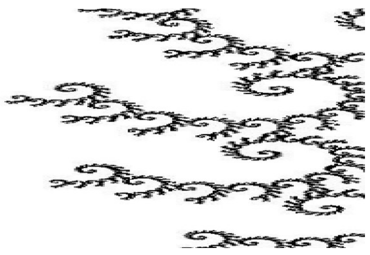
UNITY3D TOOLS AND SCRIPTS



ARS Pack #1 Compass3D + OSC + Rebase

Asset Store URL

NOTICE: The OSC library uses a 3rdparty library called SharpOSC. The library license is MIT. The license text file is in the 'Plugins' folder.

This asset is a pack with three parts.

One part to handle mobile device accelerometer and compass to compute 3D orientation. Can be used in games, tools or 360 players.

Other part to handle OSC messages in a local network. Can be used in digital interactive software.

And another part to rebase 3D coordinates that can be used with VR coordinates.

The parts:

a) The Compass3D
b) The OSC send/receiver
c) The VR 3D coordinate rebase

a) The Compass3D

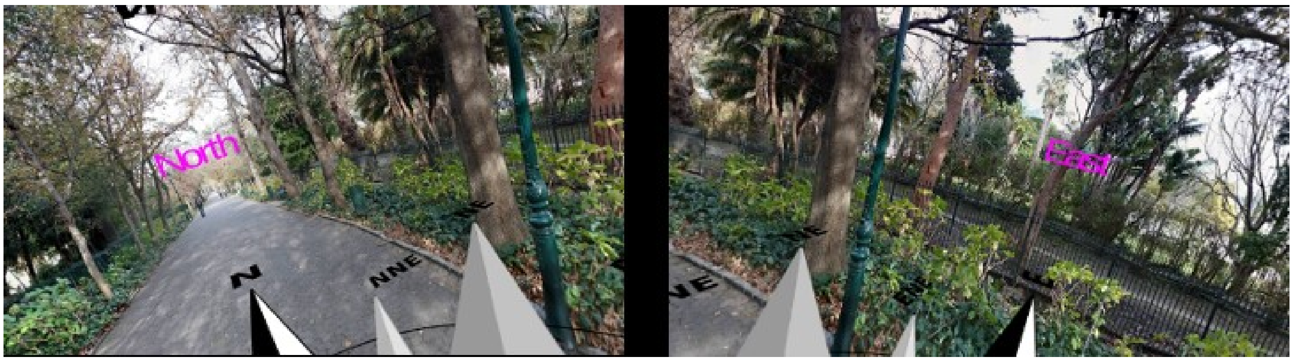
The Compass3D class computes the virtual world compass orientation based on the accelerometer and magnetometer inside the mobile device.

Can be used as a template to create compass app or a 360 viewer.

NOTICE: This implementation is not a high precision implementation. It does not use the gyroscope to compute YAxis rotation.

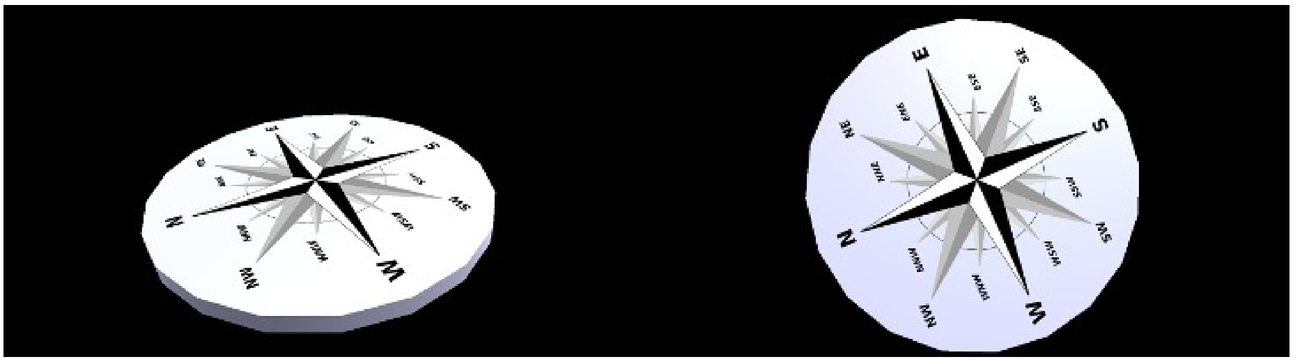
360 Viewer example (camera rotation):

```
void Update()
{
   transform.rotation = Compass3D.CameraRotation;
   if (Input.GetMouseButtonDown(0))
      Compass3D.ResetNorth();
}
```



Compass3D example (object orientation):

```
void Update()
{
    transform.rotation = Compass3D.ObjectRotation;
}
```



Android tech demo:

[Android Earth Defender 360]

You can see the example scene:

- Demo/Demo Compass3D/Scenes/test360CameraOrientation
- Demo/Demo Compass3D/Scenes/testCompassObjectOrientation

b) OSC send/receiver

The OSC send/receiver are built to use C# Reflection to make it easy to use.

The tool work with bundles also (bundles are groups of messages).

Send Example:

```
OSC.OSCSender sender;
void Start () {
   sender = OSC.aquireSender("localhost", 5000);
}
void OnDestroy() {
   sender.close();
}

//inside the code
sender.send("/singleFloat",35.0f); // sends 1 float
                                   // to /singleFloat
sender.send("/complexType",35.0f, 24, "test");
                       // sends 1
                       // complex message
                       // composed by a float,
                       // an int and a string.

//work with bundles
OSC.OSCBundle bundle = new OSC.OSCBundle(System.DateTime.Now);
bundle.addMessage("/bundleIntSeq", 1, 2, 3, 4);
bundle.addMessage("/bundleFloatSeq", 1.0f, 2.0f, 3.0f, 4.0f);

sender.send(bundle);
```

Receive Example:

```
OSC.OSCListener listener;
void Start () {
   listener = OSC.aquireListener(this, 5000);

   listener.bind("/singleFloat", "singleFloat");
   listener.bind("/bundleIntSeq", "bundleIntSeq" );
   listener.bind("/bundleFloatSeq", "bundleFloatSeq" );
}
void OnDestroy() {
   listener.close();
}

public void singleFloat(float f)
{
   Debug.Log("Received singleFloat: " + f.ToString());
}
```

```
    public void bundleIntSeq(int v1, int v2, int v3, int v4)
    {
        //when use bundles, can consult the timetag
        if (listener.ComesFromBundle)
        {
            Debug.Log("Received bundleIntSeq " +
                        listener.BundleTimeTag.Timestamp);
        }
    }

    public void bundleFloatSeq(float v1, float v2,
                                float v3, float v4)
    {

        //when use bundles, can consult the timetag
        if (listener.ComesFromBundle)
        {
            Debug.Log("Received bundleFloatSeq " +
                        listener.BundleTimeTag.Timestamp);
        }
    }
```

You can see the example scene:

- Demo/Demo OSC/Scenes/testOSCSendReceiveTypes
- Demo/Demo OSC/Scenes/testOSC_Send
- Demo/Demo OSC/Scenes/testOSC_Receive

c) 3D coordinate rebase

The coordinate rebase can be used to map a real space to a 3D virtual space.

It maps a non-orthogonal basis to another non-orthogonal basis.

For example:

If you have a square inside the Unity that measures 4,4,4 (width, height, depth) and your real space measured with the HTC Vive, for example, is 2.2,2.4,2.3 .

Your pointer in the 3D space will not cover the entire room.

You can use the 3D space rebase to fix the relationship between the spaces.

First you need to set the Unity space you want to cover.

```
rebase.setTargetOrigin(target);
rebase.setTargetX(tx);
rebase.setTargetY(ty);
rebase.setTargetZ(tz);
```

Second you need to fill the VR real distances with the correct reference

```
rebase.setOrigin(origin);
rebase.setX(x);
rebase.setY(y);
rebase.setZ(z);
```

Compute the calibration matrix

```
rebase.computeCalibrationMatrix();
```

And finally, you need to feed the Rebase component to get the virtual space mapped point.

```
transform.position = rebase.rebase(VRPosition.position);
```

You can see the example scene:

- Demo/Demo Rebase/Scenes/testRebase
- Demo/Demo OSC+Rabase/Scenes/testRebaseOSCReceiver

CONTACT INFORMATION

contact@unity.thegeneralsolution.com