Report Lambda Examples

```
using System;
using System.IO;
using System.IO.Compression;
using VoxelFarm.SpatialLambda;

namespace TestLambdas
{
   // This Report Lambda uses a Sum to compute the volume of an object

   public class EntityVolume : VoxelFarm.SpatialLambda.IVoxelFarmReportLambda
   {
      public SpatialLambdaResult Done(IVoxelFarmReportDoneLambdaHost host)
      {
         var result = new SpatialLambdaResult(0);
         return result;
      }

      public SpatialLambdaResult RunReport(IVoxelFarmReportLambdaHost host)
      {
         // get inputs
         int entity = host.InputEntity("0", "Volume Source",
            VoxelFarm.SpatialLambda.EntityType.VoxelTerrain |
            VoxelFarm.SpatialLambda.EntityType.BlockModel |
            VoxelFarm.SpatialLambda.EntityType.MaterialTracking |
            VoxelFarm.SpatialLambda.EntityType.VoxelGenerator);

         // ask the platform to load voxels for the entity including volume for each voxel
         int voxels = host.LoadVoxels(entity, VoxelFarm.SpatialLambda.Attribute.Volume, "");

         double sum = 0.0;

         int channelCount = host.GetChannelCount(voxels);
         for (int channel = 0; channel < channelCount; channel++)
         {
            // get array with volume values, one per voxel
```

```
            float[] volumes = host.GetVolumeBuffer(voxels, channel);

            // the array can be null if all volume values are zero
            if (volumes != null)
            {
                // add volumes together
                double voxelSize = host.GetVoxelVolume();
                foreach (float v in volumes)
                {
                    sum += v * voxelSize;
                }
            }
        }

        // report sum results
        int sumId = host.StartSum("Object", "Volume");
        host.Sum(sumId, sum);

        var result = new SpatialLambdaResult(0);
        return result;
    }
}

// This Report Lambda uses a List to create a block model file in CSV format out
// of another volumetric object

public class CreateBlockModel : VoxelFarm.SpatialLambda.IVoxelFarmReportLambda
{
    public SpatialLambdaResult RunReport(IVoxelFarmReportLambdaHost host)
    {
        // get inputs
        int entity = host.InputEntity("0", "Source",
            VoxelFarm.SpatialLambda.EntityType.VoxelTerrain |
            VoxelFarm.SpatialLambda.EntityType.BlockModel |
            VoxelFarm.SpatialLambda.EntityType.MaterialTracking |
            VoxelFarm.SpatialLambda.EntityType.VoxelGenerator);

        // get buffer with volumes
        int voxels = host.LoadVoxels(entity, VoxelFarm.SpatialLambda.Attribute.Volume, "");

        float[] volumes = host.GetVolumeBuffer(voxels, 0);
```

```
    // list voxels with any volume in them
    if (volumes != null)
    {
        double voxelSize = host.GetVoxelSize();
        double[] centroids = host.GetVoxelCentroids();

        var list = host.StartList("blocks");
        int index = 0;
        foreach (float v in volumes)
        {
            if (v > 0.0)
            {
                double x = centroids[index];
                double y = centroids[index + 1];
                double z = centroids[index + 2];
                string item = x + "," + y + "," + z + "," + voxelSize + "," + voxelSize + "," + voxelSize + "," + v;
                host.List(list, item);
            }
            index += 3;
        }
    }

    return new SpatialLambdaResult(0);
}


public SpatialLambdaResult Done(IVoxelFarmReportDoneLambdaHost host)
{
    string compressionFolder = host.CreateTempFolder("compress");
    string uploadsFolder = host.CreateTempFolder("uploads");
    string blockModelFileName = compressionFolder + "block_model.csv";
    string headersFileName = uploadsFolder + "headers.csv";
    StreamWriter file = new StreamWriter(blockModelFileName);
    StreamWriter headers = new StreamWriter(headersFileName);
    file.WriteLine("XC,YC,ZC,XS,YS,ZS,DENSITY");
    headers.WriteLine("XC,YC,ZC,XS,YS,ZS,DENSITY");
    headers.Close();
    host.ScanList("blocks", (item) =>
    {
        file.WriteLine(item);
```

```
            return false;
        });
        file.Close();
        ZipFile.CreateFromDirectory(compressionFolder, uploadsFolder + "block_model.zip");
        host.AttachFile("block_model.zip", uploadsFolder + "block_model.zip");
        host.AttachFile("headers.csv", headersFileName);
        return new SpatialLambdaResult(0);
    }
  }
}
```

Revision #1
Created 17 March 2025 19:37:48 by admin
Updated 17 March 2025 19:41:49 by admin