Introduction

Aujourd'hui est un grand jour !! Nous allons créer la lumière !!

Les normals

Reprenez le terrain et rajoutez les normales au Vertex Buffer (Pensez à aussi modifier la vertex déclaration). Pour rappel une normale est un float3.

Modifiez le shader pour prendre en compte la normal (modification des structures d'entrée et de sortie du Vertex Shader et donc aussi l'entrée du Pixel Shader)

Voila du code qui vous permet de créer les normales du terrain. En revanche à vous de l'adapter à votre code et à la class DirectX de Vecteur.

```
unsigned int i = 0;
for (unsigned short z = 0; z < m_sizeZ; ++z)
{
	for (unsigned short x = 0; x < m_sizeX; ++x, ++i)
	{
		const cVector3& pos = vertexBuffer[i].position;
		vertexBuffer[i].normal.x = 0.0f;
		vertexBuffer[i].normal.y = 0.0f;
		vertexBuffer[i].normal.z = 0.0f;
		cVector3 normal;
		float diffHeight;
		if (z > 0)
		{
			if (z + 1 < m_sizeZ)
				diffHeight = m_height[i - m_sizeX] - m_height[i + m_sizeX];
			else
				diffHeight = m_height[i - m_sizeX] - m_height[i];
		}
		else
			diffHeight = m_height[i] - m_height[i + m_sizeX];
		vertexBuffer[i].normal += cVector3(0.0f, 1.0f, diffHeight).Normalize ();
		if (x > 0)
		{
			if (x + 1 < m_sizeX)
				diffHeight = m_height[i - 1] - m_height[i + 1];
			else
				diffHeight = m_height[i - 1] - m_height[i];
		}
		else
			diffHeight = m_height[i] - m_height[i + 1];
		vertexBuffer[i].normal += cVector3(diffHeight, 1.0f, 0.0f).Normalize ();
		vertexBuffer[i].normal.Normalize ();
```

```
        }
}
```

Afficher la normal comme couleur (à la place de la texture).
Vous ne voyez pas comment faire ? La normal est un float3 et une couleur est un float4 il suffit donc juste de faire un return float4(MaNormal, 1.0f);
En réalité ceci ne suffirait pas car une normal est uniquement un vecteur normalisé et peut donc avoir des composantes négatif. Pour pallier à ceci on pourrait transformer de [-1.0f, 1.0f] à [0.0f, 1.0f].
Pour cela rien de plus simple il suffit de faire MaValeur * 0.5 + 0.5f

Ajout d'une directionnelle

Rajoutez en variable global dans le shader la direction et la couleur de la lumière.

Pour la couleur le type de donnée est float3 ou float4 avec le w à 1.0f.

Pour les utiliser il faut faire comme pour la matrice de WorldViewProj, c'est à dire récupérer le Handle du côté C++ pour pouvoir lui donner une valeur.

Rajoutez le calcul de l'éclairage de la directionnelle dans le pixel shader.

Pour la diffuse il s'agit d'un simple Dot product (via l'Intrinsic dot(vect1, vect2)).

Pour le specular il vous faudra d'abord rajouter la position de la camera. Ensuite appliquer les formules des slides pour pouvoir faire le produit scalaire.
Petit rapel:
Calcule du vecteur de réflexion de la lumière. Puis un dot avec le vecteur de direction de la camera pour le pixel en cours(au vertex shader faites la différence entre la position world et la position world de la camera).
Et ensuite mettre à une puissance(intrinsic pow).

Penser à gérer les cas où les coefficients sont négatif.

Une omni !!

Rajoutez en variable global au shader la position d'une omni ainsi que ça couleur et la distance maximum à laquelle elle éclaire.

Rajoutez le calcul de l'éclairage de l'omnidirectionnelle au Pixel Shader.

Pour l'éclairage diffuse il s'agit de multiplié le dot product par le rapport entre la distance à la lumière et la distance max. A vous de trouver comment avoir la direction de la lumière.

Les lumières s'accumule/s'additionne.