ADDING INSTANCED INDIRECT SUPPORT TO SHADERS

Vegetation Studio can use instanced indirect rendering on vegetation. This allows the vegetation instanced to be rendered direct from a compute buffer on the GPU allowing for larger batches than the 1023 max of normal rendering. In addition to this there is a final compute shader pass before rendering that does GPU fristum culling and LOD selection.

You can modify most shaders to support Vegetation Studios instanced indirect implementation with a cginc include file and 3 lines of code in the shader. Copy the VS_Indirect.cginc file to the same folder as the shader and add the following lines to the shader. This will set up the shader to call the setup function in the include file where each instance is loading the instance info.

Adding this to a shader does not break normal use of the shader. The setup function is only called when used with the InstancedIndirect API.

The VS_indirect.cginc file is located in the Shader subfolder of VegetationStudio. If someone wants to include this file with a asset store asset they are free to do this.

Currently the instanced indirect rendering is only working on single submesh models. I am working on support for multi submesh.

[enlighter lang=”csharp”]

```
#pragma instancing_options procedural:setup
#pragma multi_compile GPU_FRUSTUM_ON __

#include “VS_indirect.cginc”
```

[/enlighter]

If the shader is a grass and plant shader you can change the first #pragma to this. That will add a function that scales in/out the grass at the vegetation distance for a smoother transition.

[enlighter lang=”csharp”]

```
#pragma instancing_options procedural:setupScale
```

[/enlighter]

If you are updating Speedtree shaders or other shaders that already has the instancing_options pragma add the procedural:setup to the existing #pragma

[enlighter lang=”csharp”]

```
#pragma instancing_options assumeuniformscaling lodfade maxcount:50 procedural:setup
```

[/enlighter]

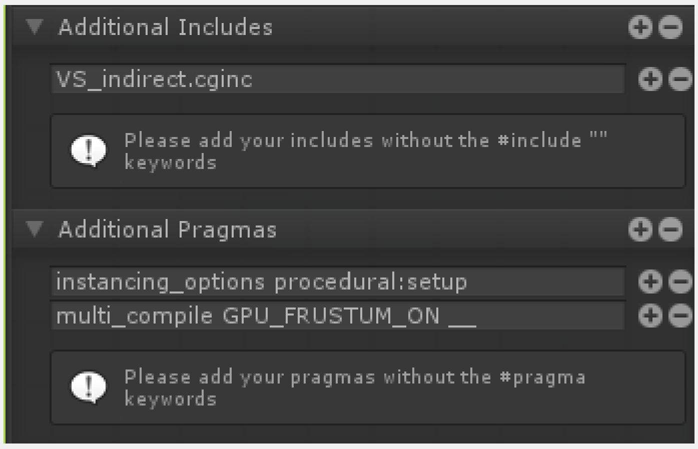
Here is the content of the VS_Indirect.cginc file.

[enlighter lang=”csharp”]

```
#ifdef UNITY_PROCEDURAL_INSTANCING_ENABLED

struct IndirectShaderData
{
float4x4 PositionMatrix;
float4x4 InversePositionMatrix;
float4 ControlData;
};

#if defined(SHADER_API_GLCORE) || defined(SHADER_API_D3D11) ||
defined(SHADER_API_GLES3) || defined(SHADER_API_METAL) || defined(SHADER_API_VULKAN) ||
defined(SHADER_API_PS4) || defined(SHADER_API_XBOXONE)
StructuredBuffer<IndirectShaderData> IndirectShaderDataBuffer;
StructuredBuffer<IndirectShaderData> VisibleShaderDataBuffer;
#endif
#endif

void setupScale()
{
#ifdef UNITY_PROCEDURAL_INSTANCING_ENABLED
#ifdef GPU_FRUSTUM_ON
unity_ObjectToWorld = VisibleShaderDataBuffer[unity_InstanceID].PositionMatrix;
unity_WorldToObject = VisibleShaderDataBuffer[unity_InstanceID].InversePositionMatrix;
#else
unity_ObjectToWorld = IndirectShaderDataBuffer[unity_InstanceID].PositionMatrix;
unity_WorldToObject = IndirectShaderDataBuffer[unity_InstanceID].InversePositionMatrix;
#endif

#ifdef FAR_CULL_ON_PROCEDURAL_INSTANCING
#define transformPosition mul(unity_ObjectToWorld, float4(0,0,0,1)).xyz
#define distanceToCamera length(transformPosition – _WorldSpaceCameraPos.xyz)
float cull = 1.0 – saturate((distanceToCamera – _CullFarStart) / _CullFarDistance);
unity_ObjectToWorld = mul(unity_ObjectToWorld, float4x4(cull, 0, 0, 0, 0, cull, 0, 0, 0, 0, cull, 0, 0, 0,
0, 1));
#undef transformPosition
#undef distanceToCamera
#endif
#endif
}

void setup()
{
#ifdef UNITY_PROCEDURAL_INSTANCING_ENABLED
#ifdef GPU_FRUSTUM_ON
unity_ObjectToWorld = VisibleShaderDataBuffer[unity_InstanceID].PositionMatrix;
unity_WorldToObject = VisibleShaderDataBuffer[unity_InstanceID].InversePositionMatrix;
#else
unity_ObjectToWorld = IndirectShaderDataBuffer[unity_InstanceID].PositionMatrix;
unity_WorldToObject = IndirectShaderDataBuffer[unity_InstanceID].InversePositionMatrix;
#endif
#endif
}
[/enlighter]
```

AMPLIFY SHADER EDITOR

If you are using Amplify Shader Editor to make the shader you can add the same settings there. Copy the VS_indirect.cginc to the same folder as the shader and set the include and pragmas in the shader editor like this.


Additional Includes

VS_indirect.cginc

Please add your includes without the #include "" keywords

Additional Pragmas

instancing_options procedural:setup

multi_compile GPU_FRUSTUM_ON __

Please add your pragmas without the #pragma keywords