02564, REAL-TIME GRAPHICS - Exercice 2 Vertex Buffer Objects and Frame Buffer Objects

OLIVIER ROUILLER - S090842
Department of Informatics and Mathematical Modelling
Technical University of Denemark

February 15, 2010

1 Part 1

We change the program to draw the terrain in a single triangle strip and using a vertex buffer object. In the following code, we create a vertex array and a normal array, fill them with the terrain vertices and normals and taking care of the order of the vertives in order to be able to render it in a single strip.

- To draw the terrain in one strip we have to repeat the last vertex and the first vertex of each row.
- A vertex array object is used to keep track of a buffer object.

```
static GLuint buffers[2];
static bool buff_obj=false;
static int num_vertices;
if(!buff_obj){
    buff_obj = true;
    glGenBuffers(2, buffers);
    vector<Vec3f> vertices;
    vector<Vec3f> normals;

    // Preparing the datas to copy in the vertex array
    for(int j=-30;j<30;++j){
        // repeating first vertex of the row
        vertices.push_back(Vec3f(-30,j+1,terra.height(-30,j+1)));
        normals.push_back(terra.normal(-30,j+1));

        for(int i=-30;i<30;++i){
            normals.push_back(terra.normal(i,j+1));
            vertices.push_back(Vec3f(i,j+1,terra.height(i,j+1)));
            normals.push_back(terra.normal(i,j));
            vertices.push_back(Vec3f(i,j,terra.height(i,j)));
        }
        // repeating last vertex of the row
        vertices.push_back(Vec3f(29,j,terra.height(29,j)));
        normals.push_back(terra.normal(29,j));
    }
    num_vertices = vertices.size();

    // setting up the vertex buffer and copying datas
    glBindBuffer(GL_ARRAY_BUFFER, buffers[0]);
    glBufferData(GL_ARRAY_BUFFER, vertices.size()*sizeof(Vec3f),
        &vertices[0], GL_STATIC_DRAW);
    glVertexPointer(3, GL_FLOAT, 0, 0);
    // setting up the normal buffer and copying datas
    glBindBuffer(GL_ARRAY_BUFFER, buffers[1]);
    glBufferData(GL_ARRAY_BUFFER, normals.size()*sizeof(Vec3f),
        &normals[0], GL_STATIC_DRAW);
    glNormalPointer(GL_FLOAT, 0, 0);
}
```

We then render the terrain by calling a triangle strip over the buffer objects in the following way :

```
glEnableClientState(GL_VERTEX_ARRAY);
glEnableClientState(GL_NORMAL_ARRAY);
glDrawArrays(GL_TRIANGLE_STRIP, 0, num_vertices);
glDisableClientState(GL_VERTEX_ARRAY);
glDisableClientState(GL_NORMAL_ARRAY);
```

2 Part 2

The timing of the rendering using a display list and a buffer object gives almost the same result. With a buffer object, the average framerate is 1224.53 fps against 1214.12 fps with a display list on the VR-databar hardware. In conclusion it seems using a display list seems to be as fast as using a vertex buffer object. We could then prefer to use it for a static model because of the simplicity of the setup.

3 Part 3

Finaly we use a frame buffer object to render the view of the user in a texture and then display it on a bilboard (3.1). The code use to do so is in most part the code showed in the slides.

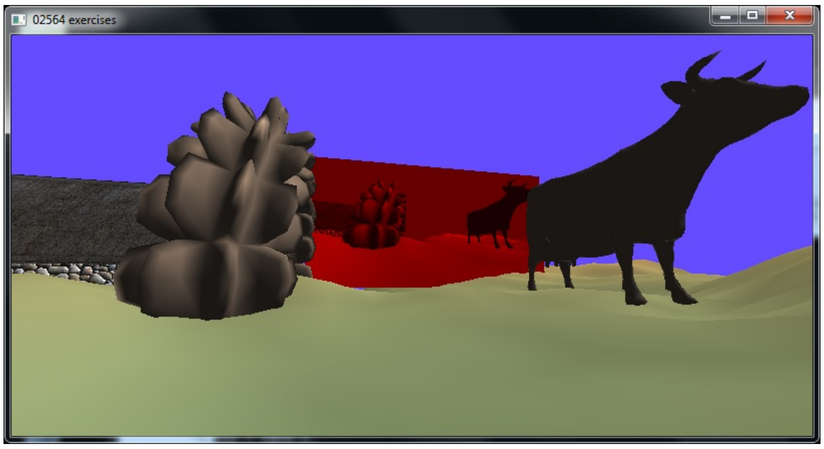

Figure 3.1: A bilboard rendered with a frame buffer object

As a funny extension we use a additional frame buffer object to create an effect of infinite dephth (3.2). The process used above is repeated by switching of buffer at every rendering.

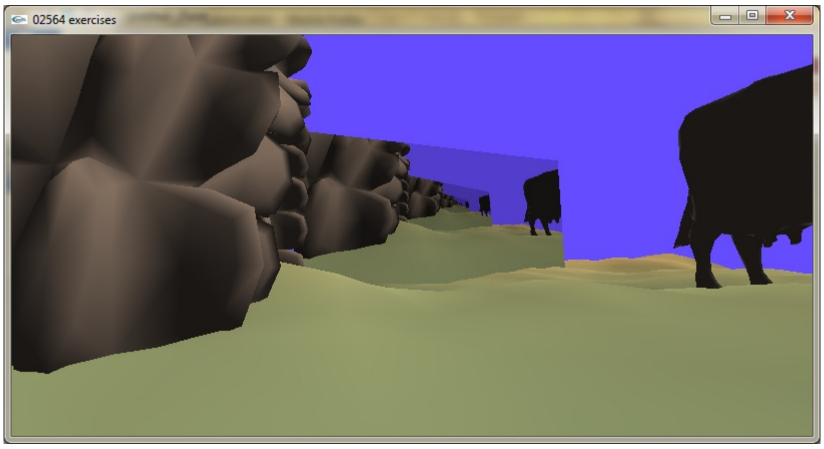

Figure 3.2: A funny extension