Basic Surface Shader

```
Shader "MyShaders/BasicSurfaceShader_Opaque" {
    Properties
    {
        // Texture Properties
        _MainTex ("Texture", 2D) = "white" {}

        // Metallic PBR Lighting Properties
        _Albedo ("Albedo", Color) = (0,0,0,1)
        [HDR] _Emission ("Emission", Color) = (0,0,0,1)
        _Metallic ("Metallic", Range(0,1)) = 0
        _Smoothness ("Smoothness", Range(0,1)) = 0
    }

    SubShader
    {
        // These tags will be important later for differentiating between alpha and opaque objects.
        Tags
        {
            "RenderType" = "Opaque"
            "Queue" = "Geometry"
        }


        CGPROGRAM
        /* This declaration defines different aspects about the surface function.
        In this case, we are specifying a Standard PBR shader with shadows rendered
        on forward rendering path.*/
        #pragma surface surf Standard fullforwardshadows

        /*This allow usage of DirectX 9 Shader Model 3.0, meaning better quality lighting.*/
        #pragma target 3.0

        // Properties
        sampler2D _MainTex;
        fixed4 _Albedo;
        half3 _Emission;
        half _Metallic;
        half _Smoothness;


        /*
        ============================================================================================================
        INPUT STRUCT
        ============================================================================================================
        */

        // This struct defines the input data of the shader.
        struct Input
        {
            /* Note the naming convention here.  We add a PREFIX 'uv' next to the texture name _MainTex.
            This allows for the editor/outside code to affect the UV scale and tiling/offset values. */
            float2 uv_MainTex;
        };

        /*
        ============================================================================================================
        INPUT STRUCT
        ============================================================================================================
        */
```

```
        /*
        ============================================================================================================
        SURFACE FUNCTION
        ============================================================================================================
        */

        /*This is a function which will generate the necessary vertex and fragment code when Unity compiles
        the shader.  Things to note:

        SurfaceOutputStandard is a struct that contains all the data needed for Unity to convert the shader
        using a PBR lighting model.
        Note the keyword inout.  This is HLSL syntax.  All parameters in functions for HLSL are PASS BY VALUE.
        This means that without any keywords, the parameter value being inputted will NOT be modified.  To
        make the value PASS BY REFERENCE, use the inout keyword before the parameter.
        inout = making a primitive or struct value a PASS BY REFERENCE value.
        */
        void surf(Input input, inout SurfaceOutputStandard output)
        {
            // Grab color from texture.
            fixed4 color = tex2D(_MainTex, input.uv_MainTex);
            // Multiply the pixel color by _Albedo property.
            color *= _Albedo;
            // Associate our properties to the SurfaceOutputStandard struct.
            output.Albedo = color.rgb;
            output.Metallic = _Metallic;
            output.Smoothness = _Smoothness;
            output.Emission = _Emission;
        }

        /*
        ============================================================================================================
        SURFACE FUNCTION
        ============================================================================================================
        */

        ENDCG
    }

    // If GPU hardware cannot render the shader above, use a builtin shader instead.
    Fallback "Standard"
}
```

Final Result