Unity3D中的地形转成模型

作者 浪尖儿 (/u/df408a2ef65b) + 关注
2016.08.17 17:02* 字数 569 阅读 483 评论 0 喜欢 2
(/u/df408a2ef65b)

Unity3D中的地形转成模型

起因

为什么要把地形转成模型呢？在Unity3D中创建地形很方便，用它自带的地形编辑工具，各种跌宕起伏的地形都能很容易的创建出来。但是也有一些不方便的地方，比如创建好的地形不能整体缩放，只能通过修改长宽等参数进行调整。偏偏就有这样的需求，我们要把地形放到虚拟研讨厅的桌子上当做数字沙盘去展示。如果把地形对象的长宽高都缩小的话，高度图也要缩小，地形效果就太不好了。如果地形对象能像模型对象一样，随意的缩放就好了。

解决方法

终于在网上找到了解决方法，有大神贡献了一个脚本，能够把地形对象转换成模型对象，模型格式为obj，可以直接导入到Unity3D中使用。

```
using UnityEngine;
using UnityEditor;
using System;
using System.Collections;
using System.IO;using System.Text;

enum SaveFormat { Triangles, Quads }
enum SaveResolution { Full=0, Half, Quarter, Eighth, Sixteenth }

class ExportTerrain : EditorWindow
{
   SaveFormat saveFormat = SaveFormat.Triangles;
   SaveResolution saveResolution = SaveResolution.Half;

   static TerrainData terrain;
   static Vector3 terrainPos;

   int tCount;
   int counter;
   int totalCount;
   int progressUpdateInterval = 10000;

   [MenuItem("Terrain/Export To Obj...")]
   static void Init()
   {
      terrain = null;
      Terrain terrainObject = Selection.activeObject as Terrain;
      if (!terrainObject)
      {
         terrainObject = Terrain.activeTerrain;
      }
      if (terrainObject)
      {
         terrain = terrainObject.terrainData;
         terrainPos = terrainObject.transform.position;
      }

      EditorWindow.GetWindow<ExportTerrain>().Show();
   }

   void OnGUI()
   {
      if (!terrain)
      {
         GUILayout.Label("No terrain found");
         if (GUILayout.Button("Cancel"))
         {
            EditorWindow.GetWindow<ExportTerrain>().Close();
         }
         return;
      }
      saveFormat = (SaveFormat) EditorGUILayout.EnumPopup("Export Format", saveFormat);

      saveResolution = (SaveResolution) EditorGUILayout.EnumPopup("Resolution", saveResoluti

      if (GUILayout.Button("Export"))
      {
         Export();
      }
   }

   void Export()
   {
      string fileName = EditorUtility.SaveFilePanel("Export .obj file", "", "Terrain", "obj"
      int w = terrain.heightmapWidth;
      int h = terrain.heightmapHeight;
      Vector3 meshScale = terrain.size;
      int tRes = (int)Mathf.Pow(2, (int)saveResolution );
      meshScale = new Vector3(meshScale.x / (w - 1) * tRes, meshScale.y, meshScale.z / (h -
      Vector2 uvScale = new Vector2(1.0f / (w - 1), 1.0f / (h - 1));
      float[,] tData = terrain.GetHeights(0, 0, w, h);

      w = (w - 1) / tRes + 1;
      h = (h - 1) / tRes + 1;
      Vector3[] tVertices = new Vector3[w * h];
      Vector2[] tUV = new Vector2[w * h];

      int[] tPolys;

      if (saveFormat == SaveFormat.Triangles)
      {
         tPolys = new int[(w - 1) * (h - 1) * 6];
      }
      else
      {
         tPolys = new int[(w - 1) * (h - 1) * 4];
```

```
}

// Build vertices and UVs
for (int y = 0; y < h; y++)
{
   for (int x = 0; x < w; x++)
   {
      tVertices[y * w + x] = Vector3.Scale(meshScale, new Vector3(-y, tData[x * tRes,
      tUV[y * w + x] = Vector2.Scale( new Vector2(x * tRes, y * tRes), uvScale);
   }
}

int  index = 0;
if (saveFormat == SaveFormat.Triangles)
{
   // Build triangle indices: 3 indices into vertex array for each triangle
   for (int y = 0; y < h - 1; y++)
   {
      for (int x = 0; x < w - 1; x++)
      {
         // For each grid cell output two triangles
         tPolys[index++] = (y * w) + x;
         tPolys[index++] = ((y + 1) * w) + x;
         tPolys[index++] = (y * w) + x + 1;

         tPolys[index++] = ((y + 1) * w) + x;
         tPolys[index++] = ((y + 1) * w) + x + 1;
         tPolys[index++] = (y * w) + x + 1;
      }
   }
}
else
{
   // Build quad indices: 4 indices into vertex array for each quad
   for (int y = 0; y < h - 1; y++)
   {
      for (int x = 0; x < w - 1; x++)
      {
         // For each grid cell output one quad
         tPolys[index++] = (y * w) + x;
         tPolys[index++] = ((y + 1) * w) + x;
         tPolys[index++] = ((y + 1) * w) + x + 1;
         tPolys[index++] = (y * w) + x + 1;
      }
   }
}

// Export to .obj
StreamWriter sw = new StreamWriter(fileName);
try
{

   sw.WriteLine("# Unity terrain OBJ File");

   // Write vertices
   System.Threading.Thread.CurrentThread.CurrentCulture = new System.Globalization.Cul
   counter = tCount = 0;
   totalCount = (tVertices.Length * 2 + (saveFormat == SaveFormat.Triangles ? tPolys.L
   for (int i = 0; i < tVertices.Length; i++)
   {
      UpdateProgress();
      StringBuilder sb = new StringBuilder("v ", 20);
      // StringBuilder stuff is done this way because it's faster than using the "{0}
      // Which is important when you're exporting huge terrains.
      sb.Append(tVertices[i].x.ToString()).Append(" ").
         Append(tVertices[i].y.ToString()).Append(" ").
         Append(tVertices[i].z.ToString());
      sw.WriteLine(sb);
   }
   // Write UVs
   for (int i = 0; i < tUV.Length; i++)
   {
      UpdateProgress();
      StringBuilder sb = new StringBuilder("vt ", 22);
      sb.Append(tUV[i].x.ToString()).Append(" ").
         Append(tUV[i].y.ToString());
      sw.WriteLine(sb);
   }
   if (saveFormat == SaveFormat.Triangles)
   {
      // Write triangles
      for (int i = 0; i < tPolys.Length; i += 3)
      {
         UpdateProgress();
         StringBuilder sb = new StringBuilder("f ", 43);
         sb.Append(tPolys[i] + 1).Append("/").Append(tPolys[i] + 1).Append(" ").
            Append(tPolys[i + 1] + 1).Append("/").Append(tPolys[i + 1] + 1).Append(" "
```

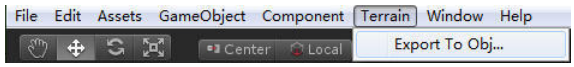
```
                Append(tPolys[i + 2] + 1).Append("/").Append(tPolys[i + 2] + 1);
            sw.WriteLine(sb);
         }
      }
      else
      {
         // Write quads
         for (int i = 0; i < tPolys.Length; i += 4)
         {
            UpdateProgress();
            StringBuilder sb = new StringBuilder("f ", 57);
            sb.Append(tPolys[i] + 1).Append("/").Append(tPolys[i] + 1).Append(" ").
               Append(tPolys[i + 1] + 1).Append("/").Append(tPolys[i + 1] + 1).Append(" "
               Append(tPolys[i + 2] + 1).Append("/").Append(tPolys[i + 2] + 1).Append(" "
               Append(tPolys[i + 3] + 1).Append("/").Append(tPolys[i + 3] + 1);
            sw.WriteLine(sb);
         }
      }
   }
   catch(Exception err)
   {
      Debug.Log("Error saving file: " + err.Message);
   }
   sw.Close();

   terrain = null;
   EditorUtility.DisplayProgressBar("Saving file to disc.", "This might take a while...",
   EditorWindow.GetWindow<ExportTerrain>().Close();
   EditorUtility.ClearProgressBar();
}

void UpdateProgress()
{
   if (counter++ == progressUpdateInterval)
   {
      counter = 0;
      EditorUtility.DisplayProgressBar("Saving...", "", Mathf.InverseLerp(0, totalCount,
   }
}
}
```

使用方法

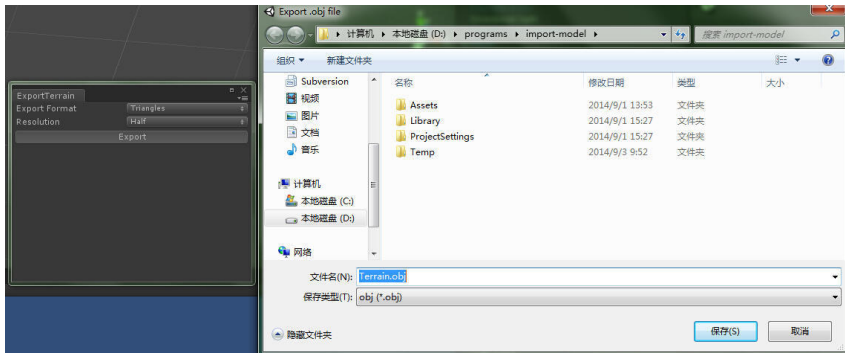
给脚本命名为ExportTerrain.cs。
这个脚本文件一定要放到工程中Assets文件夹下面的Editor文件夹中（没有的话自己创建）才能正常工作。
完成前两步之后，unity菜单项会多出一个Terrain/Export To Obj...的菜单(4.3以前的版本本来就有Terrain菜单，只是多了个子菜单；4.3版本里面默认没有Terrain菜单了)。



terrain-menu

选择场景要转成模型的地形对象。如果什么都没选的话，会使用默认的active terrain。

然后选择Terrain/Export To Obj...菜单，弹出下面的对话框，选择导出格式（triangles or quads）、Mesh分辨率（full, half, quarter, eighth or sixteenth）、文件名和路径，然后点击Export。



terrain-expoty

等待进度条跑完之后就OK了，obj文件就导出成功了。

See Also

和地形转模型的方法相反的一个操作，还有模型转地形的方法，类似的也是有脚本完成的，感兴趣可以参考Object2Terrain[2].

参考

http://wiki.unity3d.com/index.php?title=TerrainObjExporter (http://wiki.unity3d.com/index.php?title=TerrainObjExporter)
http://wiki.unity3d.com/index.php?title=Object2Terrain (http://wiki.unity3d.com/index.php?title=Object2Terrain)