ANNOUNCING EUCLIDEON SDK 1.2

3 September 2013

Euclideon is excited to announce the release of revision 1.2 of our award-winning Euclideon Unlimited Detail SDK. UDSDK is the only technology in the world that can deliver the following ground-breaking and completely unique features:

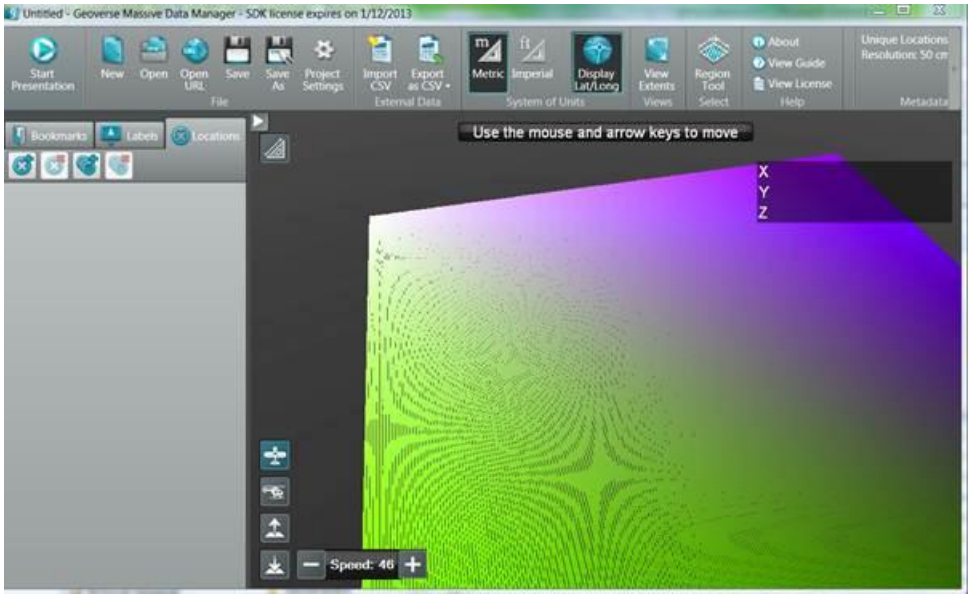
- **Instant Loading** of point-clouds of any size up to the amount addressable by the host OS – currently 128TB.
- **Streaming Point Clouds** of any size at ultra-high resolutions at frame rates of at least 35FPS from hard drives, network storage, USB drives and even across the Internet via HTTP and securely encrypted HTTPS.
- **Conversion** of 3.5B points per hour and reduction to as little as 17% of original data size.
- **Federation** of huge multi-city data-sets onto a single, remotely accessible cloud server.
- **Innovative** new display techniques such as 3D Anaglyphic Stereo display with unique Real-Time Depth Adjustment technology.

All these are features your organization is already taking advantage of. Euclideon has listened carefully to feedback from our partners and this has all been incorporated in revision 1.2 of UDSDK. Here is a brief summary of the new features.

Embedded Classification Data - NEW

Customizable user-defined classification data (i.e. attributes such as intensity) from a variety of sources can now be embedded directly into the points stored in our UDS/UDG formats by applications using the udExport part of the SDK (such as Geoverse Convert). This means that all of the classification data your organization already has or uses can be displayed by applications at the ultra-high resolutions and frame rates you have grown accustomed to seeing from Unlimited Detail.

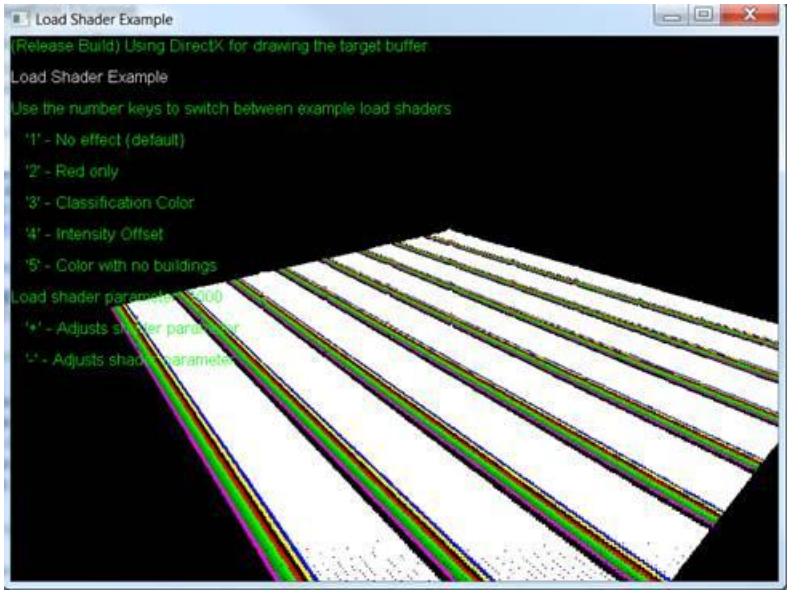
Our UDSDK Samples Pack includes source code and a pre-compiled binary for a sample called **AttributeExport** that outputs a UDS file called **AttributeExport.UDS**. This Unlimited Detail file includes a variety of attributes, as can be seen below:



Applications built with the UDSDK can easily access these attributes in the same way that 3D points in the scene are currently accessed. This opens the door to a very wide range of new applications for Unlimited Detail – vegetation management, power-line maintenance, mining and geology just to name a few.

The conversion functions of UDSDK allow users to embed their own custom classification data into the points in the UDS file during conversion, so the data can be completely customized from beginning to end. The details of this are also demonstrated by the **AttributeExport** sample.

Displaying the classification data requires the use of a small 'shader' program called a **Load Shader**. The **LoadShader** sample demonstrates how to create a few different types of **Load Shaders** to utilize the embedded classification data in the UDS file:



Example of a Load Shader isolating particular points based on their classification and displaying these in a different colour.

A **Load Shader** is a fragment of C++ code that looks like this example. **Load Shaders** can be used to completely customize the display of points based on classification data in an efficient way:

```
//=============================================================================================
// Classify load shader.
// This load shader accesses the 8-bit classification data and looks up a color to display based on
// the value.
// The code layout of this shader shows a C++ inheritance style layout.
//=============================================================================================
struct ClassifyShader : public udLoadShader
{
  ClassifyShader() : udLoadShader()
  {
    destroy            = Destroy;
    getShaderAttributes = GetShaderAttributes;
    getDefaultValues    = GetDefaultValues;
    shadePoint          = ShadePoint;
  }

  static void Destroy(struct udLoadShader *) {}
  static void GetShaderAttributes(struct udLoadShader *, const char ** a_attributeString) { *a_attributeString = "udClassificati
  static void GetDefaultValues(struct udLoadShader *, void * const a_attributeData[]) { *(uint8_t*)a_attributeData[0] = 0; }
  static uint32_t ShadePoint(struct udLoadShader *, const void * const a_attributeData[])
  {
    uint8_t srcClassify = *(const uint8_t*)a_attributeData[0];

    // Assuming LAS classification data
    switch(srcClassify & 0x1F)
    {
      case(0): return 0xFF000000; // Created, never classified
      case(1): return 0xFF000000; // Unclassified
      case(2): return 0xFFFF00FF; // Ground
      case(3): return 0xFF008F00; // Low Vegetation
      case(4): return 0xFF00BF00; // Medium Vegetation
      case(5): return 0xFF00FF00; // High Vegetation
      case(6): return 0xFFFF0000; // Building
      case(7): return 0xFF000000; // Low Point (noise)
      case(8): return 0xFFFFFF00; // Model Key-point
      case(9): return 0xFF0000FF; // Water

      default: return 0xFFFFFFFF;
    }
  }

};
```

Geometric Query Support - NEW

The other major feature that UDSDK 1.2 introduces is world-leading support for highly sophisticated Constructive Solid Geometry (CSG) queries. CSG queries in various forms applicable to point cloud data have existed for a long time, but they have always suffered one major handicap – they are pretty slow, because the algorithms used all suffered from the same classic speed problems as all point cloud software did before Unlimited Detail was created.

udQuery can now:

- Retrieve points from a dataset that satisfy a geometric constraint
- Define geometric constraints using volume primitives and Constructive Solid Geometry (CSG) combinations.
- Use maximum point count and minimum separation filters when querying points
- Alternate 'rendering' of the visible points to a list of model space X,Y,Z values and obtain the attribute values
- Query N Nearest Neighbours

Our CSG query method is world-leading because these queries are now conducted using the Unlimited Detail algorithm. This means that these queries can be previewed in real time – as demonstrated by the **QueryTestGUI** sample, where we loaded up a 5.3GB dataset of the Southern Expressway in South Australia and defined a polygonal region without any slow-down or massive computation:

Notice how the selected area is completely accurate down the individual 3D points, even though the selection of the 3D polygon happens in real time. Once the area is selected, a CSG query can be formed to output the data in whatever form the user wishes to see it. This includes only gathering a user-specified percentage of the points in the volume to create a ‘Draft’ version.

Other forms of query and filtering for the displayed query points such as inclusive/exclusive display are also possible:

This system is hugely customizable and delivers a lot of value to customers with existing classified data-sets.