Extract from

NCTech Application Notes & Case Studies

Download the complete booklet from

nctechimaging.com/technotes

Application Note – Using Vuforia to Display Point Clouds and Meshes in Augmented Reality

Date: 5 December 2017
Author: Chantelle Mundy, VR Applications Developer
Organisations involved: NCTech
Products used: Unity, Vuforia, LASirisVR

This Application Note aims to explain the workflow used to create a mobile augmented reality (AR) application to display point cloud or mesh data. The app combines Unity3d with the Vuforia SDK to create an augmented reality experience, and utilises a Unity Asset Store package to convert a point cloud into a Unity Prefab. This App Note will also discuss the limitations of target-based AR experiences, as well as the differences between Android and iOS devices which impact the content of the AR application.

01. Software Installation

Unity 2017.2 (This version of Unity comes with Vuforia Editor tools)
www.unity3d.com

Point Cloud Free Viewer
www.assetstore.unity3d.com

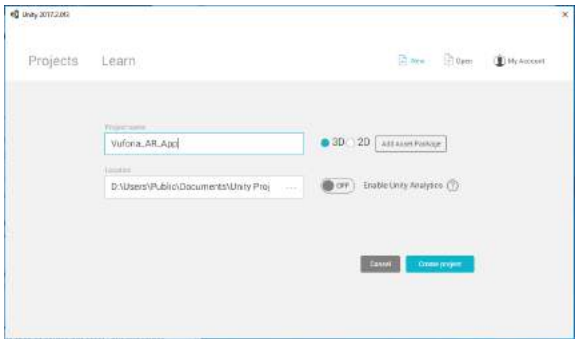
Xcode 6 or later[1] (iOS only)
www.developer.apple.com

Meshlab[2]
www.meshlab.net

Android Studio/SDK (Android only)
www.developer.android.com

02. Setting Up the Unity Project

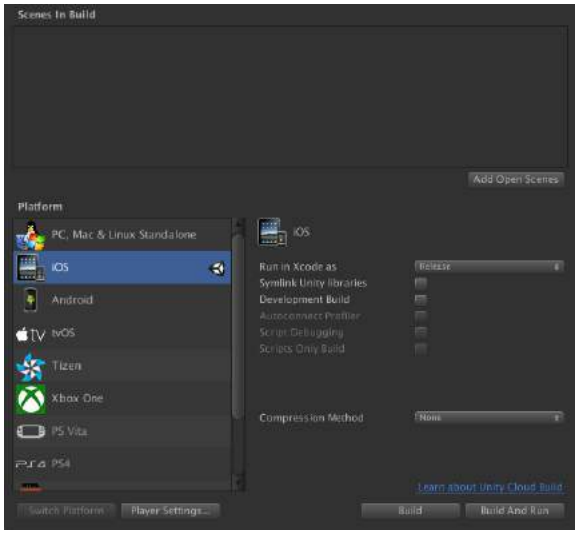
This workflow uses Unity 2017.2, which comes with Vuforia support built-in, to make the mobile application. Begin by opening Unity and creating a new 3D project by selecting "New" and ensuring that the "3D" option is checked; name your project and click "Create Project" to open up the project in the Unity Editor.



Img. 02.01. Creating a new Unity Project.

02.01. Setting the Build Platform

Once the Editor window opens, the project must be configured to support the platform that you want to run the app on; select "File" > "Build Settings", then choose either "Android" or "iOS" from the platform choices and click "Switch Platform" to configure the project. After this is complete, the Build Settings window can be closed.

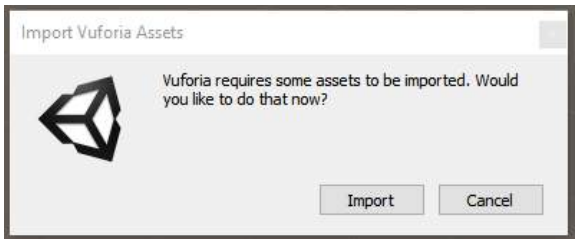


Img. 02.02. Selecting the Platform that the app will be created for.

02.02. Enabling Vuforia Support

While Vuforia is integrated with the Editor in Unity 2017.2, the assets required to make the AR app need to be imported into the project and Vuforia AR Support must also be enabled in the Player Settings. To enable this, locate the "Vuforia Augmented Reality Supported" checkbox in "Edit" > "Project Settings" > "Player" > "XR Settings" and ensure that it is checked.

Next, the Vuforia assets should be imported by clicking on the "Create" tab in the Hierarchy which can be found in the Editor window. From the drop-down select "Vuforia", followed by "AR Camera" to bring up a window that will prompt you to import the assets.



Img. 02.03. Importing the Vuforia assets into the project.

Click the "Import" option in the pop-up window; once the package has been imported there will be a new ARCamera object in the Hierarchy and several new folders will appear in the Assets folder. These folders contain the components which will allow you to build the AR application later on, so do not remove them from the project. Finally, save the current scene via "File" > "Save

[1] Xcode is only required for developing iOS apps; to install Xcode, you must sign up for a free Apple Developer account and will need to be using a Mac. For more information on how to create an Apple Developer Account see Section 07.01.

[2] Meshlab is only required if the AR application will display a point cloud and is used to convert the point cloud to *.OFF format; other applications can be used as an alternative provided that they output *.OFF files.

Scenes" and give it an appropriate name such as "AR Scene"; this scene will form the basis of the augmented reality application.

03. Converting Point Clouds (Android Only)

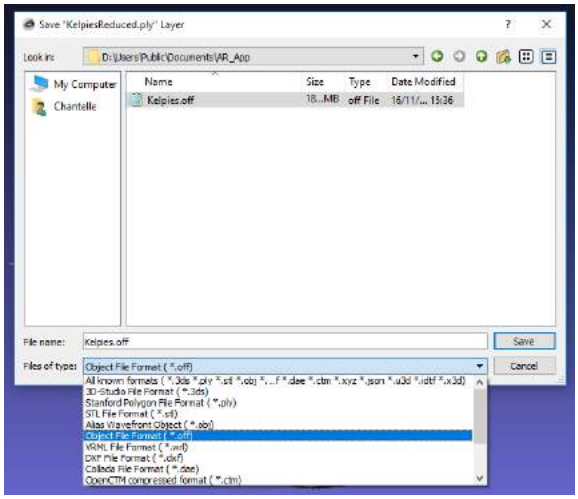
It is possible to convert point cloud files into a Unity Prefab using a free plugin available from the Unity Asset Store; however due to differences in the rendering capabilities of Android and iOS devices, point clouds can only be displayed on Android devices. If creating the app for an iOS device or a weaker Android device, meshes should be used as an alternative.

03.01. Point Cloud Conversion to *.OFF Format

The plugin that is used in this workflow requires the point cloud file to be in *.OFF format so that it can be converted to a Prefab. The plugin is capable of converting point clouds of up to 10 million points, however we recommend sampling your point cloud until it has approximately 4.6 million points which offers the optimum performance on Android devices.

To convert a point cloud file to *.OFF format this workflow makes use of the open source mesh editing software, Meshlab. Open Meshlab and select "File" > "Import Mesh" and select the point cloud file that needs to be converted; note that Meshlab supports a limited amount of file formats so ensure that the point cloud file has been saved as one of these supported formats - in this Application Note, the point cloud file was saved as a *.PLY file before importing it into Meshlab.

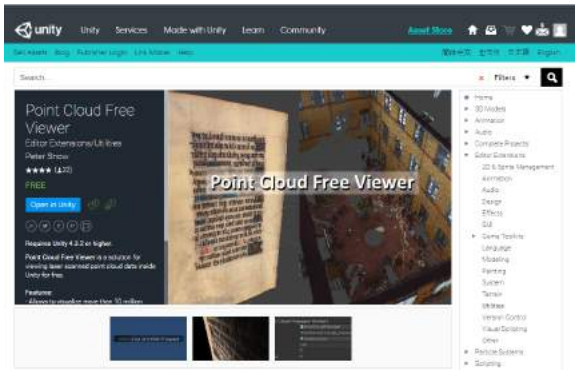
Once the point cloud has finished importing into Meshlab, select "File" > "Export Mesh As" which will open the Export window. Click on the "Files of Type" drop-down and select "Object File Format (*.off)" as the file type to export to; press the "Save" button to export the point cloud, and then, ensuring that the "Color" option under the "Vert" column is ticked, press "OK" to confirm the export options.



Img. 03.01. Exporting the point cloud as an *.OFF file in Meshlab.

03.02. Importing the Unity Plugin

Follow the link to the Point Cloud Free Viewer (see Section 01.) which will take you to the Unity Asset Store. Either sign in to a Unity account or make one if you do not currently have one, and click the option on the Point Cloud Free Viewer page to "Add to Downloads"; this will make it easier to import the plugin into the project.



Img. 03.02. The Unity Asset Store showing the free plugin that will be used to convert the point cloud into a Unity Prefab.

Next, ensuring that your Unity project is open, click on the option to "Open in Unity" which will open the plugin page inside the Unity Editor. Click on the "Import" button on the store page to add the plugin to your project; a pop-up window will appear, allowing you to choose what parts of the plugin you wish to import. Ensure that all of the assets are ticked and then click "Import" to begin importing the plugin into the project. Another pop-up window prompting you to update the API of the project will appear; click on "I Made a Backup. Go Ahead!" to continue.

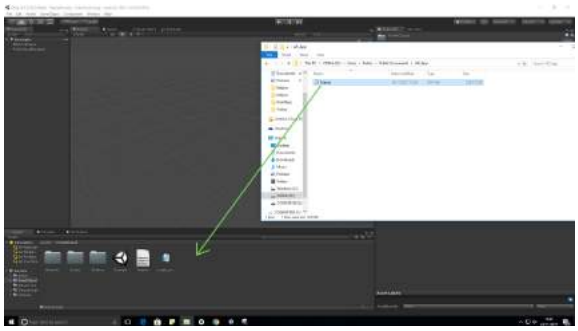
Img. 03.03. The import pop-up window showing the plugin components that can be imported.

03.03. Using the Plugin

Once the plugin has finished importing, open up the folder called "PointCloud" that can be found in the Project Window; within this folder is where the plugin assets are stored. This plugin provides an example scene that can be used to convert a point cloud into a Prefab; double click on the asset called "Example" to open up the example scene.

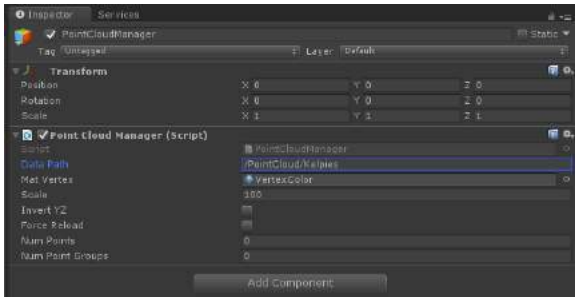
Img. 03.04. Locating the Point Cloud Free Viewer plugin assets.

Outside of the Unity Editor, open up a File Explorer or Finder window and locate the *.OFF point cloud file. Next, drag and drop this file into the PointCloud folder in the Assets section of the Unity Editor to import it into the project[3].



Img. 03.05. Adding the *.OFF file to the Unity project.

In the example scene there is a GameObject called "PointCloudManager" which can be found in the project Hierarchy; click on this GameObject to bring up its components in the Inspector window. Under the "Point Cloud Manager (Script)" component, there is a parameter called "Data Path" which is the path that the plugin will use to access the *.OFF point cloud file. Change the "xyzrgb_manuscript" part of the existing Data Path to the exact name of your point cloud file. In this example, the Data Path is "/PointCloud/Kelpies" because the *.OFF file is located in the PointClouds folder and is called "Kelpies".



Img. 03.06. The Inspector window showing the components of the PointCloudManager GameObject and the new Data Path.

Depending on the size of the point cloud that is being used, the "Scale" parameter in the "Point Cloud Manager (Script)" component may have to be altered; a good base value to set the Scale to is 1, but the scale of the produced Prefab can be changed later on if it is too large/small. Finally, tick the option to "Force Reload" and then press the Play button at the top of the Unity Editor to begin converting the point cloud; this can take several minutes to complete for dense or large point clouds.

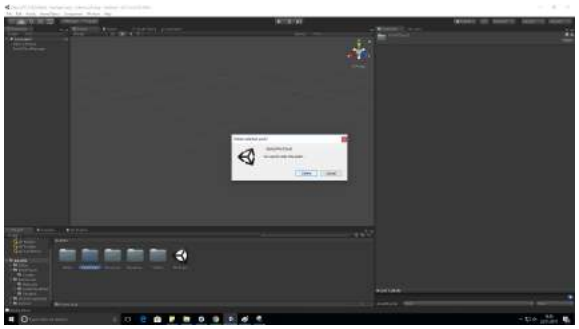
Once the conversion is complete, press the Play button again to leave Play Mode. Next, the plugin must be removed from the project as the project will fail to build with it included. However, first the "Materials" and "Shaders" folders that can be found inside the PointCloud folder must be moved to the "Resources" folder as their

[3] The *.OFF file must be dragged and dropped into the project rather than imported from within Unity as it does not recognise this type of file format when browsing assets.

contents is required to add colour to the point cloud Prefab. To do this, click and drag both folders in the Project window to the Resources folder in the left-hand panel.

Img. 03.07. The new layout of the Unity project's asset folders.

Finally, remove the PointCloud folder from the project by clicking on the "Assets" folder then right-clicking on the PointCloud folder and selecting the option to "Delete" the folder. A pop-up window will appear to prompt you to confirm the deletion, select "Delete" again to remove the plugin from the project.



Img. 03.08. Removing the Point Cloud Free Viewer plugin from the Unity project.

04. Importing Meshes (iOS/ Android)

As an alternative to using point cloud files in the AR app, meshes can be used; this can provide several performance benefits (see Section 09). Meshes can be in any of the following formats: *.FBX, *.dae, *.3DS, *.dxf, or *.obj, and should be less than 200MB in size. Meshes with approximately 600,000 faces offer good performance when used in this application, however meshes with fewer faces will improve the performance considerably on weaker devices.

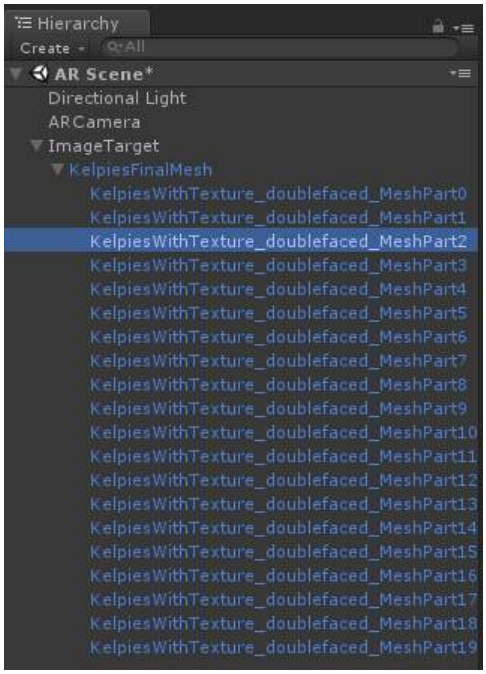
04.01. How to Import a Mesh

To import a mesh into the Unity project, right-click inside the Assets folder in the Project window and select "Create" > "Folder" to add a new folder to the project. Call the folder "Meshes" and double-click it to open the folder; this is where the mesh will be stored. Right-click inside the Meshes folder and select the option to "Import New Asset...", then locate the mesh you want to use. After the mesh has finished importing, create a new folder called "Textures" within the Meshes folder, open it and then repeat the import process for any textures that belong to the mesh.

Apply the texture to the mesh by going into the Meshes folder, and then opening the "Materials" folder that was created upon importing in the mesh. Any materials which are related to the mesh will be stored inside this folder; click on a material to bring up its properties in the Inspector. Within these properties, click on the small circle next to the Albedo property of the material and select the texture that needs to be applied to the mesh.

Img. 04.01. Applying a texture to the meshes material.

If there are no materials belonging to the mesh, a new one can be created by right-clicking in the Meshes folder then selecting "Create" > "Material"; give the material a name and then apply the texture using the aforementioned method. Apply the material to the mesh by expanding the mesh GameObject in the Hierarchy window by clicking on the arrow next to it which will reveal the child GameObjects which make up the overall mesh. Drag the new material onto each of the child meshes to assign it to the GameObject.



Img. 04.02. The mesh GameObject expanded in the Hierarchy to reveal each individual mesh object; drag the material onto the name of each of these individual mesh objects.

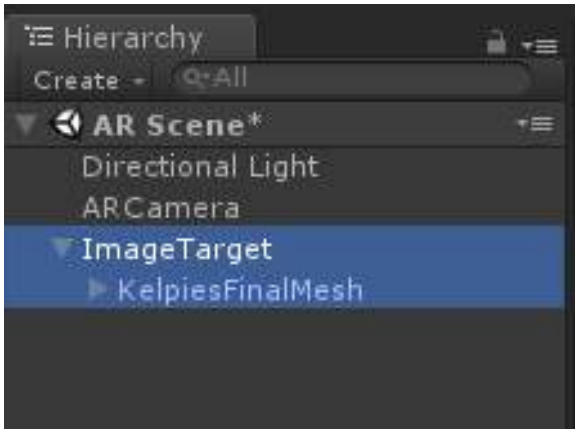
05. Creating an AR Scene

Open the AR Scene which was saved at the start of the project; this should be located in the Assets folder and can be opened by double-clicking on the asset with your Scene name assigned to it. If the Point Cloud Free Viewer plugin has been used, you will be prompted to save the changes which were made to the Example Scene, select the "Don't Save" option to continue on to the AR Scene.

05.01. Adding the AR Components

In the Hierarchy window there should be three GameObjects listed, these are "Main Camera", "Directional Light" and "ARCamera"; this augmented reality application only requires one camera in the Scene, so remove the Main Camera from the Hierarchy. This can be achieved by right-clicking on the Main Camera GameObject in the Hierarchy and selecting the "Delete" option. Next, add an ImageTarget into the Scene via "Hierarchy" > "Create" > "Vuforia" > "Image". This is the GameObject which will trigger the AR experience once the app is running on the device.

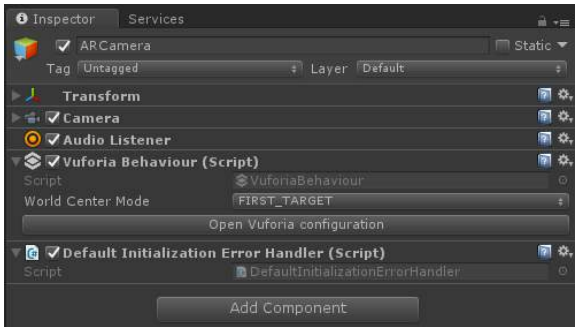
Drag and drop the mesh or point cloud asset that you are using onto the "ImageTarget" GameObject in the project Hierarchy to set it as a child of the ImageTarget; anything that is a child of the ImageTarget GameObject will be hidden in the final app until the target is detected by the camera.



Img. 05.01. The project Hierarchy after adding the mesh or point cloud as a child of the ImageTarget GameObject.

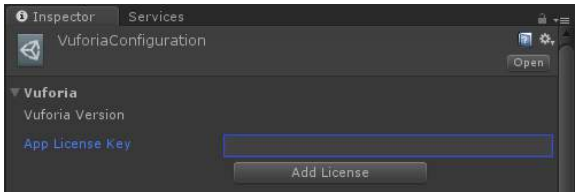
05.02. Creating a Target Database

The next stage is to create a database containing the target that will be recognised by the camera in the AR application and to add a free license key to the ARCamera. To add the license key, click on the ARCamera GameObject in the Hierarchy and, in the Inspector window, click on "Open Vuforia Configuration" which can be found on the "Vuforia Behaviour (Script)" component.



Img. 05.02. Opening the Vuforia Configuration to add a license key.

The Configuration settings will open in the Inspector window and there will be a section labelled "App License Key" which is where the license key needs to be pasted. To get a free developer license for the app, click the "Add License" button under the App License Key box to get to the Vuforia Developer Portal website.

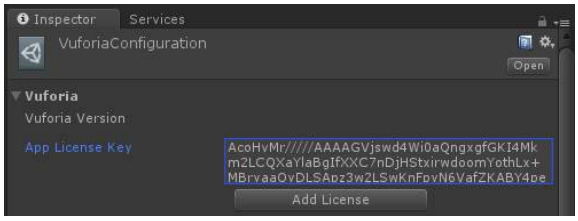


Img. 05.03. The Vuforia App License settings.

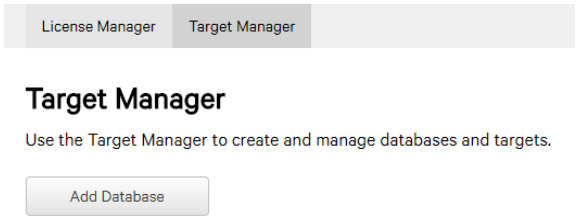
You will be prompted to log in to a Vuforia account in order to manage your license keys; either log in or create a new account and after being redirected to the License

Manager page, click the option to "Get Development Key". Next, enter the name of your app in the "App Name" input field, check the box to agree to the Vuforia Developer Agreement and click confirm to generate a license key.

Once the site directs you back to the License Manager, your app will be added to the list of licenses; click on the name of your app to view the license key. Copy the key and then paste it into the "App License Key" field in the Unity Editor.



Img. 05.04. Adding the license key to the ARCamera in the Unity project.

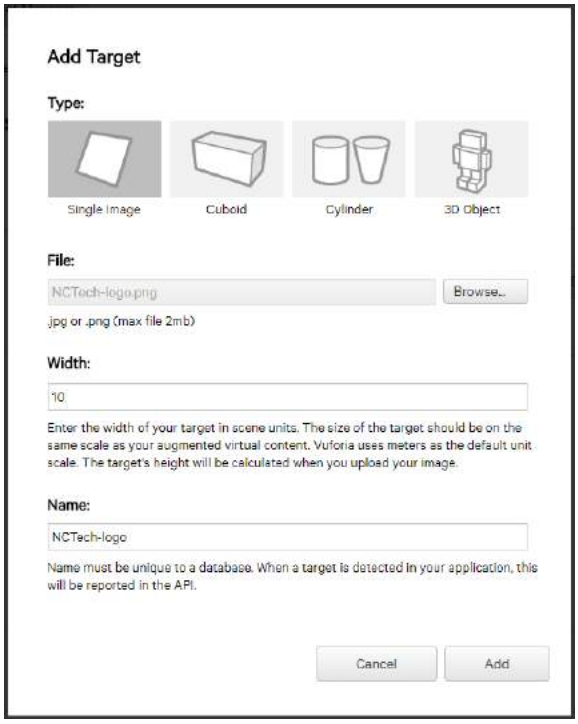


Img. 05.05. Creating a new Database using Vuforia's Target Manager.

Return to the Vuforia Developer Portal where the License Manager can be found, and click on the tab called "Target Manager" to begin creating a Database. In the Target Manager click on the "Add Database" button to open the "Create Database" window. Give a name to the new Database in the "Name" field, tick the "Device" option under the Database "Type" to ensure that the created Database is stored on the mobile device, then click "Create" to make the Database.

05.03. Adding Targets to the Database

Once the Database has been created, targets can be added to it which will be recognised by the app. To add a new target, click on the name of the recently created Database to open it, then click the "Add Target" button which will create a pop-up window prompting you to enter the details of the new target. Select the target type, for the app created in this Application Note the target should be "Single Image", set the "Width" of the target to "10", then upload an image to be the target by clicking the "Browse" button and selecting the image file. Finally, press the "Add" button to add this target to the Database.



Img. 05.06. Adding a new target to the Database.

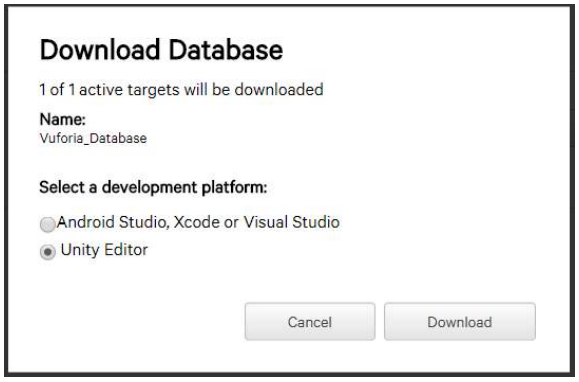
Any images that act as AR targets need to have lots of "features". These are several points in the image which the ARCamera will try to locate in order to determine if a known target is in view. An image with a lot of features will contain several "complex" shapes which are easily identifiable. Once a target has been added to the Database it is possible to find out if it contains enough feature points to be a "good" target image by returning to your Database's overview page and looking at the "rating" that has been assigned to it. The star rating corresponds to how recognisable a target is, so a target with 5 stars will be easily recognised in the app whereas a target with only 1 or 2 stars will be very difficult for the ARCamera to find and track.

Img. 05.07. The Database with an Image Target added to it; notice the star rating of 5 which means the image has a lot of feature points, so can be easily recognised by the ARCamera.

05.04. Downloading the Database

After a suitable image has been added to the Database as a target, the Database can be downloaded and imported into the Unity project so that it can be incorporated into the app. To download it, while on the Database's page, click the option to "Download Database (All)" in the top

right which will open a pop-up window. Ensure that the development platform is set to "Unity Editor" in this pop-up window and then click "Download" to download the Database as a Unity Package File.

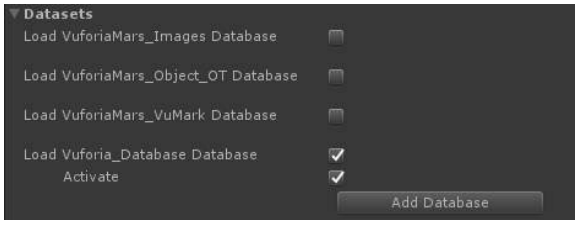


Img. 05.08. Downloading the Database of targets.

05.05. Including the Database in the Unity Project

Return to the Unity project and right-click inside the Assets folder in the Project window; select the option to "Import Package" > "Custom Package..." and import the Database that was previously downloaded. This will bring up another import window, ensure that all of the assets are ticked and click "Import" to start importing the Database into the project.

Next, the ARCamera and ImageTarget GameObject need to be configured to use the new Database. Begin by clicking on the ARCamera object in the project Hierarchy and then clicking on "Open Vuforia Configuration" in the Inspector to return to the window where the license key was previously added. Under the "Datasets" option, untick the pre-existing Databases and then tick the checkbox relating to the Database that was just imported; this will reveal a new checkbox to "Activate" the Database, ensure that this is also ticked. The Database is now loaded into the project

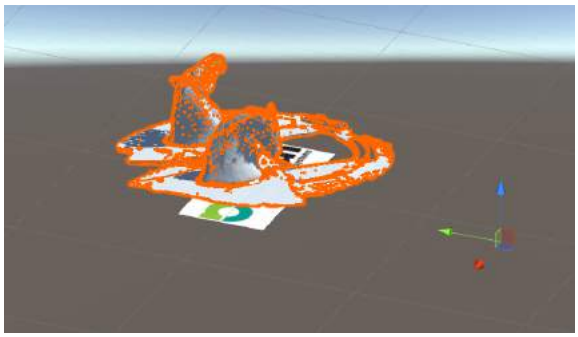


Img. 05.09. Activating the Database within the Unity project.

To configure the ImageTarget GameObject, click on it in the project Hierarchy and then look at the Inspector window; this GameObject will have an "Image Target Behaviour (Script)" Component attached to it. In this Component, ensure that the "Type" is set to "Predefined" and then select your recently made Database and target in the "Database" and "Image Target" drop-down fields respectively.

Img. 05.10. Configuring the ImageTarget to the correct image.

After setting the ImageTarget, the mesh or point cloud may need to be rescaled and repositioned to fit the size of the ImageTarget. This can be done by selecting the GameObject relating to your mesh or point cloud in the project Hierarchy and using the transform arrows to move the GameObject to a position on top of the ImageTarget. Raise your object slightly in the y-axis so that it sits just above the ImageTarget.



Img. 05.11. Use the transform arrows (the green, red, and blue arrows seen above) to position the mesh/point cloud on top of the ImageTarget.

06. Building the Project for Android

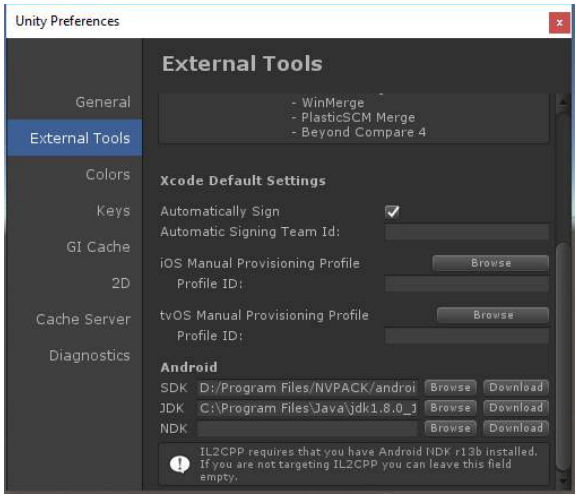
In order to build and use the app on Android, the device will need to have Developer Options enabled. To enable Developer Options, while on the device locate the build number in Settings > About Phone > Build Number[4] and tap the number seven times. Once Developer Options are enabled, turn on "USB Debugging" mode to allow the AR application to be transferred to the Android device for testing. Ensure that the Build Platform of the Unity project has been set to "Android" in "File" > "Build Settings"; for more information on how to set the Build Platform see Section 02.01.

06.01. Adding the Android SDK to Unity

If you are using Unity to make an Android app for the first time, the Android SDK must be downloaded and installed

[4] This may be different for other devices, check with the device manufacturer for more details on finding the build number.

on the development PC, see Section 01 for a link to the SDK. Once Android Studio and Android SDK tools have been installed, return to the Unity Editor and select "Edit" > "Preferences" > "External Tools". Here you will find a section titled "Android" with an option to browse for the SDK that was just installed; click the "Browse" button and then locate the Android SDK folder. Click the "Select Folder" button once the folder is highlighted to confirm the SDK location. Once this is done, close the window by clicking the red cross in the top left.

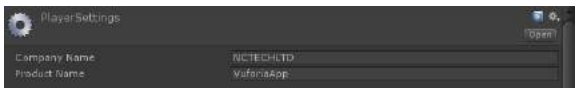


Img. 06.01. Locating the Android SDK tools in Unity.

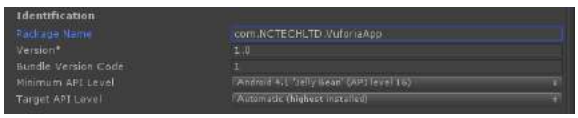
06.02. Setting the Build Configuration - Android

Next, the Player Settings need to be configured to support building to Android devices; to begin the configuration, select "Edit" > "Project Settings" > "Player" to open up the Player Settings panel in the Inspector.

At the top of the Player Settings panel, change the "Company Name" field to the name of the development company, and the "Product Name" field to what you want your app to be called, then scroll down to the Other Settings section. Under the Identification section, change the "Package Name" to a variation of "com.*CompanyName*.*ProductName*", where *ProductName* and *CompanyName* are the exact names which were just set at the top of the Player Settings, see image 06.02 and 06.03. for more detail.



Img. 06.02. Changing the Company and Product Names.



Img. 06.03. Setting the Package Name; ensure that the Package Name uses the exact Company and Product Names.

06.03. Building and Running the App - Android

Finally, to test the app on an Android device, plug the device into the development PC ensuring that Developer Options are enabled on it and then go to "File" > "Build & Run" in the Unity Editor. This will create a window which prompts you to name the build file, give it a name and then select "Save" to begin building the project; once this is complete the app will automatically launch on the Android device. It will remain on the device until it is uninstalled by the user using the same method as you would for other Android apps.

07. Building the Project for iOS

To begin building the project for an iOS device, an Apple Developer Account is required; this is available for free to develop and test applications on an iOS device without releasing them on the App Store. This account will give you access to beta versions of software such as Xcode and will also allow you to "sign" your apps; this is a necessity for testing the app as unsigned apps cannot be run on iOS devices. Also, ensure that the Build Platform of the Unity project has been set to "iOS" in "File" > "Build Settings"; for more information on how to set the Build Platform see Section 02.01.

07.01. Creating an Apple Developer Account

To create a Developer Account, go to https://developer.apple.com/ and select the "Account" option at the top of the website. This will prompt you to either sign-in or create a new Apple ID; if you do not currently have an Apple ID then click on the option to create one and fill out your details in the signup form, after completing the creation process you will be redirected back to the sign-in page.

Using your Apple ID, sign in to the Developer Portal; this will take you to the Apple Developer Agreement terms which should be read and then agreed to in order to gain access to a Developer Account. After accepting the terms and conditions, the setup for creating an Apple Developer Account is completed, giving you access to downloading Xcode and testing apps on iOS devices.

07.02. Setting the Build Configuration - iOS

Next, the Player Settings in the Unity project must be configured in order to build the application; to open the Player Settings, go to "Edit" > "Project Settings" > "Player" and the Player Settings will open in the Inspector window. At the top of the Player Settings, enter the name of the development company in the "Company Name" field as well as the name of what the app will be called in the "Product Name" field.

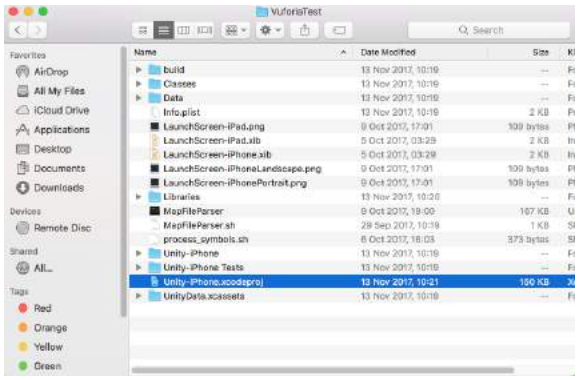
Img. 07.01. Setting the Company and Product Names in the Player Settings.

Under the Identification section of the Player Settings, enter a variation of "com.*CompanyName*.*ProductName*" in the "Bundle Identifier" field, where *CompanyName* and *ProductName* are the exact names which were just entered at the top of the Player Settings. See images 07.01. and 07.02. for more detail. Finally, in the Configuration section of the Player Settings, enter a short description of what the iOS camera will be used for in the "Camera Usage Description" field, for example this could be "Camera used for AR tracking".

Img. 07.02. Changing the Bundle Identifier to match the Company and Product Names of the current project.

07.03. Building and Running the App - iOS

To test the app on an iOS device, the project must first be built in Unity and then signed in Xcode before it is transferred to the iOS device. Build the app within Unity by selecting "File" > "Build Settings" and then clicking on the "Add Open Scenes" button to ensure that the Vuforia AR scene that has been created is the one which will be present in the app. Next, click the "Build" button which will prompt you to choose a location to save the build folder; ensure that a location outside of the project's Asset folder has been chosen, name your build and then click "Save" to begin the build process. This process may take several minutes and once it is completed, a Finder window will open to the newly created build file.

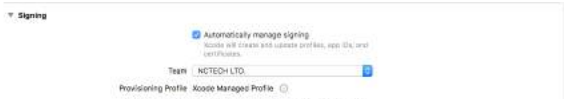


Img. 07.03. Open the build in Xcode by double-clicking on the file with the "xcodeproj" extension - this will be found in a folder with the name of your build.

Inside the folder where the build was saved will be a file with the .xcodeproj extension; double-click on this file to open up the build in Xcode. Next, with the build open in Xcode, plug an iOS device into the development Mac and ensure that it is set as the build device in Xcode; this can be determined by looking in the top left corner of Xcode where "Unity-iPhone" can be seen, see image 07.04. Next to this should be the name of your iOS device, however if it says "Generic iOS Device" or the name of another device, click on the name of the device to open a drop-down menu which will allow you to select the device of your choice.

Img. 07.04. Selecting the desired iOS device to transfer the app to.

The next step is to sign-in to your Apple Developer Account in order to "sign" the app; to do this, go to "General" > "Signing" > "Team". Click on the selection box next to "Team" and then select the option to "Add an Account..." which will prompt you to enter your Apple ID and password; enter your details to add your Developer Account to Xcode. After this, ensure that your Developer Account is set as the "Team" by selecting it in the selection box previously mentioned.



Img. 07.05. Set the Signing Team to your Apple Developer Account.

Finally, press the play button (a triangle pointing right) in the top left of Xcode to begin building the app on the connected iOS device. This will take several minutes to complete and when it is done, the app will automatically launch on the connected iOS device. It will remain on the device until it is uninstalled by the user using the same method as you would for other iOS apps.

08. Testing the App

In order to test the application, print out a copy of the image that was uploaded to the Vuforia Database; this can be in either black and white, or colour and can be any size, however larger targets work best to produce smooth AR experiences. For the best results, place the printed target on a flat surface in a well-lit room then, with the app loaded on the mobile device, point the device in the direction of the image, ensuring that the full target can be seen on the device's screen. Once the target has been recognised, the mesh or point cloud will appear on top of the image. Move your device around the target to get a full 360 view of the object; alternatively, pick the target up and move it around in your hands to view the 3D object that way.

09. Conclusion

Augmented reality apps can be created in Unity which allow you to view a mesh or point cloud using an image to act as the trigger which sustains the AR experience. This method of AR has its benefits and limitations when compared to targetless AR alternatives such as ARKit and ARCore, which are discussed below.

09.01. Benefits and Limitations of Target Based AR

A target based approach to creating an augmented reality experience using the Vuforia SDK can provide the user with a stable AR application that can be created quickly and easily to display 3D content. As this method of AR has existed in the market for several years, the technology itself is more stable than ARKit/ARCore alternatives, and can be implemented on old and new devices alike provided that the device has a camera.

Despite the benefits, however, target based AR does have its limitations, the most prominent of these being that the experience takes place on the target. Having the AR experience take place on an image target limits the user's freedom of movement and largely impacts the quality of the experience itself. The user's experience can be easily interrupted when trying to view the 3D content from various angles as the image target can become obscured from the device's camera view. This is not the case when using targetless AR alternatives such as ARKit and ARCore which allow the user full freedom of movement to view the augmented content.

Another factor is the inconvenience of obtaining and keeping a physical copy of the image to act as the target; when releasing a target based AR app, it should be taken into consideration that not everyone will have access to a means of printing out their image target, and once an image target is obtained, the user must have it on their person if they want to use the app on the go which could be an inconvenience.

09.02. Android Vs iOS

There are differences between the two platforms, iOS and Android which will limit the content of an augmented reality apps as well as other types of mobile application. Due to differences in the rendering capabilities between iOS and Android, it is not possible to render point clouds on iOS devices. This means that for iOS development when wanting to display 3D content, a mesh must be used instead; this adds an extra level of complexity to the workflow if needing to convert a point cloud to a mesh however it does offer greatly improved performance over point cloud usage.

While it is possible to display point clouds on Android devices it is not always an appropriate solution for creating smooth AR experiences on mobile. This is due to the performance costs of rendering the vast quantity of points that are present in a point cloud compared to that of a mesh. Meshes offer a better alternative to point clouds as backface culling can be used to only render faces which are seen by the camera, reducing the performance costs dramatically. As a result of this, it is more practical to use meshes where possible regardless of the platform that the app is being designed for; this will also allow the app to run on less powerful devices which would be unable to render point clouds at all.