn off unnecessary objects
- Use Section Box to isolate what you need to see
- Export to FBX
- Break the model into sub groups by views if the model is too large.

Start by creating a detached copy of your Revit model. This will give you the freedom to delete things without worrying about messing up the Project file. The objective is to minimize the number of polygons in the scene. Create a 3D view, delete anything that you do not need to see in that view. Turn off elements that are unnecessary for the scene using Visibility Graphics settings. Use the Section Box function to define the area of things that you need in the scene. For example, if you only need to see the Lobby area and not the entire building, create a section box around the Lobby area, showing everything you can see standing inside the lobby, but hide everything else outside the lobby.

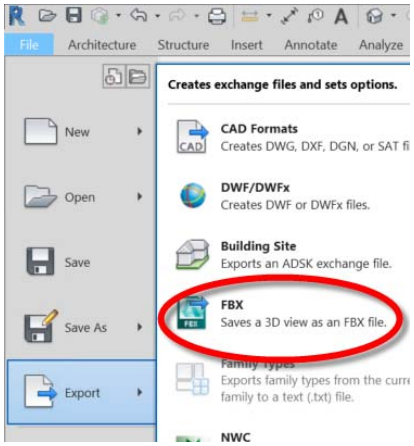




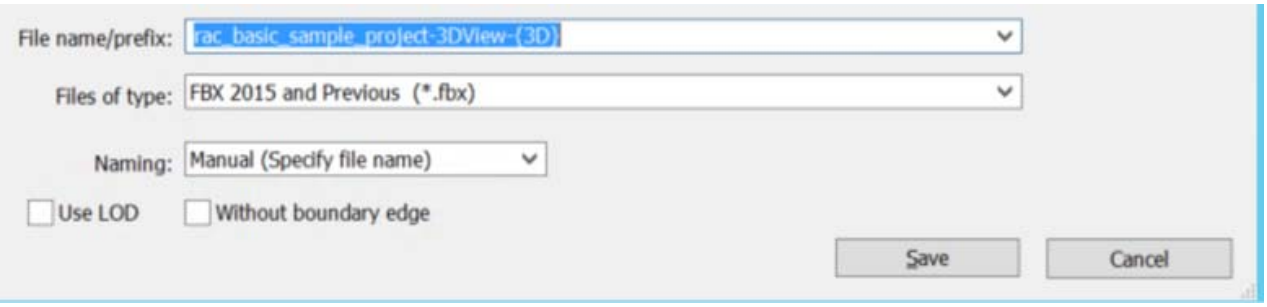
Export from Revit to FBX

To export to FBX in Revit, select from the drop down menu,

File > Export > FBX



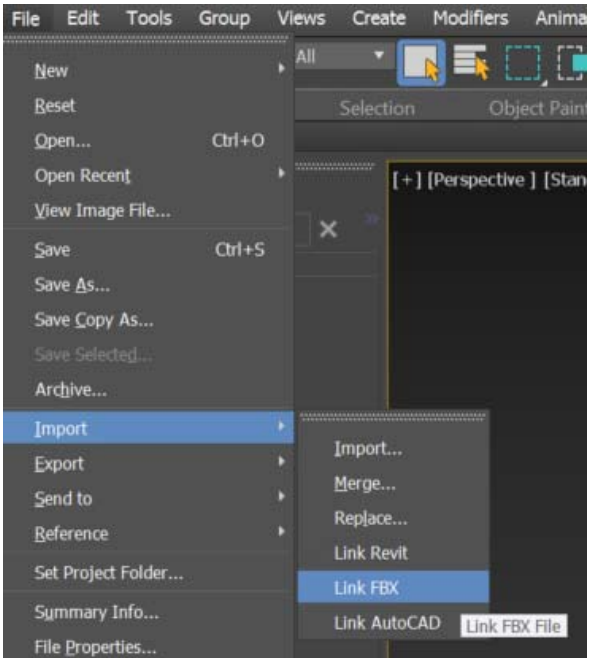
- Use LOD will create lower polygon counts with a smaller file size, but curved geometry will appear jagged.
- Without boundary edge will hide the lines between two surfaces that come together for a more natural and realistic look.

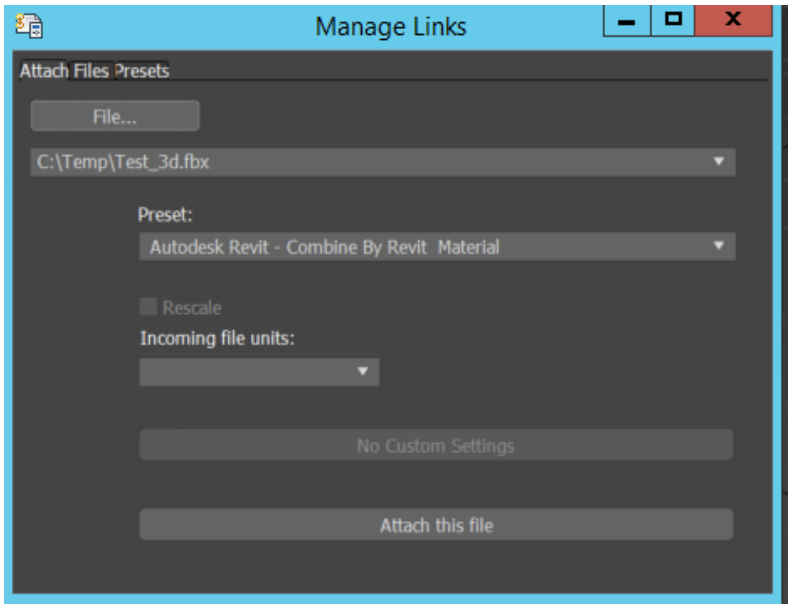


Update Material in 3DS Max

Importing the FBX directly from Revit to Unity will result in plain shaded geometries, which might be enough for early design stage where white monolithic models are sufficient for design discussion. However, for a higher quality of rendering, import the FBX model into 3D Max for lighting and material setup.

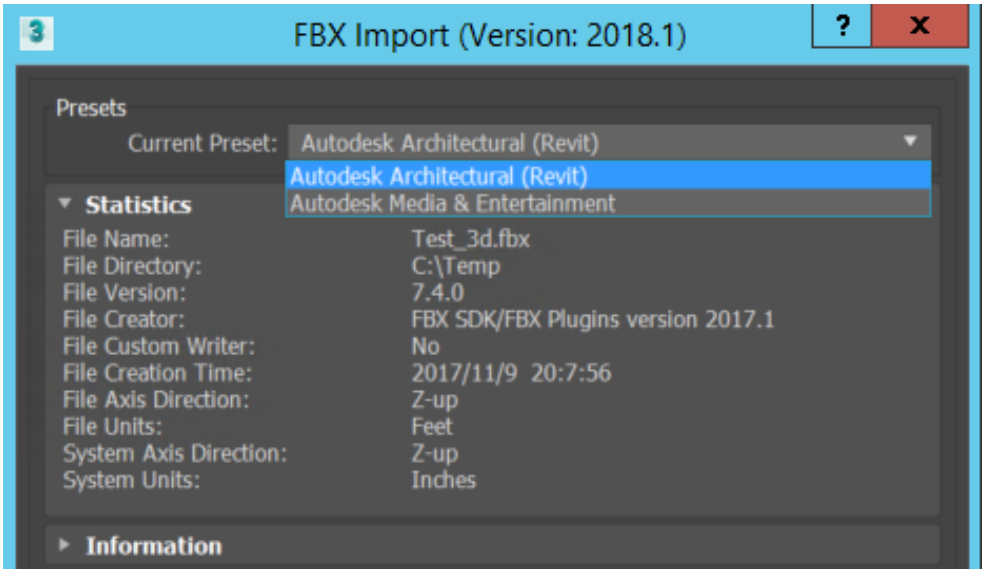
There are two ways of bringing the FBX model from Revit into 3D Max. One way is linking the FBX file into 3DS Max, the other is importing. There are pros and cons for both methods. Linking FBX files into Max allows for an easier process of continuous updates and changes between Revit and 3DS Max. Linking also offers a “Select Combine Entities By Revit Material” option in the File Link Setting dialog box. However, linking will certainly limit the flexibility of rearranging objects inside 3DS Max. On the other hand, importing FBX file into 3DS Max will offer greater flexibility in what you can do to the objects, but will not maintain the link to the FBX file, so any action taken to rearrange the objects, will need to repeated the next time you re-import an updated FBX file.





Here are some effects after selecting “Combine Entities by Revit Material”:

- Revit entities assigned the same material become a single 3ds Max object.
- The object has the name of the material
- The object is placed on a layer that also has the name of the material.
- Revit metadata (Family and Category) does not appear in the 3ds Max Scene Explorer.
- Multi/Sub-Object material information is lost.
- Reduce Polygon Count
- Group objects as systems



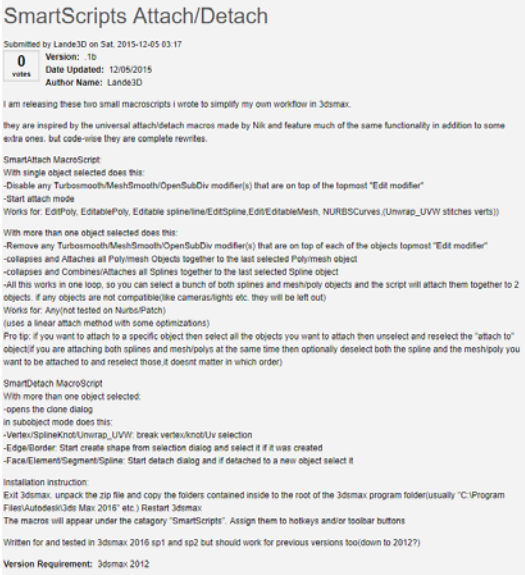
Ignore the Camera warning message, just click OK for the Camera Warnings and Errors.

Model Cleanup in 3DS Max

- Convert all objects to Editable Poly
- In editable poly, click Quick Weld
- Run Pro-Optimizer (Built-in) – visualizer
- Identify super-high poly objects, run vertex cleaner script (Shiva3d)
- After everything is optimized, select objects through proximity and materiality
- Run SmartScripts Attach/Detach for grouping (example: mullions)

Here is a list of useful cleanup scripts

- Smart attach/detach
  http://www.scriptspot.com/3ds-max/scripts/smartscripts-attachdetach
- Vertex Cleaner Script
  http://www.scriptspot.com/3ds-max/scripts/vertex-cleaner

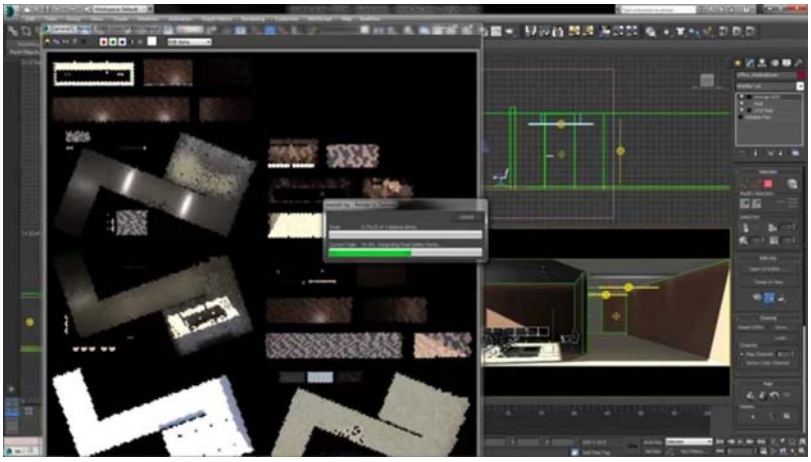


Use Scene Converter in 3DS Max

- Use Scene Converter (Rendering > Scene Converter)
- Select conversion script “Autodesk Material to Standard Material”
- Change the Standard material name to different unique material names
- Textures - Make sure your textures are sourced already from your Unity project or copied into a folder called \textures in your project
- For objects with bump maps that are not exported.
  - Converted bump map into Normal map
  - Re-assign to objects

Lighting and Materials Baking in 3ds Max

To achieve good lighting and material mapping in Unity takes a lot more work and processing power from the game engine, which can negatively affect refresh performance of the application. In VR, a dropped frame rate could cause nausea and discomfort to the VR user. Baking lighting and materials onto the objects can help create a more rendered experience without affecting the frame rate during the VR experience.
In 3ds Max, use the Render to Texture command (hotkey 0) to bake the material of the mapped surfaces into a separate image file.



Once the cleanup and lighting and materials are baked, export FBX from 3DS Max to Unity.

- Cameras and Lights are not currently imported into Unity
- If Vray lighting are baked in 3DS Max, after the model is imported into Unity, switch in Unity from standard shader to the unlit texture shader for higher efficiency and frame rate.

Setup Model in Unity for Social VR

Now that we have imported the 3D FBX Model into Unity, we need to setup the VR and networking to complete the social VR settings. We will also need to create a Prefab to represent the player and show a simple function of creating a Selfie in VR.

- Light and Material setup in Unity
- NavMesh Setup
- VR setup using VRTK
- Network setup using PhotonVR
- Create Prefabs
- Create Selfie

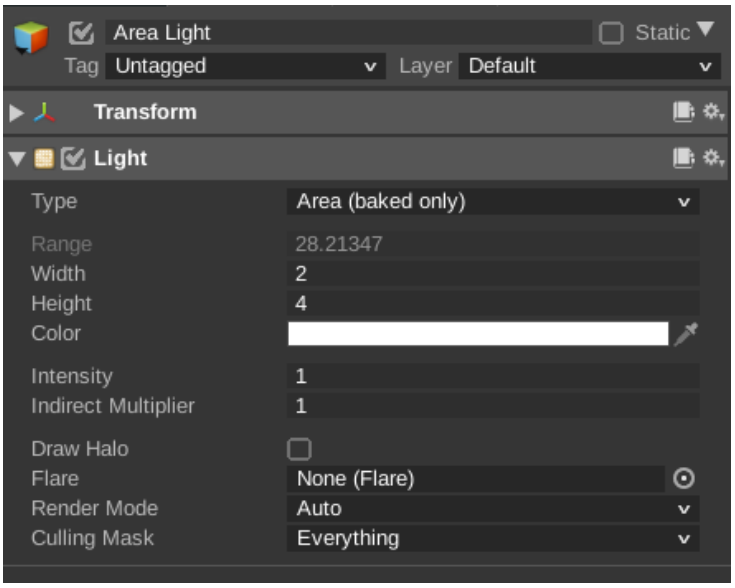
Lighting Setup in Unity

Since lights do not transfer from Revit or 3d Max via FBX into Unity, you will either bake the lighting in 3ds Max or place new lights inside Unity. There are three modes in Unity Lights. Realtime, Mixed and Baked lighting. Real time lighting is GPU intensive, and is preferred when your lights and objects are moving in the scene. With Realtime lights, dynamic shadow will be cast as objects in the scene are moved. Baked lighting is used when objects and lights do not move, pre-computing the lighting condition and baking lightmap will make the application run faster and smoother.

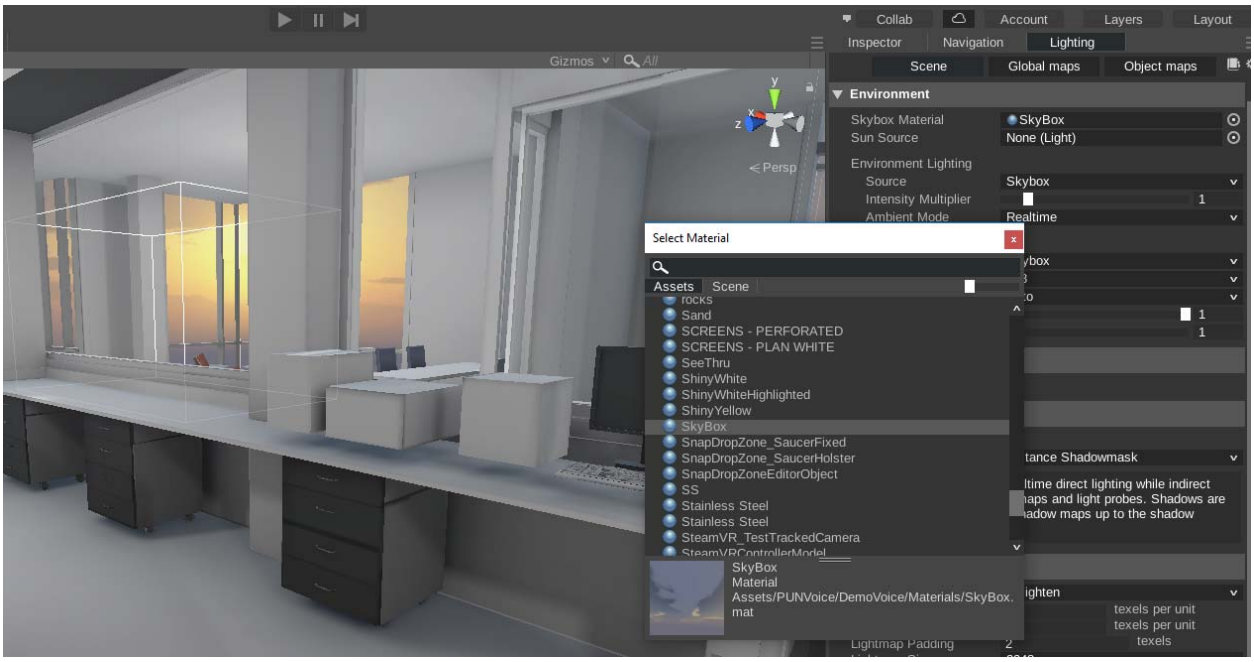
- Leave the Sun (Directional Light) in real time
- Add Area light (bake option only) to simulate ceiling mounted lights

(ex: 2x2 or 2x4 ratio, adjust for unit in ft-in)

- Change color as needed
- Use point light for downlights
- HDR Light (change default sky)



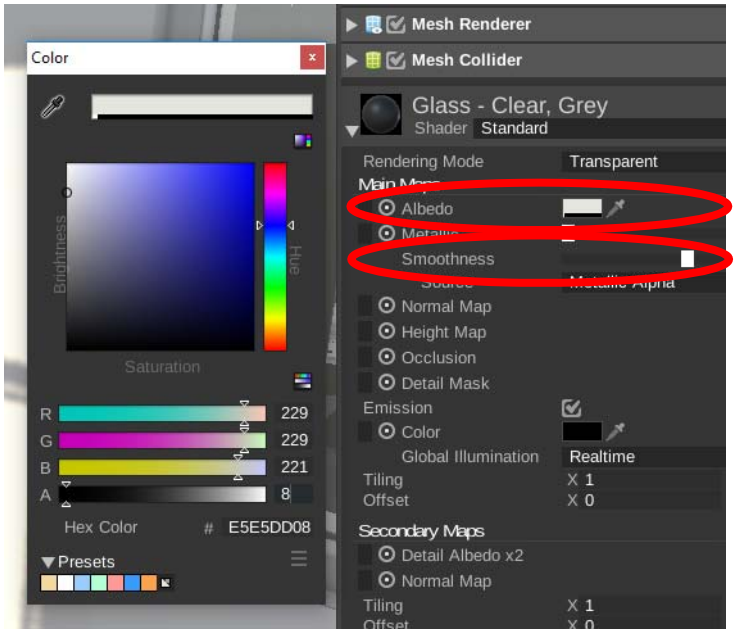
Skybox



Even though the Skybox is not a real light, it can be found under lighting tab > Environment section, where you can select a sample skybox material or any HDR images.

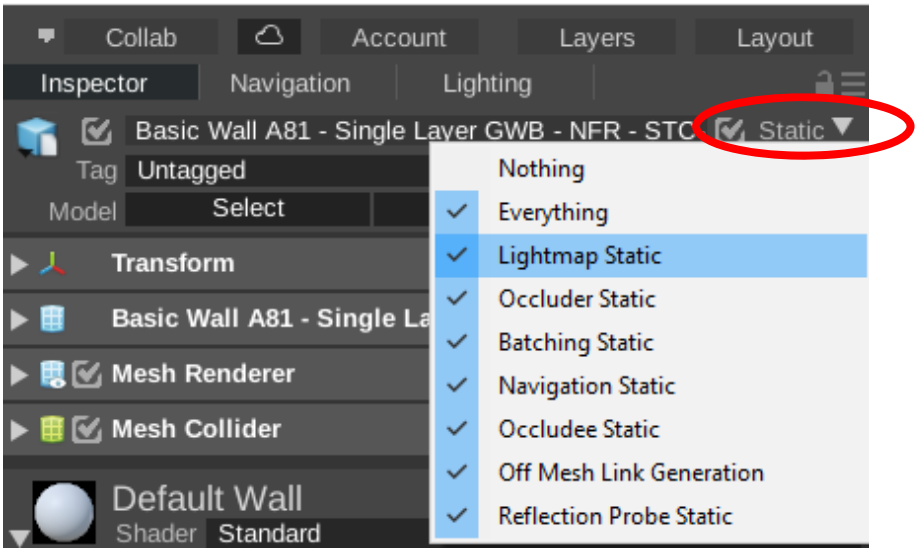
How to adjust Glass Transparency

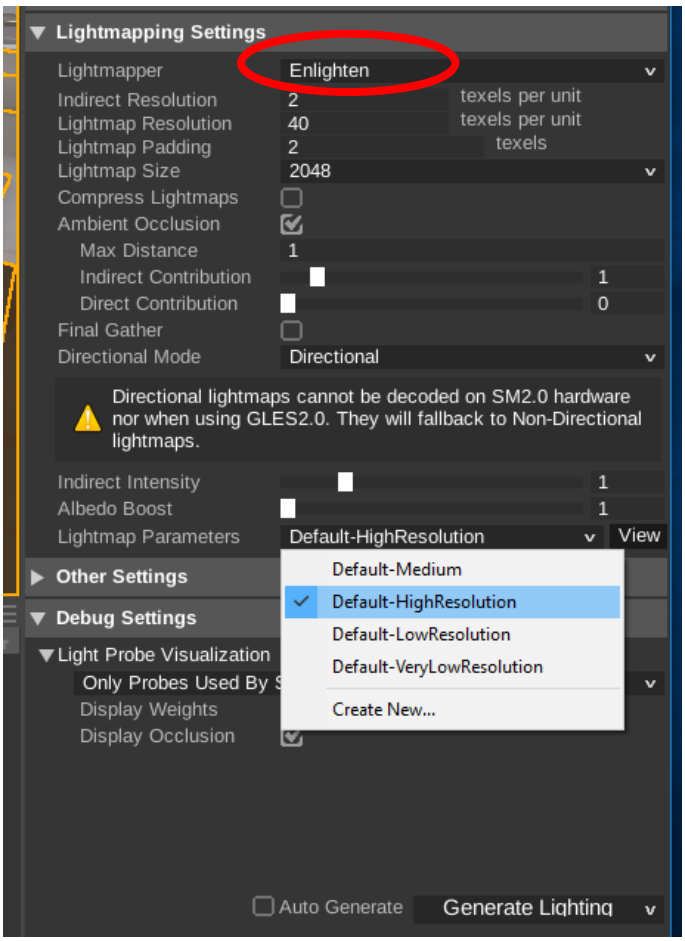
- Select the Glass Panel object
- Under Glass Shader
- Select the Albedo color
- Changing the Alpha channel value to 0 for full transparency
- Max out smoothness



Lightmap Static and Lightmapping Settings

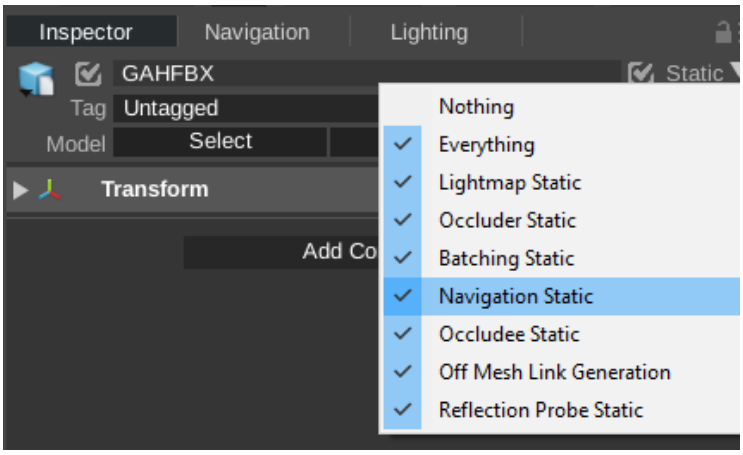
- If light baking in Unity3D, using Enlighten
- Select all non moving objects, and check Lightmap Static option
- Select all lights (other than the sun)
- Set the lights to bake mode

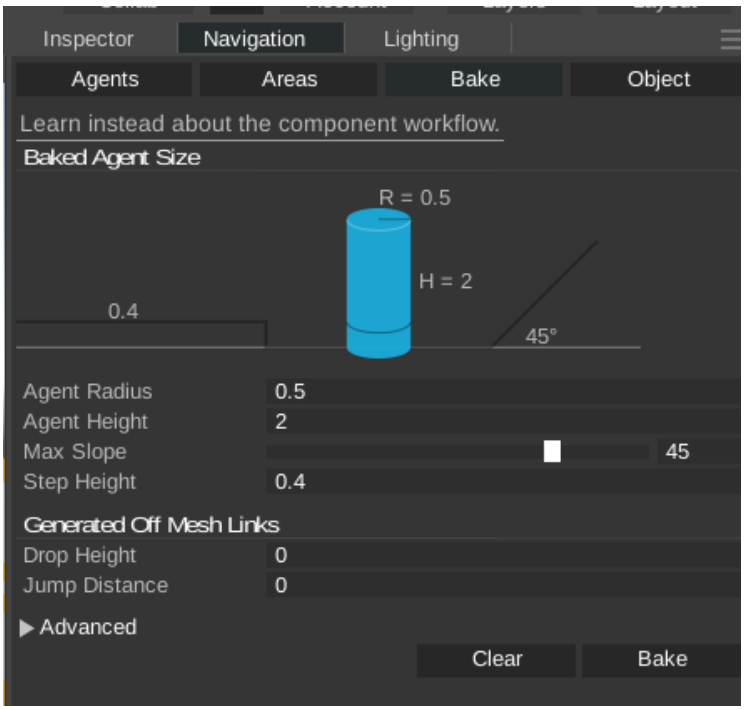




NavMesh
NavMesh is the Navigational system in Unity to guide how a player navigates in the model. Areas that are walkable vs. non-walkable areas. The full NaveMesh system contains NavMesh, NavMesh Agents, off-mesh links and NavMesh Obstacles. Here is a quick way to setup the NavMesh for your model.

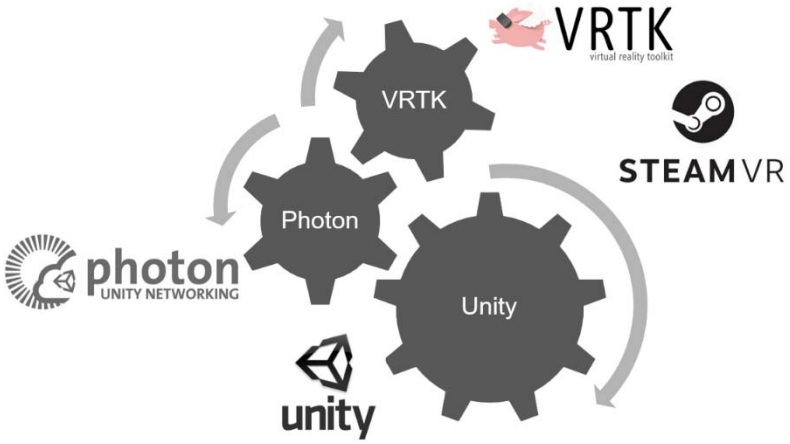
- Select all building objects
- Check Navigation Static
- **Adjust** agent size
- *Agent Radius*
- *Agent Height*
- *Max Slope*
- *Step Height*
- Open Navigation tab
- Click on Bake button
-





Unity Plug-ins

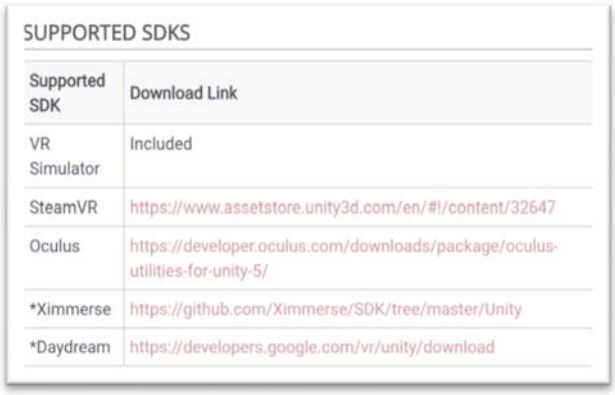
To simplify the process of setting up VR and networking, we are using two plugins for Unity. They are Photon and VRTK. Photon (Exit Games) is a third party platform for building multiplayer games in Unity, and the VRTK (Stone Fox) is a toolkit for Unity that helps developer build VR applications in Unity.



VR setup using VRTK

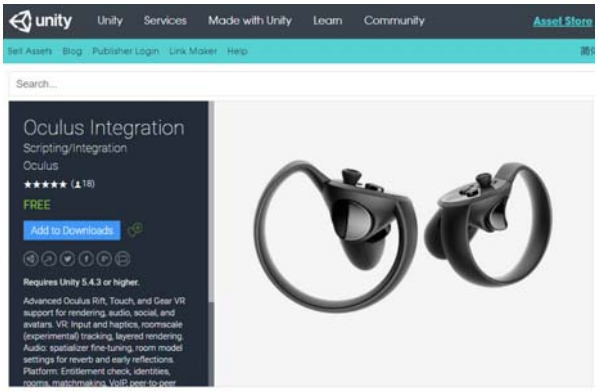
To simplify the VR setup process, we added the Virtual Reality Toolkit (VRTK) from StoneFox. The VRTK is a set of scripts that makes it easier to setup the basic VR environment. Depending on which headset system that you are using, you need to download the appropriate set of SDK codes for that system. For Vive, download the SteamVR SDK, and for Oculus Rift, download the Oculus SDK.

- Download SteamVR (Vive) or Oculus (Rift) SDK
- Download Virtual Reality Toolkit (VRTK) - https://vrtoolkit.readme.io/



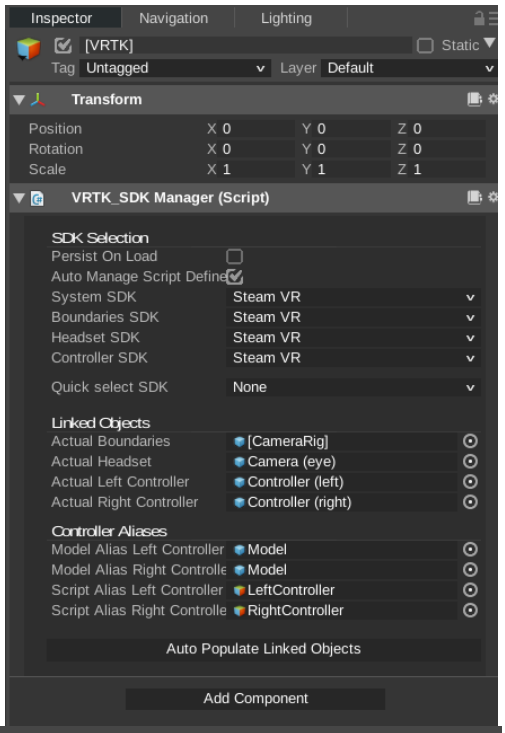
SUPPORTED SDKS

Supported SDK	Download Link
VR Simulator	Included
SteamVR	https://www.assetstore.unity3d.com/en/#!/content/32647
Oculus	https://developer.oculus.com/downloads/package/oculus-utilities-for-unity-5/
*Ximmerse	https://github.com/Ximmerse/SDK/tree/master/Unity
*Daydream	https://developers.google.com/vr/unity/download



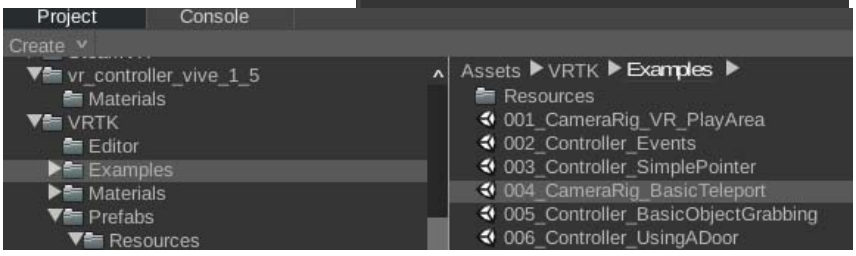
VRTK Setup Overview

- Import Assets/VRTK folder into your Unity project
- Create an empty gameobject VRTK_SDK
- Add the VRTK_SDK Manager script to the empty VRTK_SDK Game Object
- Add the SteamVR_SDK as a child to the VRTK_SDK



VR Setup Shortcut

- Open Example Scene (Basic Teleport)
- Create an empty gameobject at 0,0,0
- Rename it “Player”
- Then drag the following items into the player
  - [CameraRig]
  - [VRTK]
- Then make a prefab out of the player

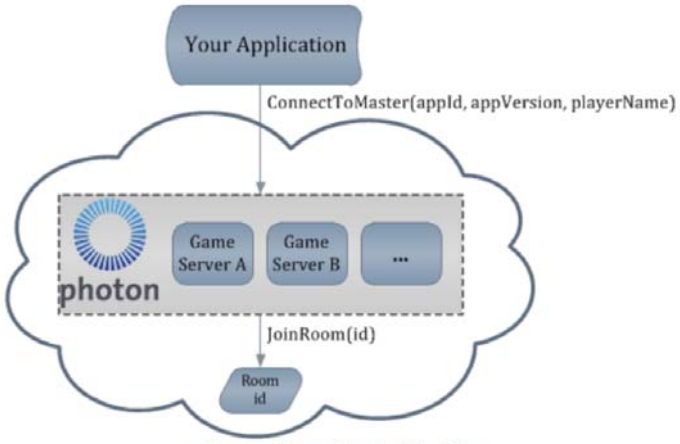


Network setup using PhotonVR

Even though Unity has built in networking capability, Unity networking is not cloud based, and it is more complex to setup and configure and it pales in performance and connectivity compared to the Photon Networking.

- **Host Model** – Photon offers dedicated server, instead of using one of the player act as a server host.
- **Connectivity** – Photon has guaranteed connectivity with high success rate. It does not fail like Unity Networking due to firewalls/routers etc.
- **Performance** – Lower latency than Unity Network due to better connection
- **Features & Maintenance** –The Photon solution is actively maintained and parts of it are available with source code.
- **Master Server** - Photon Server that posts room-names of currently played games in "lobbies". It will forward clients to the Game Server(s), where the actual gameplay happens.

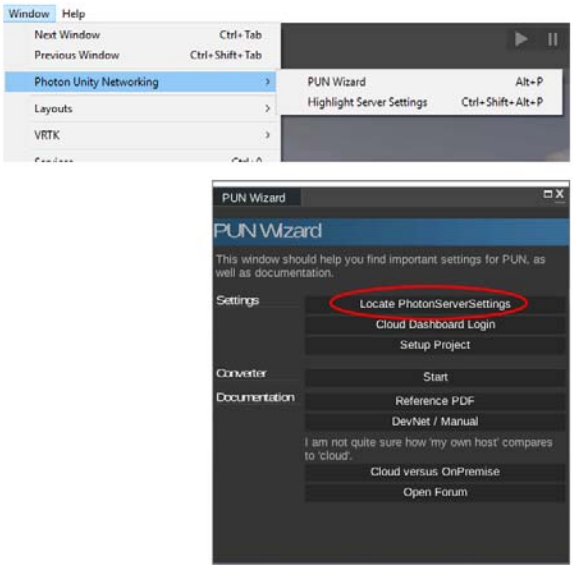
Photon VR is one of the most popular online multiplayer platform. It has simplified the process of networking multiplayer with a basic toolkit that is free for up to 20 users. Download the plugin from the Unity Asset store, and register on the Photon website to setup the account and obtain the Photon App ID.

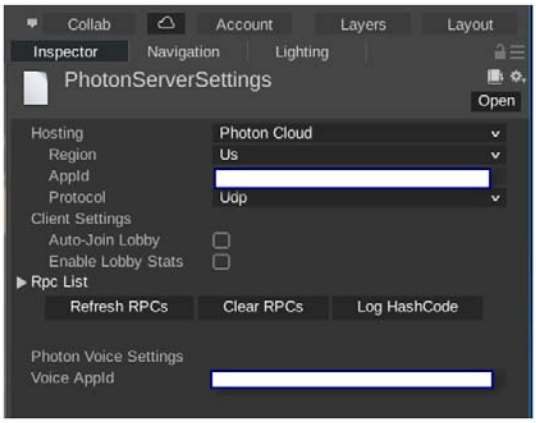


Photon Cloud Simple Workflow

- Install the Photon Unity Networking package from the Unity Asset store
- Setup Photon account
- Implement server creation and joining an existing host.
- Spawning as a player and create objects on the network.
- Network communication using State Synchronization and Remote Procedure Calls.
- Interpolating and predicting values between data packages.

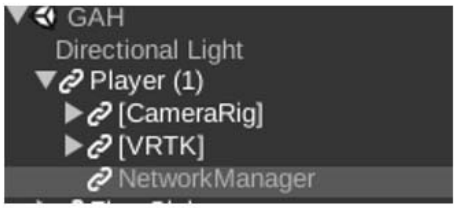
Download and install the Photon package into the Unity Project, the Photon Unity (PUN) wizard is in the Window> Photon Unity Networking> PUN Wizard, Select the locate Photon Server Settings where you can enter your own app ID from the account you just setup.

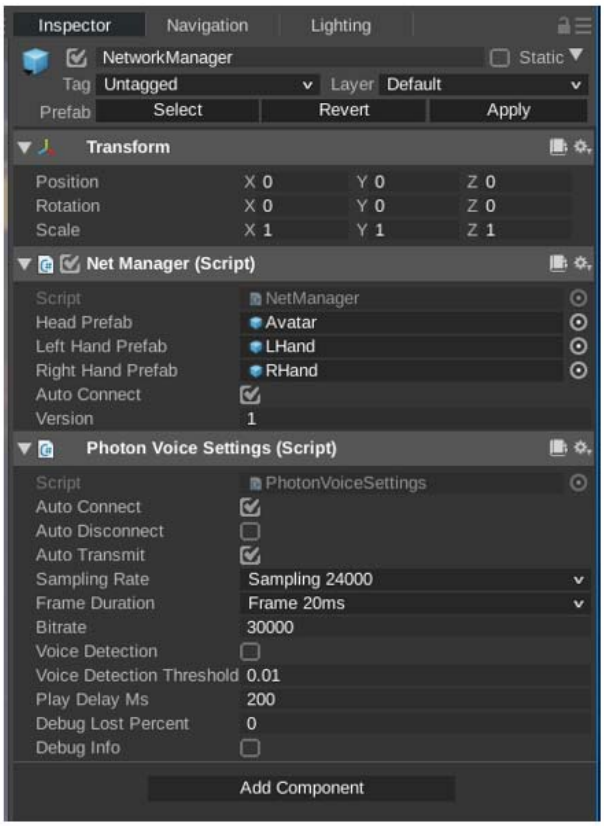




Create Network Manager

- Create an empty game object named NetworkManager under Player
- Create Net Manager Script (see sample)
- Create Photon Voice Settings script

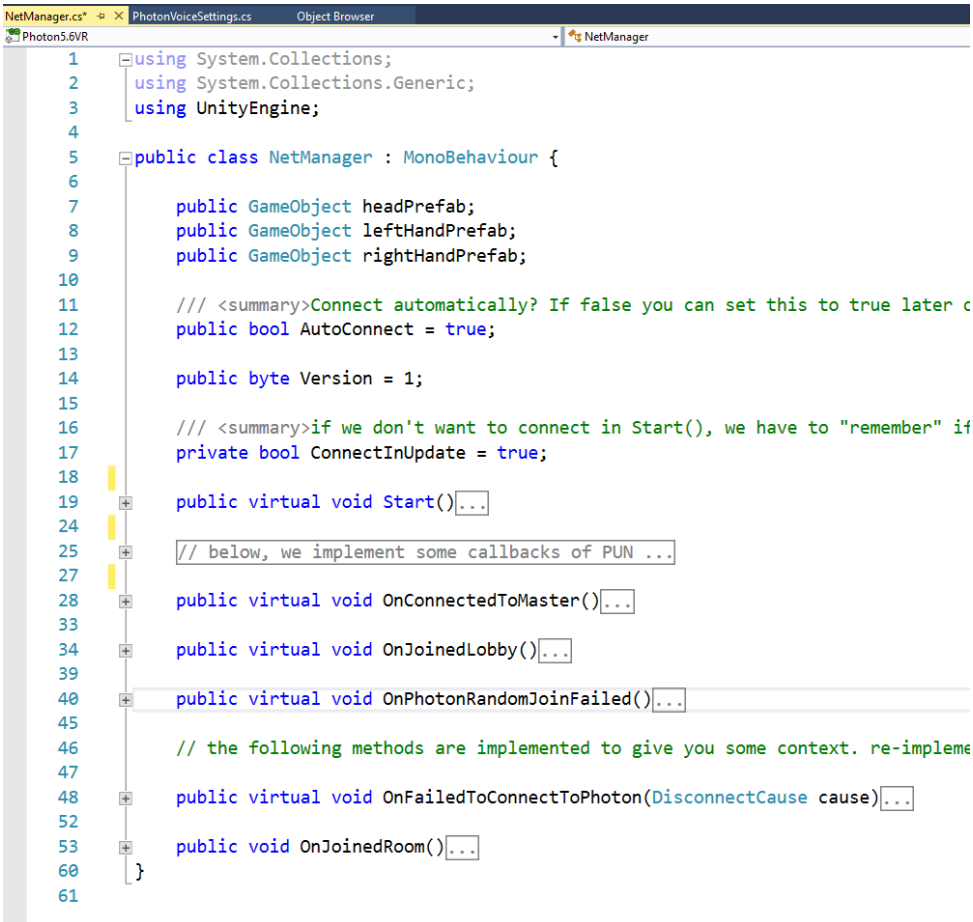




Net_Manager script

- Add connection string to Start()
- Add OnConnectedToMaster() ➔ PhotonNetwork.JoinRandomRoom()
- Add OnPhotonRandomJoinFailed() ➔ PhotonNetwork.CreateRoom()
- Add OnFailedToConnectoToPhoton() ➔Disconnect if failed to join
- Add OnJoinedRoom ➔ Successfully joined a room
- ➔ Instantiate game objects
- PhotonNetwork.Instantiate(PrefabName, position, rotation, group, data)

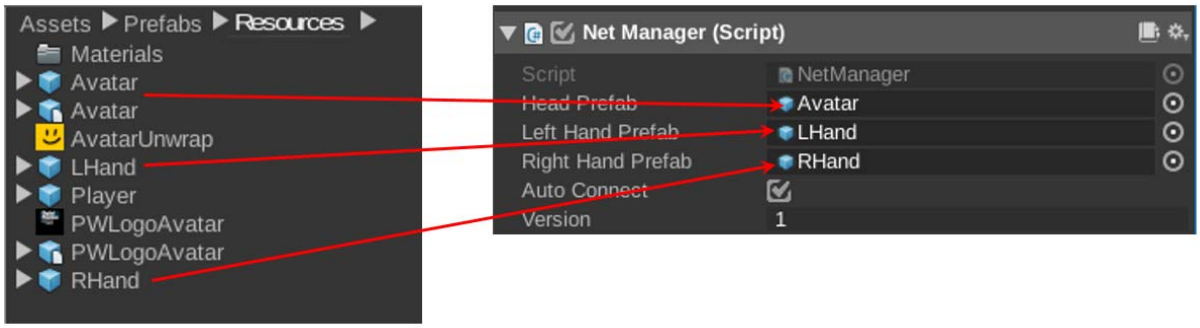
```
public class NetManager : MonoBehaviour {

    public GameObject headPrefab;
    public GameObject leftHandPrefab;
    public GameObject rightHandPrefab;
```




```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class NetManager : MonoBehaviour {

    public GameObject headPrefab;
    public GameObject leftHandPrefab;
    public GameObject rightHandPrefab;

    /// <summary>Connect automatically? If false you can set this to true later c
    public bool AutoConnect = true;

    public byte Version = 1;

    /// <summary>if we don't want to connect in Start(), we have to "remember" if
    private bool ConnectInUpdate = true;

    public virtual void Start()...

    // below, we implement some callbacks of PUN ...

    public virtual void OnConnectedToMaster()...

    public virtual void OnJoinedLobby()...

    public virtual void OnPhotonRandomJoinFailed()...

    // the following methods are implemented to give you some context. re-impleme

    public virtual void OnFailedToConnectToPhoton(DisconnectCause cause)...

    public void OnJoinedRoom()...
}
```

```
public void OnJoinedRoom()
{
    Debug.Log("OnJoinedRoom() called by PUN. Now this client is in a room. From here on,
    PhotonNetwork.Instantiate(headPrefab.name, ViveManager.Instance.head.transform.positi
    PhotonNetwork.Instantiate(leftHandPrefab.name, ViveManager.Instance.leftHand.transfor
    PhotonNetwork.Instantiate(rightHandPrefab.name, ViveManager.Instance.rightHand.transf
}
```

Prefabs must reside under a folder named “Resources”. Associate the prefab by dragging it from the Resources folder into the net manager script’s global variable slots for each prefab.



Player Prefab

To represent the player in the scene, you need to have a 3D model of an object representing the avatar, which represents the head, and 3D models for Left and Right hand. The model could as simple as a box, or as elaborate and as realistic as a human. We went with the abstract box. We also added a custom logo to the front of the box

- use any 3D model, double check orientation
- Y as UP, -Z as Forward (FBX export)
- Use the headset model from SteamVR folder
- C:\Program Files (x86)\Steam\steamapps\common\SteamVR\resources\rendermodels\generic_hmd

Drag the HMD folder into Asset folder

Selfie

Assign a function to the controller’s trigger, when the trigger is pressed, the code creates a new camera screen, showing a backfacing camera at the end of the controller, which the user can use to take a selfie. The image can then be saved out as a PNG file.

A very quick way to make a live selfie camera in your game:

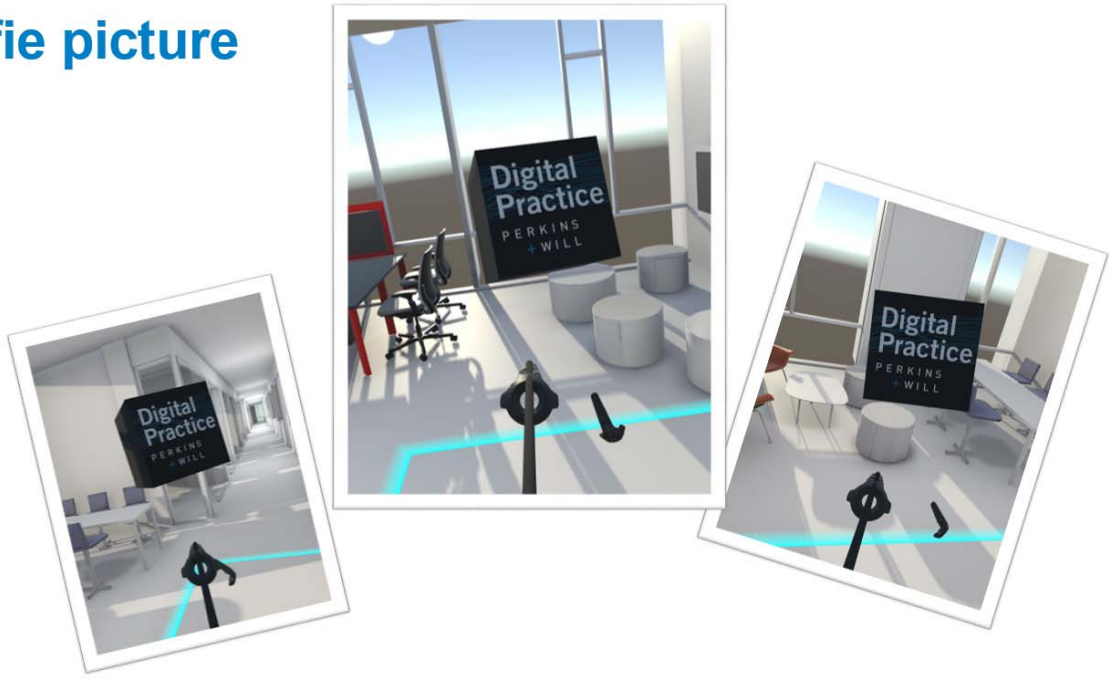
1. Create a new Render Texture asset using **Assets >Create >Render Texture**.
2. Create a new Camera using **GameObject > Camera**.
3. Assign the Render Texture to the **Target Texture** of the new Camera.
4. Create a wide, tall and thin box to represent the selfie camera screen
5. Drag the Render Texture onto it to create a Material that uses the render texture.
6. Enter Play Mode, and observe that the box’s texture is updated in real-time based on the new Camera’s output.

To save the file as a PNG, use Texture2D’s EncodeToPNG method to stream the PNG info into a saved file

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class HiResImage : MonoBehaviour {

    public int resWidth = 2550;
    public int resHeight = 3300;

    public SteamVR_TrackedObject controller;

    public Camera camera;

    private bool takeHiResShot = false;

    public static string ScreenShotName(int width, int height)
    {
        return string.Format("{0}/screenshots/screen_{1}x{2}_{3}.png",
                             Application.dataPath,
                             width, height,
                             System.DateTime.Now.ToString("yyyy-MM-dd_HH-mm-ss"));
    }

    public void TakeHiResShot()
    {
        takeHiResShot = true;
    }

    void LateUpdate()
    {
        var device = SteamVR_Controller.Input((int)controller.index);
            takeHiResShot |= device.GetTouchDown(SteamVR_Controller.ButtonMask.Trigger);
        if (takeHiResShot)
        {
            RenderTexture original = camera.targetTexture;
            RenderTexture rt = new RenderTexture(resWidth, resHeight, 24);
            camera.targetTexture = rt;
            Texture2D screenShot = new Texture2D(resWidth, resHeight, TextureFormat.RGB24, false);
            camera.Render();
            RenderTexture.active = rt;
            screenShot.ReadPixels(new Rect(0, 0, resWidth, resHeight), 0, 0);
            camera.targetTexture = original;
            RenderTexture.active = null; // JC: added to avoid errors
            Destroy(rt);
            byte[] bytes = screenShot.EncodeToPNG();
            string filename = ScreenShotName(resWidth, resHeight);
            System.IO.File.WriteAllBytes(filename, bytes);
            Debug.Log(string.Format("Took screenshot to: {0}", filename));
            takeHiResShot = false;
```

- https://answers.unity.com/questions/22954/how-to-save-a-picture-take-screenshot-from-a-camer.html

Selfie picture



Selfie Group Shot!