OpenGL Instancing

Mike Bailey
mjb@cs.oregonstate.edu

What is Instancing?

Imagine that you needed to draw multiple copies of the same object. Here would be one way to do it (assuming we are using our vertex buffer object C++ class):

```cpp
VertexBufferObject Object;
Object.Init();
Object.glBegin( GL_LINE_STRIP );
Object.glVertex3f( ??, ??, ?? );
. . .
Object.glEnd( );
. . .
for( int i = 0; i < numInstances; i++ )
{
    Object.Draw( );
}
```

This would work, but it would require `numInstances` command transmissions from the CPU to the GPU. Is there a better way?
What is Instancing?

OpenGL, like most graphics APIs (Vulkan, for example), supports a concept called Instancing in which you specify what to draw and how many times to draw it. Using our C++ class, we would use it like this:

```cpp
VertexBufferObject Object;
Object.Init();
Object.glBegin( GL_LINE_STRIP );
Object.glVertex3f( ??, ??, ?? );
...
Object.glEnd( );
...
Object.DrawInstanced( numInstances );
```

This only requires one command transmission from the CPU to the GPU. It essentially moves the execution of the for-loop over to the GPU side.

However, the big problem here is that all those copies of the Object will be drawn identically and on top of each other. Stay tuned.

An Example – Waving Grass

We will instance a single blade of grass to make a waving field:
Making Each Instance Look Differently

There is a built-in vertex shader variable called \texttt{gl_InstanceID} that tells us which instance number is being drawn right now. We can use it to change positions, transformations, colors, etc.

Here's how we draw the straight up-and-down blades of grass in a grid:

\begin{verbatim}
vertices shader:

\texttt{
#version 330 compatibility

uniform float  uTime;
uniform float  uXmin, uXmax;
uniform float  uYmin, uYmax;
uniform float  uPeriodx, uPeriody;
uniform int    uNumx, uNumy;

const float    TWOPI =  2.*3.14159265;

void main( )
{
    int ix = \texttt{gl_InstanceID} % uNumx;
    int iy = \texttt{gl_InstanceID} / uNumx;

    float x = uXmin + float(ix) * (uXmax-uXmin) / float(uNumx-1);
    float y = uYmin + float(iy) * (uYmax-uYmin) / float(uNumy-1);

    vec4 vert = vec4( x, y, gl_Vertex.zw );

    \texttt{gl_Position} = \texttt{gl_ModelViewProjectionMatrix} * vert;
}
}
\end{verbatim}

\end{verbatim}

An Example – Waving Grass

To make the blades sway, we take inspiration from a trick that Pixar used in the movie \textit{A Bug’s Life}:

![Image of parabola]

From the traveling museum exhibit \textit{The Science of Pixar}
An Example – Waving Grass

The vertex shader uses the gl_InstanceID built-in variable to place the blades of grass:

```glsl
void main()
{
    int ix = gl_InstanceID % uNumx;
    int iy = gl_InstanceID / uNumx;
    float x = uXmin + float(ix) * (uXmax-uXmin) / float(uNumx-1);
    float y = uYmin + float(iy) * (uYmax-uYmin) / float(uNumy-1);

    float kx = cos(TWOPI * uTime * float(ix) / uPeriodx);
    float ky = sin(TWOPI * uTime * float(iy) / uPeriody);

    vec4 vert = vec4(x, y, gl_Vertex.zw);
    float zsq = vert.z*vert.z;
    vert.x += kx * zsq;
    vert.y += ky * zsq;

    gl_Position = gl_ModelViewProjectionMatrix * vert;
}
```

Waving Grass – Creating the Shader Program and Setting Uniform Variables in InitGraphics():

```c
Grass.Init();
bool valid = Grass.Create("grass.vert", "grass.frag");
if( ! valid )
    fprintf(stderr, "Shader cannot be created!\n");
else
    fprintf(stderr, "Shader created.\n");
Grass.SetVerbose(false);
Grass.Use();
Grass.SetUniformVariable("uNumx", NUMX);
Grass.SetUniformVariable("uNumy", NUMY);
Grass.SetUniformVariable("uXmin", XMIN);
Grass.SetUniformVariable("uXmax", XMAX);
Grass.SetUniformVariable("uYmin", YMIN);
Grass.SetUniformVariable("uYmax", YMAX);
Grass.SetUniformVariable("uPeriodx", PERIODX);
Grass.SetUniformVariable("uPeriody", PERIODY);
Grass.UnUse();
```
Waving Grass – Create the Grass-Blade Vertex Buffer Object in InitGraphics( ):

Blade.Init();
Blade.glBegin( GL_LINE_STRIP );
for( int i = 0; i < NUMPOINTS; i++ )
{
    float z = ZMIN + (float)i * (ZMAX-ZMIN) / (float)(NUMPOINTS-1);
    Blade.glVertex3f( 0., 0., z );
}
Blade.glEnd();

Waving Grass – Drawing the Field of Grass in Display( ):

// turn on the shader and set the time:
Grass.Use( );
Grass.SetUniformVariable( "uTime", Time );

// draw the grass field:
Blade.DrawInstanced( NUMX*NUMY );
Grass.UnUse( );