

Introduction to Meshes

Recommended Reading

[Rendering Lights](#)

Video

http://www.leadwerks.com/files/Tutorials/CPP/Introduction_To_Meshes.wmv

Required Files

http://www.leadwerks.com/files/Tutorials/CPP/Introduction_To_Meshes_Files.zip

Getting Started

We'll start with this simple program:

```
#include "engine.h"

int main(int argc, char** argv)
{
    Initialize();

    //Create a graphics context
    Graphics(800,600);

    //Load SDK directory
    RegisterAbstractPath("C:\\Leadwerks Engine SDK");

    //Create a world
    if (!CreateWorld()) {
        MessageBoxA(0, "Failed to create world.", "Error", 0);
        goto exitapp;
    }

    //Create a camera
    TCamera cam=CreateCamera();
    CameraClearColor(cam, Vec4(0,0,1,1));
    RotateEntity(cam, Vec3(35,0,0));
    MoveEntity(cam, Vec3(0,0,-4));

    //Create a light
    TLight light=CreateDirectionalLight();
    RotateEntity(light, Vec3(45,45,0));
}
```

```

//Create a render buffer
TBuffer buffer=CreateBuffer(800,600,BUFFER_COLOR|BUFFER_DEPTH|BUFFER_NORMAL);

//Main loop
while(!KeyHit(KEY_ESCAPE)) {

    //Update the world
    UpdateWorld();

    //Render the scene
    SetBuffer(buffer);
    RenderWorld();

    //Render lighting
    SetBuffer(BackBuffer());
    RenderLights(buffer);

    //Swap the front and back buffer
    Flip();

}

exitapp:
Terminate();
return 0;
}

```

There are three ways to create a mesh. We can create a mesh primitive, load a mesh from a file, or create a mesh from scratch. We've already seen mesh primitives in previous tutorials. Add this code before the main loop:

```
TMesh mesh1=CreateCube();
```

We can also load a mesh from a file. We can load meshes from the OBJ, MD3, 3DW, and GMF formats. Make sure the *box.obj* file is in the same folder as your source code and add this line to the program:

```
TMesh mesh2=LoadMesh("box.gmf");
```

Let's add a check to make sure the mesh gets loaded. Add this after the previous block of code:

```

if (!mesh2) {
    MessageBoxA(0,"Failed to load mesh. ","Error",0);
    goto exitapp;
}

```

Let's also load a material and apply it to both meshes. Add this *after* both meshes have been created:

```
TMaterial mat=LoadMaterial("brick.mat");
if (!mat) {
    MessageBoxA(0,"Failed to load material. ","Error",0);
    goto exitapp;
}

PaintEntity(mesh1,mat);
PaintEntity(mesh2,mat);
```

Now let's move the meshes so they aren't on top of each other. Add this code *after* the two meshes have been created:

```
PositionEntity mesh1,Vec3(-2,0,0)
PositionEntity mesh2,Vec3(0,0,0)
```

Finally, let's add some code in the main loop to make the two meshes spin:

```
TurnEntity(mesh1,Vec3(0,0.5,0));
TurnEntity(mesh2,Vec3(0,0.5,0));
```

Here is our final code:

```
#include "engine.h"

int main(int argc, char** argv)
{
    Initialize();

    //Create a graphics context
    Graphics(800,600);

    //Load SDK directory
    RegisterAbstractPath("C:\\Leadwerks Engine SDK");

    //Create a world
    if (!CreateWorld()) {
        MessageBoxA(0,"Error","Failed to create world.",0);
        goto exitapp;
    }

    //Create a camera
    TCamera cam=CreateCamera();
    CameraClearColor(cam,Vec4(0,0,1,1));
    RotateEntity(cam,Vec3(35,0,0));
    MoveEntity(cam,Vec3(0,0,-4));
```

```

//Create a light
TLight light=CreateDirectionalLight();
RotateEntity(light,Vec3(45,45,0));

//Create a render buffer
TBuffer buffer=CreateBuffer(800,600,BUFFER_COLOR|BUFFER_DEPTH|BUFFER_NORMAL);

TMesh mesh1=CreateCube();

TMesh mesh2=LoadMesh("box.gmf");
if (!mesh2) {
    MessageBoxA(0,"Error","Failed to load mesh.",0);
    goto exitapp;
}

//Load a material
TMaterial mat=LoadMaterial("brick.mat");
if (!mat) {
    MessageBoxA(0,"Error","Failed to load material.",0);
    goto exitapp;
}

//Apply the material to the meshes
PaintEntity(mesh1,mat);
PaintEntity(mesh2,mat);

//Position the meshes
PositionEntity(mesh1,Vec3(-2,0,0));
PositionEntity(mesh2,Vec3(0,0,0));

//Main loop
while(!KeyHit(KEY_ESCAPE)) {

    TurnEntity(mesh1,Vec3(0,0.5,0));
    TurnEntity(mesh2,Vec3(0,0.5,0));

    //Update the world
    UpdateWorld();

    //Render the scene
    SetBuffer(buffer);
    RenderWorld();

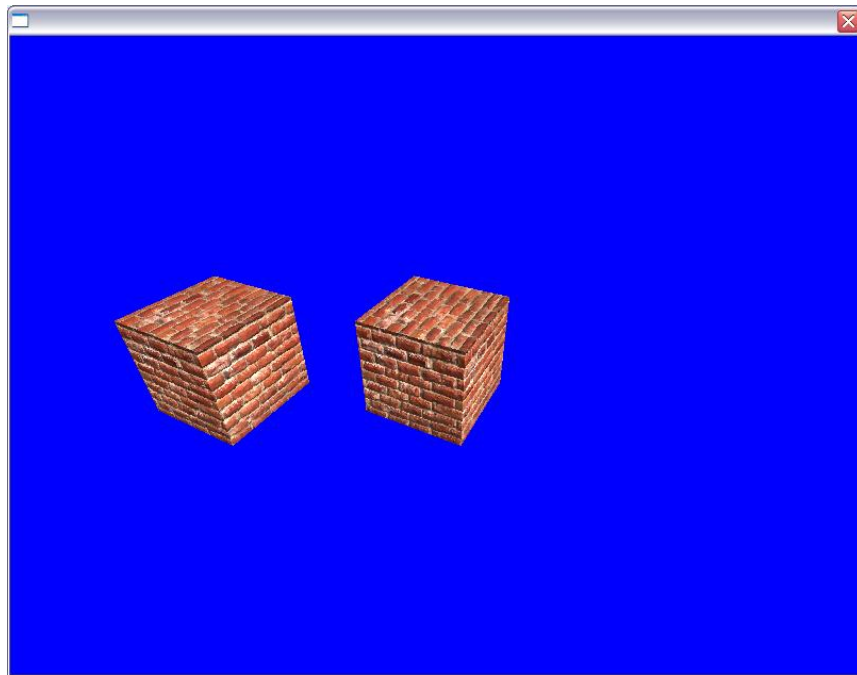
    //Render lighting
    SetBuffer(BackBuffer());
    RenderLights(buffer);

    //Swap the front and back buffer
    Flip();
}

exitapp:
return Terminate();
}

```

When we run the program we can see the mesh created with the CreateCube command on the left, and the mesh loaded from a file on the right:



Creating Meshes from Scratch

We can also create meshes from scratch by adding vertices and triangles in code. Let's create a new mesh. Add this code after the material has been loaded:

```
TMesh mesh3=CreateMesh();
```

The mesh is empty and won't be visible until we add data to it. Meshes are made of surfaces. Each surface has a vertex array, indice array, and can have a material applied to it. Surfaces move with the mesh and can't be positioned or rotated on their own. All triangles in a surface share the same material, and are drawn in one batch on the GPU.

Let's create a surface on our mesh:

```
TSurface surf=CreateSurface(mesh3);
```

Now we are going to add three vertices to the surface. We pass the vertex position to the vertex creation function. We can also pass optional data including normals and texture coordinates:

```
AddVertex(surf,Vec3(-0.5,0,0));  
AddVertex(surf,Vec3(0.5,1,0));  
AddVertex(surf,Vec3(0.5,0,0));
```

Now we can add a single triangle to the mesh. We pass the vertex indices to the triangle creation function. Vertices are numbered starting at 0:

```
AddTriangle(surf,0,1,2);
```

Any time we modify a mesh, we need to call UpdateMesh to tell the engine to calculate the bounding box and other things. Add this code after the triangle is created:

```
UpdateMesh(mesh3);
```

Finally, let's position and paint the mesh:

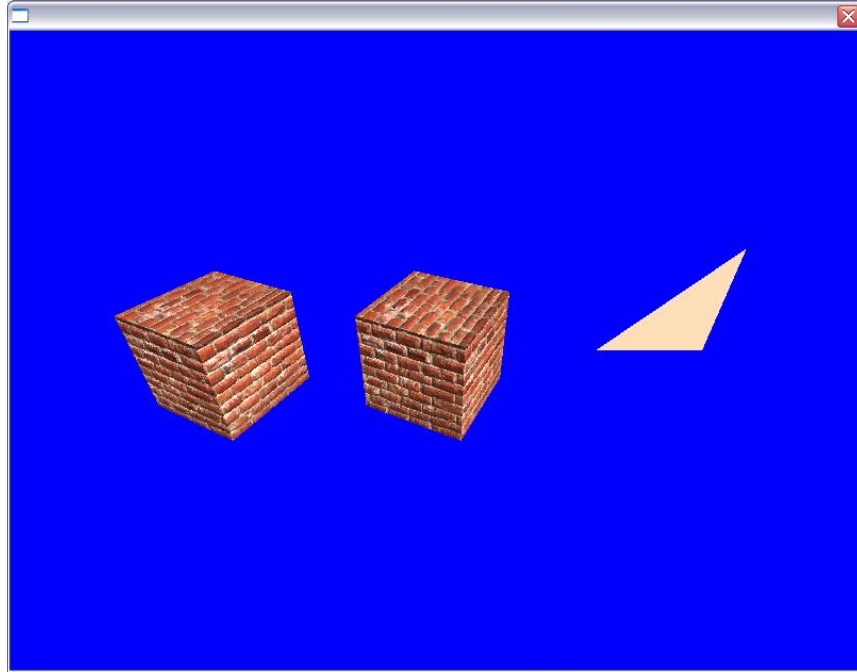
```
PositionEntity(mesh3,Vec3(2,0,0));  
PaintEntity(mesh3,mat);
```

Here is our complete mesh creation code. Note that this must be added after the material is loaded:

```
TMesh mesh3=CreateMesh();  
  
TSurface surf=CreateSurface(mesh3);  
AddVertex(surf,Vec3(-0.5,0,0));  
AddVertex(surf,Vec3(0.5,1,0));  
AddVertex(surf,Vec3(0.5,0,0));  
AddTriangle(surf,0,1,2);  
  
UpdateMesh(mesh3);  
  
PositionEntity(mesh3,Vec3(2,0,0));
```

```
PaintEntity(mesh3,mat);
```

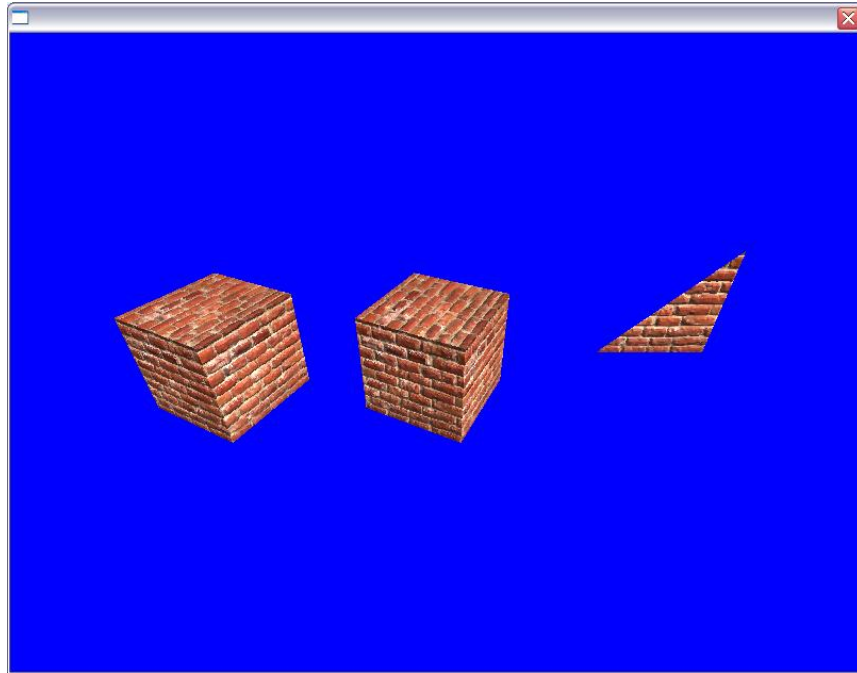
When we run our program, we can see the two boxes along with our new mesh:



But our mesh doesn't look quite right. The texture isn't appearing and the lighting seems wrong. This is because the mesh does not have texture coordinates or normals. We can add these elements to our created vertices as follows. The vertex normal is a 3-dimensional vector, and the texture coordinate is a 2-component vector. Add this code after the vertices are created, and before the call to UpdateMesh:

```
SetVertexNormal (surf,0,Vec3(0,0,-1));  
SetVertexNormal (surf,1,Vec3(0,0,-1));  
SetVertexNormal (surf,2,Vec3(0,0,-1));  
SetVertexTexCoords (surf,0,Vec2(0,1));  
SetVertexTexCoords (surf,1,Vec2(1,0));  
SetVertexTexCoords (surf,2,Vec2(1,1));
```

When we run the program we can see that now the mesh appears as we would expect:



Advanced Mesh Creation

Using these same principles we can create a third box entirely from scratch:

```
#include "engine.h"

int main(int argc, char** argv)
{
    Initialize();

    //Create a graphics context
    Graphics(800,600);

    //Load SDK directory
    RegisterAbstractPath("C:\\Leadwerks Engine SDK");

    //Create a world
    if (!CreateWorld()) {
        MessageBoxA(0,"Failed to create world. ","Error",0);
        goto exitapp;
    }

    //Create a camera
    TEntity cam=CreateCamera();
    CameraClearColor(cam,Vec4(0,0,1,1));
    RotateEntity(cam,Vec3(35,0,0));
    MoveEntity(cam,Vec3(0,0,-4));

    //Create a light
    TLight light=CreateDirectionalLight();
}
```

```

RotateEntity(light,Vec3(45,45,0));

//Create a render buffer
TBuffer buffer=CreateBuffer(800,600,BUFFER_COLOR|BUFFER_DEPTH|BUFFER_NORMAL);

TMesh mesh1=CreateCube();

TMesh mesh2=LoadMesh("box.gmf");
if (!mesh2) {
    MessageBoxA(0,"Failed to load mesh.,"Error",0);
    goto exitapp;
}

//Load a material
TMaterial mat=LoadMaterial("brick.mat");
if (!mat) {
    MessageBoxA(0,"Failed to load material.,"Error",0);
    goto exitapp;
}

//Apply the material to the meshes
PaintEntity(mesh1,mat);
PaintEntity(mesh2,mat);

//Position the meshes
PositionEntity(mesh1,Vec3(-2,0,0));
PositionEntity(mesh2,Vec3(0,0,0));

//Create a box from scratch
TMesh mesh3=CreateMesh();
TSurface surf=CreateSurface(mesh3);

AddVertex(surf,Vec3(-0.5,-0.5,-0.5));
AddVertex(surf,Vec3(-0.5, 0.5,-0.5));
AddVertex(surf,Vec3( 0.5, 0.5,-0.5));
AddVertex(surf,Vec3( 0.5,-0.5,-0.5));

AddVertex(surf,Vec3(-0.5,-0.5, 0.5));
AddVertex(surf,Vec3(-0.5, 0.5, 0.5));
AddVertex(surf,Vec3( 0.5, 0.5, 0.5));
AddVertex(surf,Vec3( 0.5,-0.5, 0.5));

AddVertex(surf,Vec3(-0.5,-0.5, 0.5));
AddVertex(surf,Vec3(-0.5, 0.5, 0.5));
AddVertex(surf,Vec3( 0.5, 0.5, 0.5));
AddVertex(surf,Vec3( 0.5,-0.5, 0.5));

AddVertex(surf,Vec3(-0.5,-0.5,-0.5));
AddVertex(surf,Vec3(-0.5, 0.5,-0.5));
AddVertex(surf,Vec3( 0.5, 0.5,-0.5));
AddVertex(surf,Vec3( 0.5,-0.5,-0.5));

AddVertex(surf,Vec3(-0.5,-0.5, 0.5));
AddVertex(surf,Vec3(-0.5, 0.5, 0.5));
AddVertex(surf,Vec3( 0.5, 0.5, 0.5));
AddVertex(surf,Vec3( 0.5,-0.5, 0.5));

AddVertex(surf,Vec3(-0.5,-0.5,-0.5));
AddVertex(surf,Vec3(-0.5, 0.5,-0.5));
AddVertex(surf,Vec3( 0.5, 0.5,-0.5));
AddVertex(surf,Vec3( 0.5,-0.5,-0.5));

SetVertexNormal(surf,0,Vec3(0.0,0.0,-1.0));

```

```
SetVertexNormal (surf,1,Vec3(0.0,0.0,-1.0));
SetVertexNormal (surf,2,Vec3(0.0,0.0,-1.0));
SetVertexNormal (surf,3,Vec3(0.0,0.0,-1.0));

SetVertexNormal (surf,4,Vec3(0.0,0.0,1.0));
SetVertexNormal (surf,5,Vec3(0.0,0.0,1.0));
SetVertexNormal (surf,6,Vec3(0.0,0.0,1.0));
SetVertexNormal (surf,7,Vec3(0.0,0.0,1.0));

SetVertexNormal (surf,8,Vec3(0.0,-1.0,0.0));
SetVertexNormal (surf,9,Vec3(0.0,1.0,0.0));
SetVertexNormal (surf,10,Vec3(0.0,1.0,0.0));
SetVertexNormal (surf,11,Vec3(0.0,-1.0,0.0));

SetVertexNormal (surf,12,Vec3(0.0,-1.0,0.0));
SetVertexNormal (surf,13,Vec3(0.0,1.0,0.0));
SetVertexNormal (surf,14,Vec3(0.0,1.0,0.0));
SetVertexNormal (surf,15,Vec3(0.0,-1.0,0.0));

SetVertexNormal (surf,16,Vec3(-1.0,0.0,0.0));
SetVertexNormal (surf,17,Vec3(-1.0,0.0,0.0));
SetVertexNormal (surf,18,Vec3(1.0,0.0,0.0));
SetVertexNormal (surf,19,Vec3(1.0,0.0,0.0));

SetVertexNormal (surf,20,Vec3(-1.0,0.0,0.0));
SetVertexNormal (surf,21,Vec3(-1.0,0.0,0.0));
SetVertexNormal (surf,22,Vec3(1.0,0.0,0.0));
SetVertexNormal (surf,23,Vec3(1.0,0.0,0.0));

SetVertexTexCoords (surf,0,Vec2(0.0,1.0));
SetVertexTexCoords (surf,1,Vec2(0.0,0.0));
SetVertexTexCoords (surf,2,Vec2(1.0,0.0));
SetVertexTexCoords (surf,3,Vec2(1.0,1.0));

SetVertexTexCoords (surf,4,Vec2(1.0,1.0));
SetVertexTexCoords (surf,5,Vec2(1.0,0.0));
SetVertexTexCoords (surf,6,Vec2(0.0,0.0));
SetVertexTexCoords (surf,7,Vec2(0.0,1.0));

SetVertexTexCoords (surf,8,Vec2(0.0,1.0));
SetVertexTexCoords (surf,9,Vec2(0.0,0.0));
SetVertexTexCoords (surf,10,Vec2(1.0,0.0));
SetVertexTexCoords (surf,11,Vec2(1.0,1.0));

SetVertexTexCoords (surf,12,Vec2(0.0,0.0));
SetVertexTexCoords (surf,13,Vec2(0.0,1.0));
SetVertexTexCoords (surf,14,Vec2(1.0,1.0));
SetVertexTexCoords (surf,15,Vec2(1.0,0.0));

SetVertexTexCoords (surf,16,Vec2(0.0,1.0));
SetVertexTexCoords (surf,17,Vec2(0.0,0.0));
SetVertexTexCoords (surf,18,Vec2(1.0,0.0));
SetVertexTexCoords (surf,19,Vec2(1.0,1.0));

SetVertexTexCoords (surf,20,Vec2(1.0,1.0));
SetVertexTexCoords (surf,21,Vec2(1.0,0.0));
SetVertexTexCoords (surf,22,Vec2(0.0,0.0));
SetVertexTexCoords (surf,23,Vec2(0.0,1.0));

AddTriangle (surf,0,1,2);
AddTriangle (surf,0,2,3);
AddTriangle (surf,6,5,4);
AddTriangle (surf,7,6,4);
```

```
AddTriangle(surf, 6+8, 5+8, 1+8);
AddTriangle(surf, 2+8, 6+8, 1+8);
AddTriangle(surf, 0+8, 4+8, 7+8);
AddTriangle(surf, 0+8, 7+8, 3+8);
AddTriangle(surf, 6+16, 2+16, 3+16);
AddTriangle(surf, 7+16, 6+16, 3+16);
AddTriangle(surf, 0+16, 1+16, 5+16);
AddTriangle(surf, 0+16, 5+16, 4+16);

UpdateMesh(mesh3);

PositionEntity(mesh3, Vec3(2, 0, 0));
PaintEntity(mesh3, mat);

//Main loop
while(!KeyHit(KEY_ESCAPE)) {

    TurnEntity(mesh1, Vec3(0, 0.5, 0));
    TurnEntity(mesh2, Vec3(0, 0.5, 0));
    TurnEntity(mesh3, Vec3(0, 0.5, 0));

    //Update the world
    UpdateWorld();

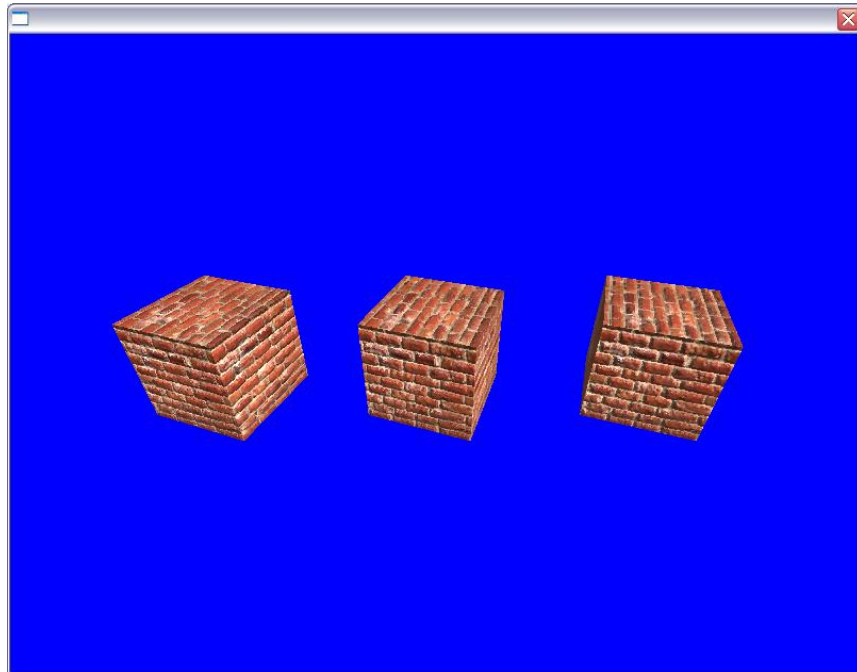
    //Render the scene
    SetBuffer(buffer);
    RenderWorld();

    //Render lighting
    SetBuffer(BackBuffer());
    RenderLights(buffer);

    //Swap the front and back buffer
    Flip();
}

exitapp:
return Terminate();
}
```

Here is the result. The three cubes are identical, but created three different ways. The first is created with a primitive creation function, the second is loaded from a file, and the third is created in code from scratch:



Copyright Leadwerks Software 2008

All Rights Reserved.