

Tutorial 8: Index Buffer

by Victor Saar

Content

This tutorial covers an easy way to optimize a program or make it faster. Index buffers are used to reduce the number of vertices needed to draw an object. You only need each vertex one time. In the index buffer is stored, which vertices of a vertex buffer are used to draw a triangle. So there are always three indices that describe one triangle. The result is, that there must be stored only a small part of the vertices, which were needed without this buffer. Now you have to store three times the indices of the vertices you want to use, but that is not so much memory, because the indices are only `short` values.

The disadvantages of an index buffer you find at lighting and texture mapping. When you use only one vertex for several triangles, you also have only one normal vector and one pair of texture coordinates for it. These problems you can observe when you start the program for this tutorial. You will see, that on three sides of the cube the texture is mapped wrong. Also when you want to use lighting, you will see, that only two sides of the cube are lit correctly, because the normal vector can be right only for one triangle per vertex.

main.cpp

At the beginning we have the two objects we need. A vertex buffer and an index buffer.

```
LPDIRECT3DVERTEXBUFFER9 pCubeVB = NULL; //vertex buffer
LPDIRECT3DINDEXBUFFER9 pCubeIB = NULL; //index buffer (NEW)
```

Here we have the arrays, which include the vertices and the indices we need. We have only eight vertices instead of 36 vertices without index buffer. We save 75% of the vertices. Then we have the indices. Three indices for each triangle. The index buffer consists only of `short` values, because we have only a small number of vertices.

```
D3DVERTEX aCubeVertices[] = {{-1.0f,-1.0f,-1.0f,0.0f,1.0f},
                             {-1.0f, 1.0f,-1.0f,0.0f,0.0f},
                             { 1.0f, 1.0f,-1.0f,1.0f,0.0f},
                             { 1.0f,-1.0f,-1.0f,1.0f,1.0f},
                             {-1.0f,-1.0f, 1.0f,0.0f,0.0f},
                             { 1.0f,-1.0f, 1.0f,0.0f,1.0f},
                             { 1.0f, 1.0f, 1.0f,0.0f,0.0f},
                             {-1.0f, 1.0f, 1.0f,0.0f,1.0f}};

short aCubeIndices[] = {0,1,2, 2,3,0, 4,5,6,
                       6,7,4, 0,3,5, 5,4,0,
                       3,2,6, 6,5,3, 2,1,7,
                       7,6,2, 1,0,4, 4,7,1};
```

We can use the buffers only when we create them first. The index buffers is created similar to the vertex buffer. The only difference is the third parameter. When you create an index buffers you have to select `D3DFMT_INDEX16` for `short` index values or `D3DFMT_INDEX32` for `int` index values. We have to take the first.

```
g_App.GetDevice()->CreateVertexBuffer(sizeof(aCubeVertices),
                                       D3DUSAGE_WRITEONLY,
                                       D3DFVF_CUSTOMVERTEX,
                                       D3DPOOL_MANAGED,
                                       &pCubeVB,
```

```

        NULL);
g_App.GetDevice()->CreateIndexBuffer(sizeof(aCubeIndices),
        D3DUSAGE_WRITEONLY,
        D3DFMT_INDEX16,
        D3DPOOL_MANAGED,
        &pCubeIB,
        NULL);

```

Before we can draw the cube, we have to fill the buffer objects with the information. We lock the buffers and copy the memory used by the arrays into the buffers. Then we unlock them.

```

pCubeVB->Lock(0, sizeof(pData), (void**)&pData, 0); //lock buffer
memcpy(pData, aCubeVertices, sizeof(aCubeVertices)); //copy data
pCubeVB->Unlock(); //unlock buffer

pCubeIB->Lock(0, sizeof(pData), (void**)&pData, 0); //lock buffer
memcpy(pData, aCubeIndices, sizeof(aCubeIndices)); //copy data
pCubeIB->Unlock(); //unlock buffer

```

At the end we have to change the main loop. First we set the vertex stream source to the vertex buffer. Then we have to tell the device, where to take the indices from. The last function draws the indexed vertices. The first parameter says, that we have a triangle list. The second is the minimum vertex index. The third is the number of vertices we want to use. The last parameter is the number of triangle we want to draw.

```

//set stream source
g_App.GetDevice()->SetStreamSource(0, pCubeVB, 0, sizeof(D3DVERTEX));
//set indices (NEW)
g_App.GetDevice()->SetIndices(pCubeIB);
//draw triangles (NEW)
g_App.GetDevice()->DrawIndexedPrimitive(D3DPT_TRIANGLELIST, 0, 0, 8, 0, 12);

```

This is all about index buffers. Before you use them, make sure that they are really useful for you. When you want to use them, check out the program Crossroads in the download section. With it you can export different file formats to C++ source code.