Lecture title: Real-time in-game shader design and Programming

Prerequisites: Direct3D, HLSL, previous class on game programming

Description: This class is designed to teach student how to create fast and efficient ingame shader that best describe the concept art. The class is structured as a real world programming task where the artist and engine engineer provide a visual in-game target and spec for rendering. The class also explores challenges in facing in-game rendering such as slow shader functions, inefficient normal maps, and overuse of texture maps.

Objective:
1. To understand production game graphics programming requirements
2. Interface with concept art department to achieve desired look in game.
3. Implement fast and efficient real-time game shader.

Class structure:
1. 3 days hands-on programming class
2. 2 programming assignments with similar specs and workflow

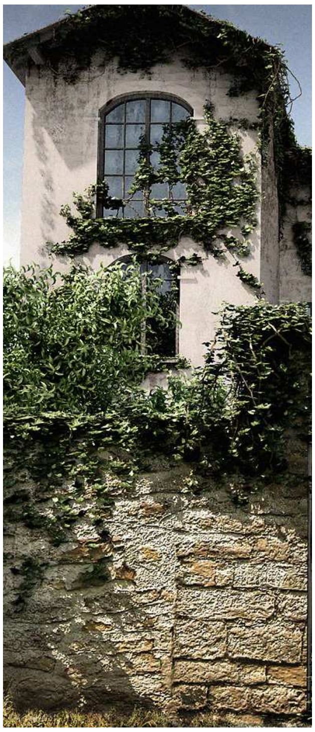
References:
1. Game Developer Conference course on real-time shading.
2. In-game working rendered objects.

Lecture Structure and Detailed flow

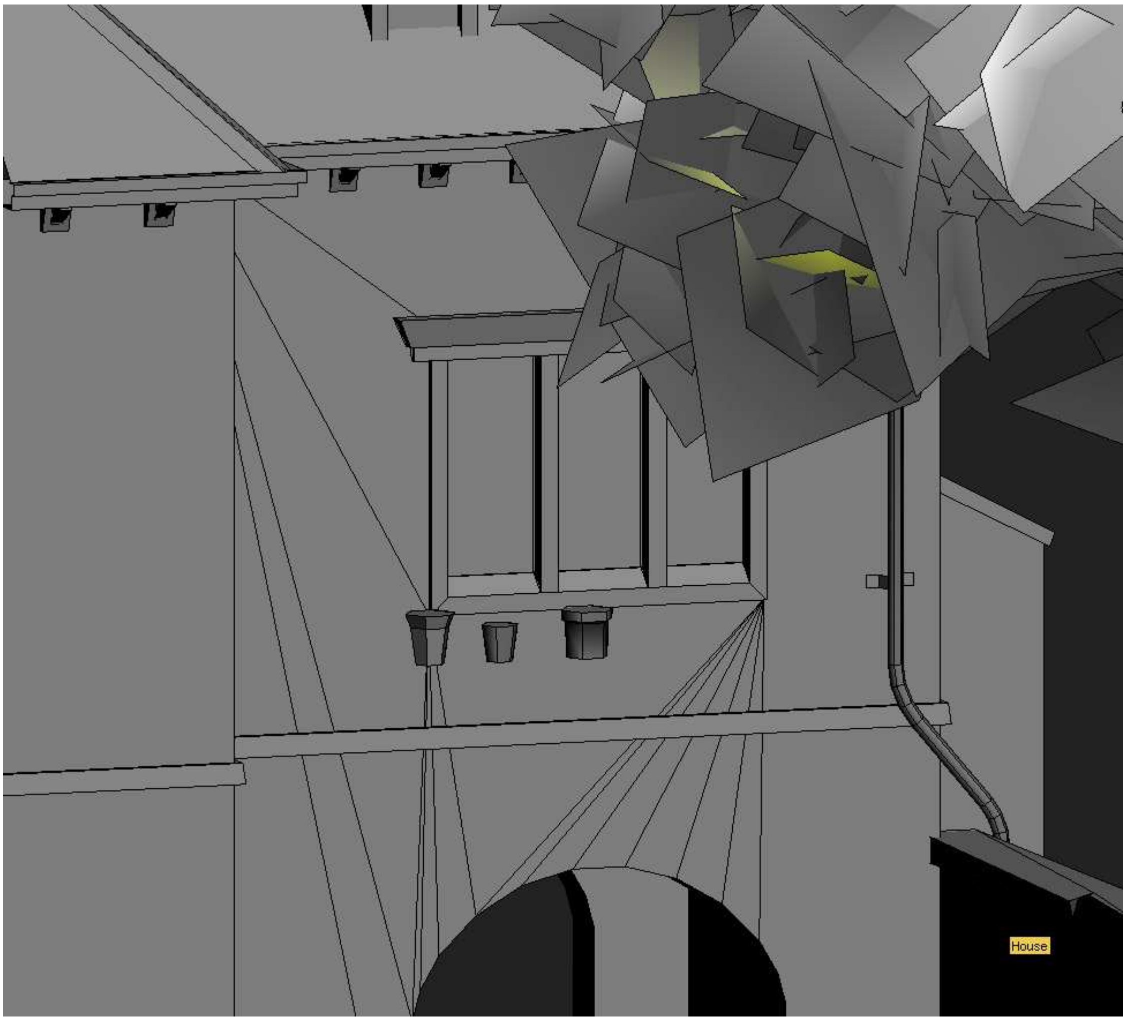
Specification for Consumer House Shading

Art: 2,000 tris, up to 2048x2048 diffuse texture, normal map and mask as appropriate. The limits are the absolute maximum. *if you can build great house in a fraction of the resources and cheapest shader please do so.*

Code Spec**:** please use the RGB specularity model and alpha channel glow. Real-time rendering should look like Concept Art below



1. Artist Concept Drawing

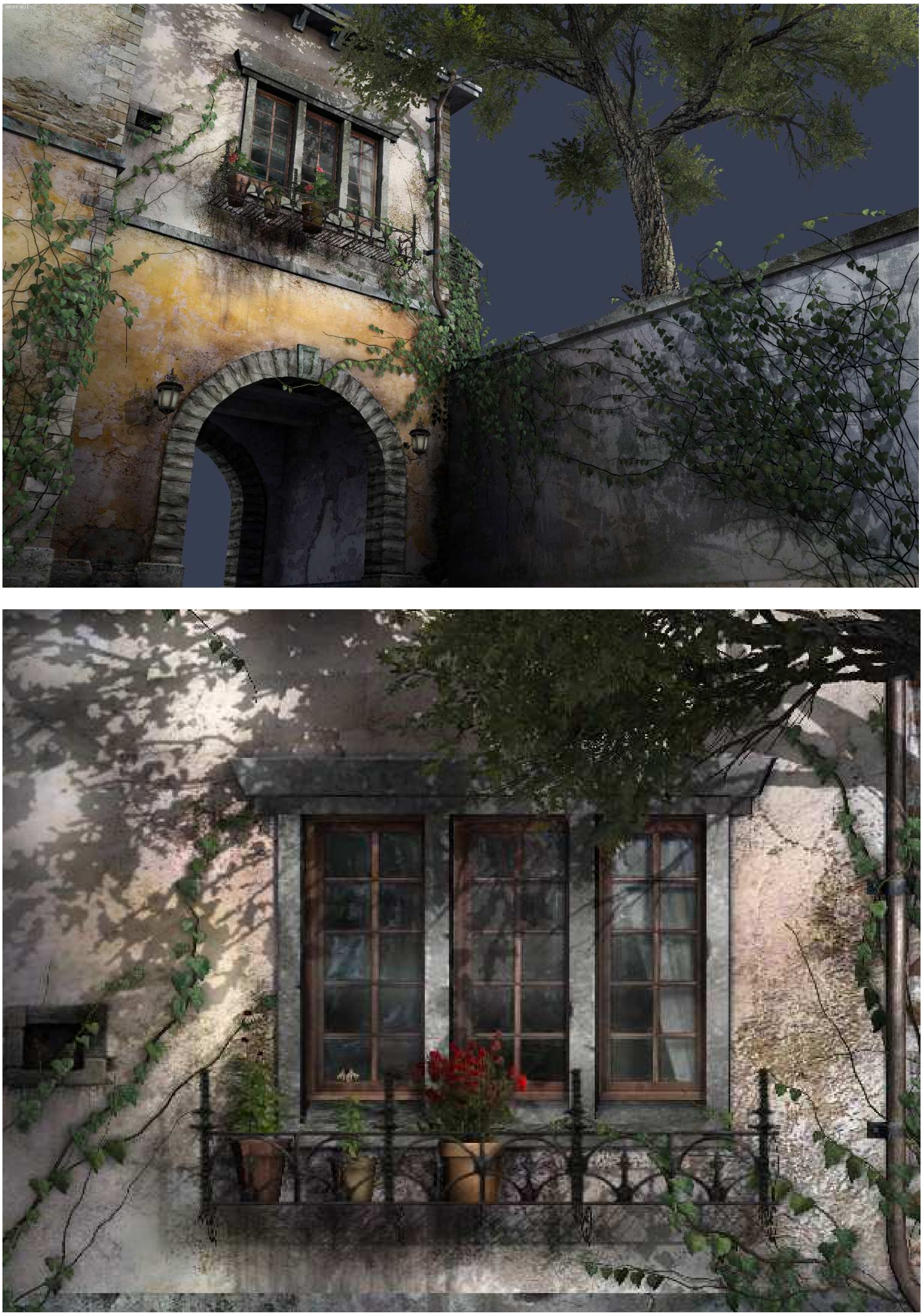


2. Artist Model before real-time shaders

3. Realtime Lighting Shader Created for House

```
float4 af(v2f In, uniform float4 lightColor) : COLOR
{
        ……
        float3 input1 = UIColor_8221;
        float4 TextureMap_8914 = tex2D(TextureMap_8914Sampler, In.texCoord.xy);
        float input7 = TextureMap_8914.a;

        float4 ret =    float4(0,0,0,1);
        ret = float4(input1, 1);
        ret.a = input7;
        ……
}

float4 f(v2f In, uniform float4 lightColor) : COLOR
{
        ……

        float4 TextureMap_8914 = tex2D(TextureMap_8914Sampler, In.texCoord.xy);
        float3 input2 = TextureMap_8914.rgb;
        float input7 = TextureMap_8914.a;


        float4 TextureMap_3753 = tex2D(TextureMap_3753Sampler, In.texCoord.xy);
        float3 input3 = TextureMap_3753.rgb;

        float3 N = input8;                                      //using the Normal socket
        float3 diffuseColor = input2;                   //using the Diffuse Color socket
        float diffuse = saturate(dot(N,L));        //calculate the diffuse
        diffuseColor *= diffuse;                       //the resulting diffuse color
        ret += diffuseColor;                              //add diffuse light to final color
        float3 specularColor = input3;               //using the Specular Color socket
        specularColor *= input4;                    //Multiplying Specular Color by the Specular Level
        float glossiness = 20;                          //the Glossiness socket was empty - using default value
        float3 H = normalize(L + V);                 //Compute the half angle
        float NdotH = saturate(dot(N,H));           //Compute NdotH
        specularColor *= pow(NdotH, glossiness);//Raise to glossiness power and compute final specular color
        ret += specularColor;                            //add specular light to final color
        ……
}
```

x



Real-time Game Engine Results