**Visualising Normals**

**Getting Started**

Attached to this article as a resource is a Unity package that will allow you to explore how vectors, in particular normals, change as the x, y and z values are manipulated.  Download the package and add to a new Unity project and open the *VisNormal*s scene.

**Introduction**

In the scene you will find a standard Unity sphere with a bunch of red lines pointing out from it.  These are the vertex normals inside the mesh visualised as debug draw lines (that's why you can only see them in the Scene window).

A screenshot of a video game

AI-generated content may be incorrect.

To begin all these normals face out of the vertices at 90 degrees (which is the way a default normal will be).

***Note: The normals shown here aren't necessarily what they would be after the shader has processed them as these are in world space and the processed ones are in screen space.  However, the mathematics is exactly the same.  Just assume here that the view is always looking along the Z axis (that's the axis going in and out of the screen in the screen/clip space).***

To manipulate these normals, look to the Inspector while the sphere is selected.  You'll find a ShowNormals script attached.  Here you can manipulate the x, y and z components of the normal.  These values are multiplied with the corresponding normal coordinates.

A screenshot of a computer

AI-generated content may be incorrect.

**Normals and Brightness**

One thing normals are used for, as you'll have found in the previous exercises, is determining brightness.  This brightness in the Lambert lighting model is determined by the **angle** between the **normal** and the **light source**.  The more a normal faces the viewer the stronger the brightness at that point on the surface.

If you change the Nz value to 2 (as shown below) and leave the others at one, you can see the normals lean over in the Z direction.  If this were in a shader it would make the side of the sphere facing the viewer brighter (with the side facing away much darker).

A screenshot of a computer

AI-generated content may be incorrect.

If you change the Nx, Ny and Nz values to be different to each other you are actually modifying the direction of the normal and this will impact on the brightness of the associated vertex.

For example, if you manipulate the Nx, Ny and Nz such that the normals are pointing away from the viewer then you'll make the side facing the viewer a lot darker (see below).

A screenshot of a computer

AI-generated content may be incorrect.

Normals, usually have a length of 1 when used in many mathematical computations, but as you can see when you change the coordinates, it will affect the length.  If you change the Nz value you will undoubtedly change the length as the normal starts to lean over.  This will change the angle. The only time you won't change the angle is when Nx, Ny and Nz are changed equally and in this case the brightness doesn't change because the angle hasn't changed (see below).

A screenshot of a computer

AI-generated content may be incorrect.

**Seeing this on a Shader**

To see how manipulating the normal in real time affects the brightness you can apply this shader to the sphere:

1. Shader "Holistic/NormalPlay"
2. {
3. Properties
4. {
5. \_myX ("Nx", Range(-10, 10)) = 1
6. \_myY ("Ny", Range(-10, 10)) = 1
7. \_myZ ("Nz", Range(-10, 10)) = 1
8. }
9. SubShader
10. {
12. CGPROGRAM
13. #pragma surface surf Lambert
15. half \_myX;
16. half \_myY;
17. half \_myZ;
18. struct Input
19. {
20. float2 uv\_myDiffuse;
21. };
23. void surf(Input IN, inout SurfaceOutput o)
24. {
25. o.Albedo = 1;
26. o.Normal = float3(\_myX, \_myY, \_myZ);
27. }
29. ENDCG
30. }
31. Fallback "Diffuse"
32. }

***Note 1: Changing Nx, Ny and Nz in the shader DOES NOT change the normals being shown in the Scene.  These normals are in world space and belong to the mesh BEFORE the shader is applied.***

***Note 2:  Remember when using a shader, YOU become the viewer and the Z axis is coming out of the computer screen at YOU.  Not the viewer game object shown in the scene.***

If you manipulate the Nz of the shader to the extreme you'll see the side of the sphere get brighter.  This is because you have bent the normals around to be facing more toward and away from you.

A screenshot of a computer

AI-generated content may be incorrect.

If you make Nz negative, the normals will start to turn away from you and you'll see the brightness diminishing.

If you aren't always sure how a shader is manipulating the involved vectors, don't be afraid to break the code down into basics, like is done here, and experiment!

Resources for this lecture

* VisNormals2.zip