sign up log in tour help

x Dismiss

Join the Stack Overflow Community

Stack Overflow is a community of 6.3 million programmers, just like you, helping each other.
Join them; it only takes a minute:

Sign up

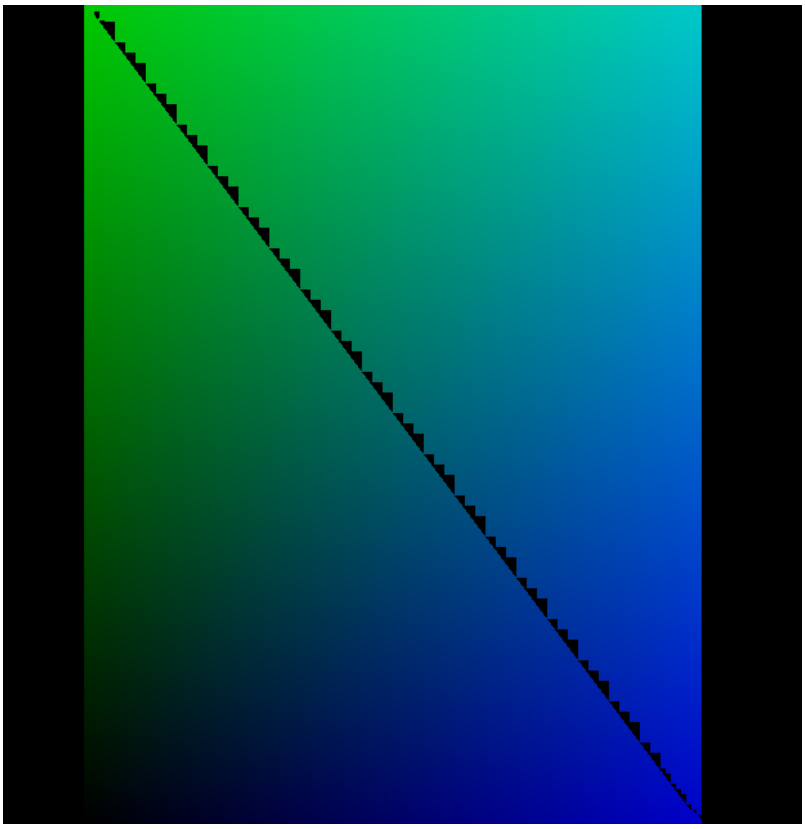
Rendering a dynamic cubemap (OpenGL)

I'm trying to render a scene 6 times and put them on the sides of a cubemap. I'd like to do this properly first before moving onto learning Geometry shaders which would allow this to be done in one pass. Here goes the code:

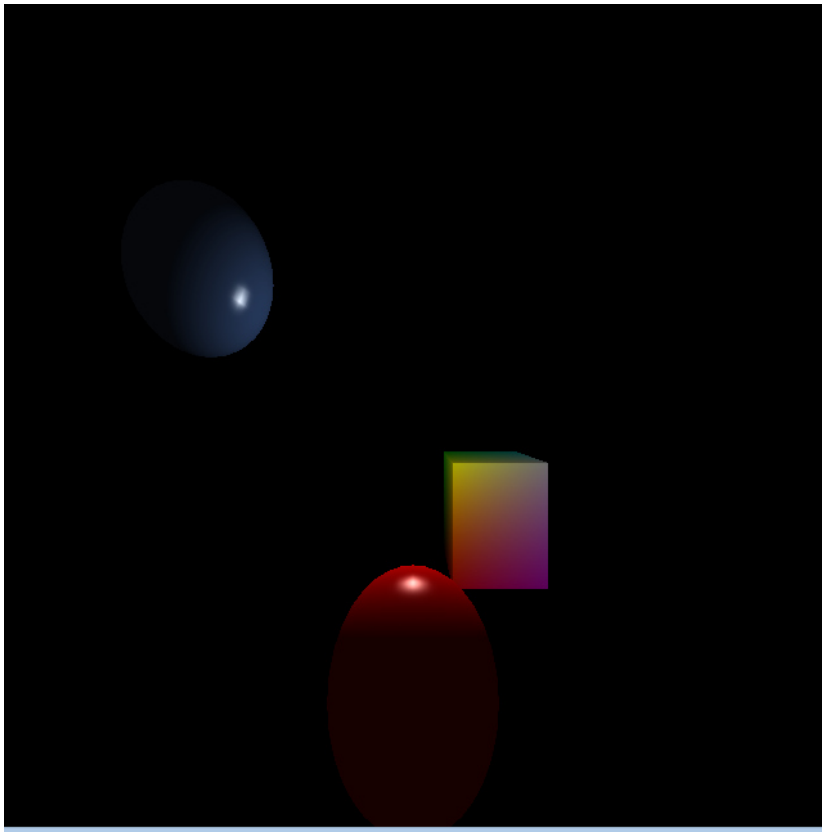
```
    void Scene::setupFBO()
    {
        glGenTextures(1, &cubemap);
        glBindTexture(GL_TEXTURE_CUBE_MAP,cubemap);
        glActiveTexture(GL_TEXTURE0);
        const int size = 128;
            // create the fbo
        glGenFramebuffers(1, &fbo);
        glBindFramebuffer(GL_FRAMEBUFFER, fbo);

        for(int i=0;i<6;i++)
        {
            glTexImage2D(GL_TEXTURE_CUBE_MAP_POSITIVE_X + i, 0, GL_RGB,
                size, size, 0, GL_RGB, GL_UNSIGNED_BYTE, 0);

        }
        glTexParameterf(GL_TEXTURE_CUBE_MAP, GL_TEXTURE_MAG_FILTER,
        GL_LINEAR);
        glTexParameterf(GL_TEXTURE_CUBE_MAP, GL_TEXTURE_MIN_FILTER,
        GL_LINEAR);
        glTexParameterf(GL_TEXTURE_CUBE_MAP, GL_TEXTURE_WRAP_S,
        GL_CLAMP_TO_EDGE);
        glTexParameterf(GL_TEXTURE_CUBE_MAP, GL_TEXTURE_WRAP_T,
        GL_CLAMP_TO_EDGE);
        glTexParameterf(GL_TEXTURE_CUBE_MAP, GL_TEXTURE_WRAP_R,
        GL_CLAMP_TO_EDGE);

        // create the uniform depth buffer
        glGenRenderbuffers(1, &depthbuff);
        glBindRenderbuffer(GL_RENDERBUFFER, depthbuff);
        glRenderbufferStorage(GL_RENDERBUFFER, GL_DEPTH_COMPONENT, size, size);
        //glBindRenderbuffer(GL_RENDERBUFFER, 0);
        GLenum drawBufs[] = {GL_COLOR_ATTACHMENT0};
        // attach it
        glFramebufferRenderbuffer(GL_FRAMEBUFFER, GL_DEPTH_ATTACHMENT, GL_RENDERBUFFER,
fbo);
        //glFramebufferTexture2D(GL_FRAMEBUFFER, GL_COLOR_ATTACHMENT0,
GL_TEXTURE_CUBE_MAP_POSITIVE_X, cubemap, 0);
        glDrawBuffers(1, drawBufs);

        glBindFramebuffer(GL_FRAMEBUFFER, 0);
        glBindTexture(GL_TEXTURE_CUBE_MAP, 0);

    }



    void Scene::pass1()
    {
        GLuint p = glGetSubroutineIndex(program->id,GL_FRAGMENT_SHADER,"pass1");
        glUniformSubroutinesuiv(GL_FRAGMENT_SHADER,1,&p);

        mat4 view;
        glBindFramebuffer(GL_FRAMEBUFFER, fbo);
        for(int i=0;i<6;i++)
        {
            glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
            glFramebufferTexture2D(GL_FRAMEBUFFER, GL_COLOR_ATTACHMENT0,
                GL_TEXTURE_CUBE_MAP_POSITIVE_X + i, cubemap, 0);
            if(i==0) //X+
                cam->Update(vec3(0),vec3(10,0,0)); // position, target
            else if(i==1) //X-
                cam->Update(vec3(0),vec3(-10,0,0));
            else if(i==2) //Y+
                cam->Update(vec3(0),vec3(0,10,0));
```

```
            else if(i == 3) //....
                cam->Update(vec3(0),vec3(0,-10,0));
            else if(i == 4)
                cam->Update(vec3(0),vec3(0,0,10));
            else if(i == 5)
                cam->Update(vec3(0),vec3(0,0,-10));

            view = cam->getViewMat();
            for(int ii=1;ii<SHAPE_COUNT;ii++){
                shapes[ii]->setViewMat(view);
                shapes[ii]->Draw();
            }

        }
    }


    void Scene::pass2()
    {
        GLuint p = glGetSubroutineIndex(program->id,GL_FRAGMENT_SHADER,"pass2");
        glUniformSubroutinesuiv(GL_FRAGMENT_SHADER,1,&p);
        glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
        glBindFramebuffer(GL_FRAMEBUFFER, 0);
        cam->Update(vec3(0,0,10),vec3(0));
        mat4 view = cam->getViewMat();
        for(int i=0;i<SHAPE_COUNT;i++){
            shapes[i]->setViewMat(view);
            (*shapes[i]).Draw();
        }
    }


void Scene::Draw(){
    pass1(); // Create the cubemap
    pass2(); // Draw the scene normally
}
```

I've modified the code from my previous project that implemented a static cubemap that was used for IBL and reflection/refraction. I'll provide the shaders codes if you think it's necessary. At this stage there's only gibberish being rendered.

End result 

The actual scene



Update The big square was the result of a bug in my setViewMat function which didn't apply the stacked transformations. It now only renders the scene, the texture is black. I've used AMD gDEBugger to see the resulting cubemap which is just black. So I think it's either my initial fbo binding or how I render each side in the first pass that's at fault.

c++ opengl rendering fbo skybox

edited Mar 5 '15 at 17:49

asked Mar 4 '15 at 1:50

Esmail
20 7

What sort of gibberish? Can you provide a screenshot? – Simon McKenzie Mar 4 '15 at 1:52

Impressive response time, I tried but I don't have 10 rep points to upload images. – Esmail Mar 4 '15 at 2:07

:) If you upload the image to imgur and put the link into your question, that will be sufficient. – Simon McKenzie Mar 4 '15 at 2:09

It's up, I had made pictures for each pass to show they each work separately but again had reputation issues with posting more than 2 links. – Esmail Mar 4 '15 at 2:39

1 Answer

OK, I finally got it working. In setupFBO I shouldn't have unbound the cubemap. I've posted the updated pass1,2 code anyways. Though the rendering works I have trouble ordering the sides as it seems the sides are not mapped as I had anticipated, X+, X- , ...

```
void Scene::pass1()
 {
   glBindFramebuffer(GL_FRAMEBUFFER, fbo);
   GLuint p = glGetSubroutineIndex(program->id,GL_FRAGMENT_SHADER,"pass1");
   glUniformSubroutinesuiv(GL_FRAGMENT_SHADER,1,&p);
   glViewport(0,0,512,512);
   mat4 view, proj;
   proj = glm::perspective(90.0f, 1.0f, 1.0f, 500.0f);
   for(int i=0;i<6;i++)
   {
       glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
      if(i==0)
         view = Camera::Update(vec3(0),vec3(1,0,0),vec3(0,1,0));// pos, target, up
      else if(i==1)
         view =  Camera::Update(vec3(0),vec3(-1,0,0),vec3(0,1,0));
      else if(i==2)
        view =  Camera::Update(vec3(0),vec3(0,1,0),vec3(0,0,1));
      else if(i == 3)
        view =  Camera::Update(vec3(0),vec3(0,-1,0),vec3(0,0,-1));
      else if(i == 4)
        view =  Camera::Update(vec3(0),vec3(0,0,1),vec3(0,1,0));
      else if(i == 5)
        view =  Camera::Update(vec3(0),vec3(0,0,-1),vec3(0,1,0));
       for(int ii=1;ii<SHAPE_COUNT;ii++){
```

```
            shapes[ii]->setProjMat(proj);
            shapes[ii]->setViewMatAndUpdate(view); // Empties the transformation stack,
pushes the new view matrix and applies all the transformations again
            shapes[ii]->Draw();
                    //Reverts back to the original stack
            shapes[ii]->setViewMatAndUpdate(cam->getViewMat());
        }
        glFramebufferTexture2D(GL_FRAMEBUFFER, GL_COLOR_ATTACHMENT0,
            GL_TEXTURE_CUBE_MAP_POSITIVE_X + i, cubemap,0);
     }
   }


void Scene::pass2()
{
    glBindFramebuffer(GL_FRAMEBUFFER, 0);
    glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
    GLuint p = glGetSubroutineIndex(program->id,GL_FRAGMENT_SHADER,"pass2");
    glUniformSubroutinesuiv(GL_FRAGMENT_SHADER,1,&p);
    glViewport(0,0,Constants::Instance()->gWidth,Constants::Instance()->gHeight);
    float  aspectRatio = 8.0f/6.0f;
    mat4 proj = cam->getProjMat();
    for(int i=0;i<SHAPE_COUNT;i++){
        shapes[i]->setProjMat(proj);
        (*shapes[i]).Draw();
    }
}
```

OK, I figured out the ordering by color coding each side of the cube, not sure why it is this way.

```
if(i==1)
    view = Camera::Update(vec3(0),vec3(1,0,0),vec3(0,1,0));
else if(i==2)
    view =  Camera::Update(vec3(0),vec3(-1,0,0),vec3(0,1,0));
else if(i==4)
    view =  Camera::Update(vec3(0),vec3(0,1,0),vec3(0,0,1));
else if(i == 3)
    view =  Camera::Update(vec3(0),vec3(0,-1,0),vec3(0,0,-1));
else if(i == 0)
    view =  Camera::Update(vec3(0),vec3(0,0,1),vec3(0,1,0));
else if(i == 5)
    view =  Camera::Update(vec3(0),vec3(0,0,-1),vec3(0,1,0));
```

edited Mar 8 '15 at 2:09

answered Mar 6 '15 at 20:09

Esmail

20 7