AR Rig

Documentation for Version 0.9.

Overview

AR Rig exists to provide Unity developers with a no-hassle augmented reality system. When assigned a Target Camera, AR Rig automatically displays the user's webcam feed and utilizes available sensors to orient and position the camera.

Applications

AR Rig allows developers to create several kinds of augmented reality experiences. First, any objects placed by hand in a Unity scene will appear as though they exist in the real world when deployed to a mobile device. This allows developers to create virtual worlds that appear to co-exist with the user's surrounding environment. Second, by using AR Rig's built-in conversion functions, virtual objects can be placed programmatically at real-world locations. Finally, these two techniques can be combined to create scenarios where some objects always appear at the user's location, and others appear based on real-world data.

Fallback Systems

AR Rig uses a multitiered hardware fallback system to support as many devices as possible.

Tier 1 – Gyroscope (Optional Compass and Location Services)

In this tier, view rotation is set using data from the gyroscope. If Location Based Position is enabled by the developer, and location services and compass data are available on the device, then an initial Unity world angle offset is generated using the compass, and view position is set using location services.

Tier 2 – Accelerometer, Compass (Optional Location Services)

This tier is enabled if the deployment device does not have a gyroscope. In this tier, an initial Unity world angle offset is generated using the compass. View rotation is set using the compass working in tandem with accelerometer data to simulate a gyroscopic view. If Location Based Position is enabled by the developer (see Editor Settings below) and location services are available on the device, then view position is set using location services.

Tier 3 – Gesture View Controls

This tier is enabled if the deployment device does not satisfy the conditions outlined in Tiers 1 and 2. In this tier, view rotation is set using screen gestures – the user may tap-and-hold on the touch screen in order to control the orientation of the camera. Location services are currently unavailable in this tier.

Instructions

To use AR Rig, (1) drag the AR Rig prefab from the Project panel into the Hierarchy panel, (2) click on it to reveal its properties in the Inspector panel, and (3) drag Main Camera from Hierarchy into the Target Camera field in the Inspector. That's it.

Alternatively, this can be accomplished in C# like so:

```
Instantiate(Resources.Load("AR Rig"));
AR_Rig ar_rig = GameObject.FindObjectOfType<AR_Rig>();
ar_rig.TargetCamera = GameObject.FindGameObjectWithTag("MainCamera");
```

Editor Settings

Debug

When checked, AR Rig will output messages into the Unity console, which may be useful for helping developers troubleshoot problems. It will also output select orientation information from available hardware to the screen via GUI, which may be useful for helping troubleshoot device deployments while not connected to an output console.

Target Camera

If a GameObject with a Camera component is assigned to Target Camera, AR Rig turns on, and becomes linked to that GameObject. If the Target Camera becomes `null`, AR Rig turns off.

Invert X Axis

When checked, the webcam texture is flipped horizontally. NOTE: At runtime, Invert X Axis reflects the contents of a key / value pair stored in PlayerPrefs called "AR_InvertXAxis." If this preference exists and is set to 1, Invert X Axis will be checked; otherwise it will be unchecked. It is recommended that you allow end-users to control this setting manually via the End-User Settings Panel.

Invert Y Axis

When checked, the webcam texture is flipped vertically. NOTE: At runtime, Invert Y Axis reflects the contents of a key / value pair stored in PlayerPrefs called "AR_InvertYAxis." If this preference exists and is set to 1, Invert Y Axis will be checked; otherwise it will be unchecked. It is recommended that you allow end-users to control this setting manually via the End-User Settings Panel.

Location Based Position

When checked, AR Rig will attempt to align the initial Target Camera Unity position with an initial user geolocation position (if available). From there, changes in the user's geolocation (if available) will be reflected in changes to the Target Camera's Unity position, simulating user movement relative to Unity GameObjects. NOTE: This behavior is typically not available in the Unity Editor. When checked, this setting also enables Location Based AR Object positioning based on geolocation data.

Location Scale

Location Scale is the multiplier by which distance (in Unity units) based on geolocation data is scaled. This setting impacts both Location Based Position and the placement of Location Based AR Objects. Location Scale accepts values from 0.0f – 100.0f. NOTE: AR Objects placed outside the Target Camera's farClipPlane will not be visible; AR Rig does not automatically cull objects. It is recommended that you manually cull AR Objects based either on raw geolocation data in relation to the user's current location or on the Location Based AR Object's Unity position (see Placing AR Objects below).

Placing AR Objects

When building your augmented reality scene, it's important to remember that *one unit in Unity is equal to one meter in the real world*. Place and scale objects in the Unity coordinate system accordingly. The AR Rig assumes that the Unity Y position axis is vertical and looks in X, Y, and Z directions while moving, based on geolocation data, in X and Z directions (adjusted for compass). The AR Rig Target Camera should be placed at the developer's desired starting Unity position and Y rotation indicating the default view direction (Target Camera X and Z will zero automatically before updating based on gyroscope and compass data or accelerometer and compass data).

Unity Position AR Objects

Objects placed in the Unity scene will move and react in relation to the camera as Unity objects do normally. An object placed in front of the camera will be in front of the camera at runtime (barring compass/geolocation wandering); an object placed behind the camera will be behind the camera at runtime, irrespective of real-world orientation.

Location Based AR Objects

AR Rig comes with several functions to help you convert between real-world geolocation coordinates and Unity coordinates, which are covered below (see also, the sample AR_Object script included in the AR Rig). AR Objects placed via geolocation data will NOT always be oriented in the Unity world as positioned in the editor. They will be moved to the appropriate Unity position based on user location and compass rotation.

Converting Latitude and Longitude to Unity Position

AR Rig can accept latitude / longitude inputs as doubles or as a Vector2 object with latitude in the X position and longitude in the Y position and output Unity world coordinates accounting for a real-world compass offset. To place an object in the world it is recommended you attach a script to the object similar to or derived from the AR_Object MonoBehavior sample, store a reference to the AR Rig, and pass geolocation coordinates to the GetUnityPosition() function as demonstrated below. NOTE: If location services are unavailable or not permitted by the user, GetUnityPosition() will return Vector2.zero. For this reason, it is important to check the returned Vector2 object against Vector2.zero.

```
AR_Rig ar_rig = GameObject.FindObjectOfType<AR_Rig>();
Vector2 position = ar_rig.GetUnityPosition(latitude, longitude);
if (position != Vector2.zero)
   transform.position = new Vector3(position.x, height, position.y);
```

Converting Unity Position to Latitude and Longitude

AR Rig can also accept Unity X and Z coordinates as doubles or as a Vector2 object with X in the X position and Z in the Y position and convert them back to geolocation data accounting for the real-world compass offset. To get a real-world geolocation latitude / longitude set from Unity coordinates, it is

recommended that you pass the desired Unity position to the AR Rig GetLatLonPosition() function as demonstrated below. NOTE: If location services are unavailable or not permitted by the user, GetLatLonPosition() will return Vector2.zero. For this reason, it is important to check the returned Vector2 object against Vector2.zero.

```
AR_Rig ar_rig = GameObject.FindObjectOfType<AR_Rig>();
Vactor2 unity_position = new Vector2(transform.position.x,
    transform.position.z);
Vector2 geo_position = ar_rig.GetLatLonPosition(unity_position);
```

End-User Settings Panel

AR Rig comes with a prebuilt settings panel that allows end-users to manipulate the presentation of the device camera. This is important because different device cameras have different orientations. Future versions of AR Rig may handle default orientations for more popular devices; however, allowing end-users the ability to correct orientation is good practice! To show the panel, use the following code:

```
AR_Rig ar_rig = GameObject.FindObjectOfType<AR_Rig>();
if (ar_rig != null) ar_rig.ShowSettingsPanel();
```

The End-User Settings Panel will automatically save the user's preferences.

Development Roadmap

AR Rig is an ongoing development project. Below are some things we're working on, as well as expected delivery dates.

Item	ETA
Support orthographic camera.	Fall 2016
Determine default axis settings for popular devices.	Fall 2016
Support device orientations other than landscape left.	Fall 2016

Recommendations

Below are recommendations to make sure AR Rig runs as smoothly as possible for your users:

- Remember to build your augmented reality scene so that one unit in Unity is equal to one meter in the real world. Should you require a different scale, you can change the value of AR Rig's Location Scale property in the Inspector panel prior to deployment.

- Although it will attempt to compensate for changes in the display, AR Rig works best in and currently only supports projects that are locked to landscape left mode. To do this, click on the File menu in Unity, choose Build Settings, and click the Player Settings button in the window that appears. In the Inspector panel, you should find a Default Orientation property under the Android and iOS tabs. Make sure this property is set to Landscape Left.

- Provide a way for users to access the End-User Settings Panel, so that they can make modifications if necessary. You may use the following code to display the panel:

```
AR_Rig ar_rig = GameObject.FindObjectOfType<AR_Rig>();
if (ar_rig != null) ar_rig.ShowSettingsPanel();
```

- Currently, AR Rig only supports in-game cameras that use a perspective projection. If the target camera is set to orthographic, AR Rig will convert it to perspective. In the future, this may not be the case, so be sure your target camera is perspective!

- It is not recommended you remove the "Unlit-Normal.shader" file (the "Unlit/Texture" shader) from the Resources folder as this automatically forces Unity to include it in the project deployment. You may remove the file and achieve the same result of forced inclusion by going to "Edit -> Project Settings -> Graphics" and adding the "Unlit/Texture" shader to the list of Always Included Shaders. This shader is the only shader currently supported by AR Rig but that could be expanded upon request in future versions. NOTE: If the "Unlit/Texture" shader is forcibly included in the project the project will still appear to function as expected in the editor but not in deployment.

- Location Based Position motion works best directly under a clear sky.

Troubleshooting

Having problems with AR Rig? First, make sure the Debug box is checked in the Inspector, then monitor the Console for messages prefaced with `*** AR RIG`, which may provide insight into what's going wrong. You can use the section on Debug Codes below to help understand what's happening internally with AR Rig. Second, visit www.badpilcrow.com/unitydev for the latest documentation. Finally, if you're still stumped, send an e-mail to unitydev@badpilcrow.com and we'll do our best to help you out.

Debug Codes

ARDC001 – Not authorized to use a webcam. Check permissions!

If `Application.HasUserAuthorization(UserAuthorization.WebCam)` returns false when attempting to access a webcam, AR Rig will report this message. AR Rig will automatically generate a request for permission in this scenario, and if that request is granted, another attempt will be made to access the webcam. You may seek permission from the user to access a webcam *before* loading AR Rig, using the following code:

```
Application.RequestUserAuthorization(UserAuthorization.WebCam);
```

ARDC002 – Falling back to [NAME], a front-facing webcam.

If AR Rig cannot find a rear-facing camera – one that points *away* from the user – it will instead resort to using the first front-facing webcam it can find. If such a camera is found, it reports this message.

ARDC003 – Couldn't locate a webcam.

This debug code appears if AR Rig failed to find a webcam attached to the device. At this point, AR Rig has searched for a rear-facing camera AND attempted to find a front-facing camera, but located neither. It's possible that the device does not have a webcam, or the webcam is not working properly.

ARDC004 – Selecting [NAME], a rear-facing webcam.

AR Rig reports this debug code when it locates a rear-facing webcam. In this case, *everything is working as intended*; however, the name of the webcam is reported to differentiate between multiple rear-facing webcams, in the event that more than one exists.

ARDC005 – Orthographic cameras are not currently supported.

Future versions of AR Rig will support orthographic camera projections; however, only perspective cameras are compatible at the moment. This debug code is presented if the target camera assigned to AR Rig is orthographic. In this case, AR Rig will convert the target camera to perspective.

ARDC006 – System supports gyroscope: [TRUE | FALSE]

AR Rig displays this debug code when attempting to utilize a gyroscope. It will report `true` when a gyroscope is available, and `false` when no gyroscope is found.

ARDC007 – Location services disabled by user.

If the user has not enabled location services on their device, AR Rig will report this debug code. The user may also be running their device in airplane mode or a battery saving state. It may be a good idea to prompt the user to enable their location services. You can check for availability by testing the value of the following read-only variable:

```
Input.location.isEnabledByUser
```

ARDC008 – TargetCamera does not have a camera component!

If AR Rig reports this debug code, then the object assigned to its TargetCamera field does not possess a Camera component. In this case, AR Rig will fail to present a feed from the user's webcam, because it is not possible to position, orient, and scale the billboard upon which the feed is rendered. To prevent this situation from happening, make sure that AR Rig's TargetCamera contains a Camera component.

ARDC009 – AR Rig only supports landscape left.

Future versions of AR Rig are expected to support multiple device orientations; however, this debug code will appear if the app is running in any mode other than landscape left.