King Abdullah University of Science and Technology

Fall 2015 CS 247 Scientific Visualization Assignment 3

Gang Liao * ID: 133267

Sunday 25th October, 2015

1 Render front-faces and back-faces

First, front faces and back faces should be rendered into texture buffers respectively. **GL_CULL_FACE** can be used to discard front or back based on argument **GL_FRONT** and **GL_BACK**.

```
void RenderBackFaces(float dim_x, float dim_y, float dim_z)
{
        enableRenderToBuffer(backface_buffer);
        // ============================================================
        // TODO: render your backfaces here
        // result gets piped to backface_buffer automatically
        // ============================================================
        glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
        glEnable(GL_CULL_FACE);
        glCullFace(GL_FRONT);
        drawQuads(dim_x, dim_y, dim_z);
        glDisable(GL_CULL_FACE);
        disableRenderToBuffer();
}
```

*Extreme Computing Research Center, Department of Computer Science, King Abdullah University of Science and Technology (KAUST). Email: liao.gang@kaust.edu.sa

2 Direct Volume Rendering

1. Opacity Correction

Since the sample rate may be changed, opacity correction equation can guarantee the correctness of opacity.

```
//Fragment shader
//opacity correction
scalar = 1 - pow((1 - scalar), step_size / base_distance);
```

2. Central Differences

In order to using Phong shading model, normal vector is necessary. Central Difference is one of the general method to compute normal vector.

```
//Fragment shader
vec3 norm;
norm.x = texture3D(vol_texture, pos + half3(delta, 0.0, 0.0)).x - texture3D
    (vol_texture, pos - half3(delta, 0.0, 0.0)).x;
norm.y = texture3D(vol_texture, pos + half3(0.0, delta, 0.0)).x - texture3D
    (vol_texture, pos - half3(0.0, delta, 0.0)).x;
norm.z = texture3D(vol_texture, pos + half3(0.0, 0.0, delta)).x - texture3D
    (vol_texture, pos - half3(0.0, 0.0, delta)).x;
norm = normalize(norm);
```

3. Early Ray Termination

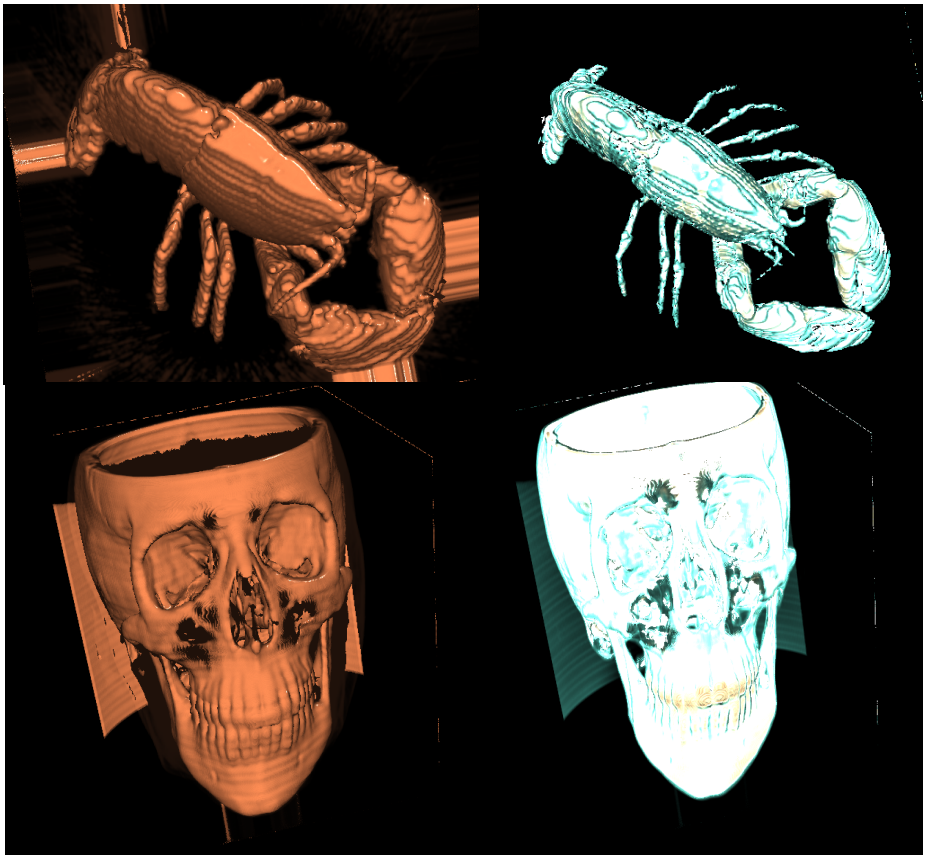
Since computing the exit of ray direction on the cube is time-consuming and quite trick, we can terminate the iteration when opacity achieve the expected value, that is early ray termination.

```
//ray termination: test if outside volume
//early ray termination
if (dst.a > 0.95)
        break;
```

3 Interactive windowing transfer function

When you increase or decrease the maximum and minimum size of window, the transfer function will be updated continuously.

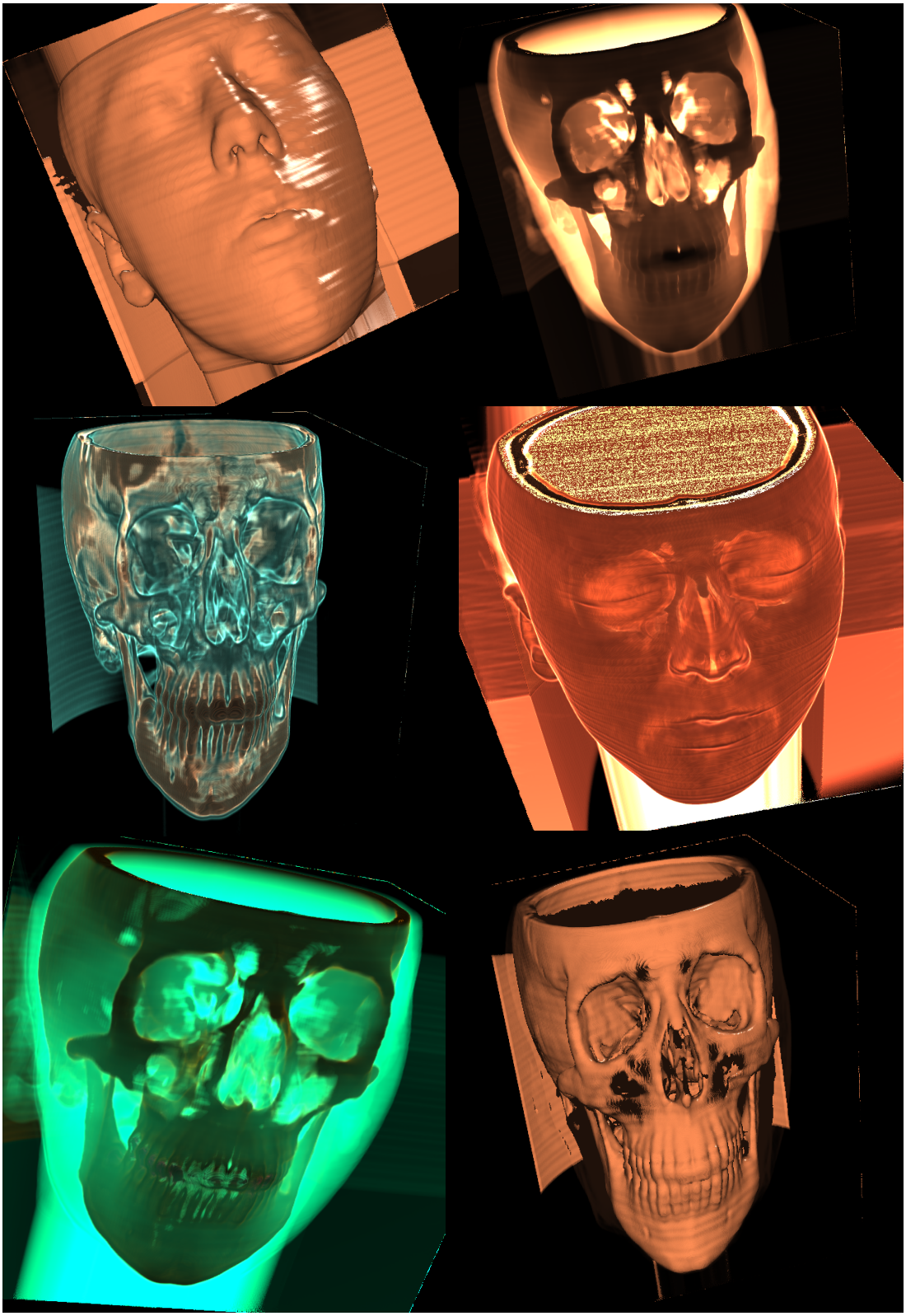
```
for (int i = 0; i < tf_win_min; i++)
{
        tf0[i * 4 + 0] = 0.0f;  tf0[i * 4 + 1] = 0.0f;
        tf0[i * 4 + 2] = 0.0f;    tf0[i * 4 + 3] = 0.0f;
}
float len = tf_win_max - tf_win_min;
for (int i = 0; i < len; i++)
{
    tf0[(i+tf_win_min) * 4 + 0] = (float)(i) / len;tf0[(i+tf_win_min) * 4 +
        1] = (float)(i) / len;
    tf0[(i+tf_win_min) * 4 + 2] = (float)(i) / len;tf0[(i+tf_win_min) * 4 +
        3] = (float)(i) / len;
}
for (int i = tf_win_max; i < 128; i++)
{
        tf0[i * 4 + 0] = 1.0f;  tf0[i * 4 + 1] = 1.0f;
        tf0[i * 4 + 2] = 1.0f;  tf0[i * 4 + 3] = 1.0f;
}
```



4 Iso-surface rendering

When you choose Iso-surface rendering, you need to track the ray direction to find the first hit postion and composite the color and opacity from hit position.

```
//data access to scalar value in 3D valume texture
vec4 value = texture3D(vol_texture, position);
scalar = value.a;
if (scalar >= iso_value && !flag)
{
        flag = 1;
        hit_pos = position;
}
```

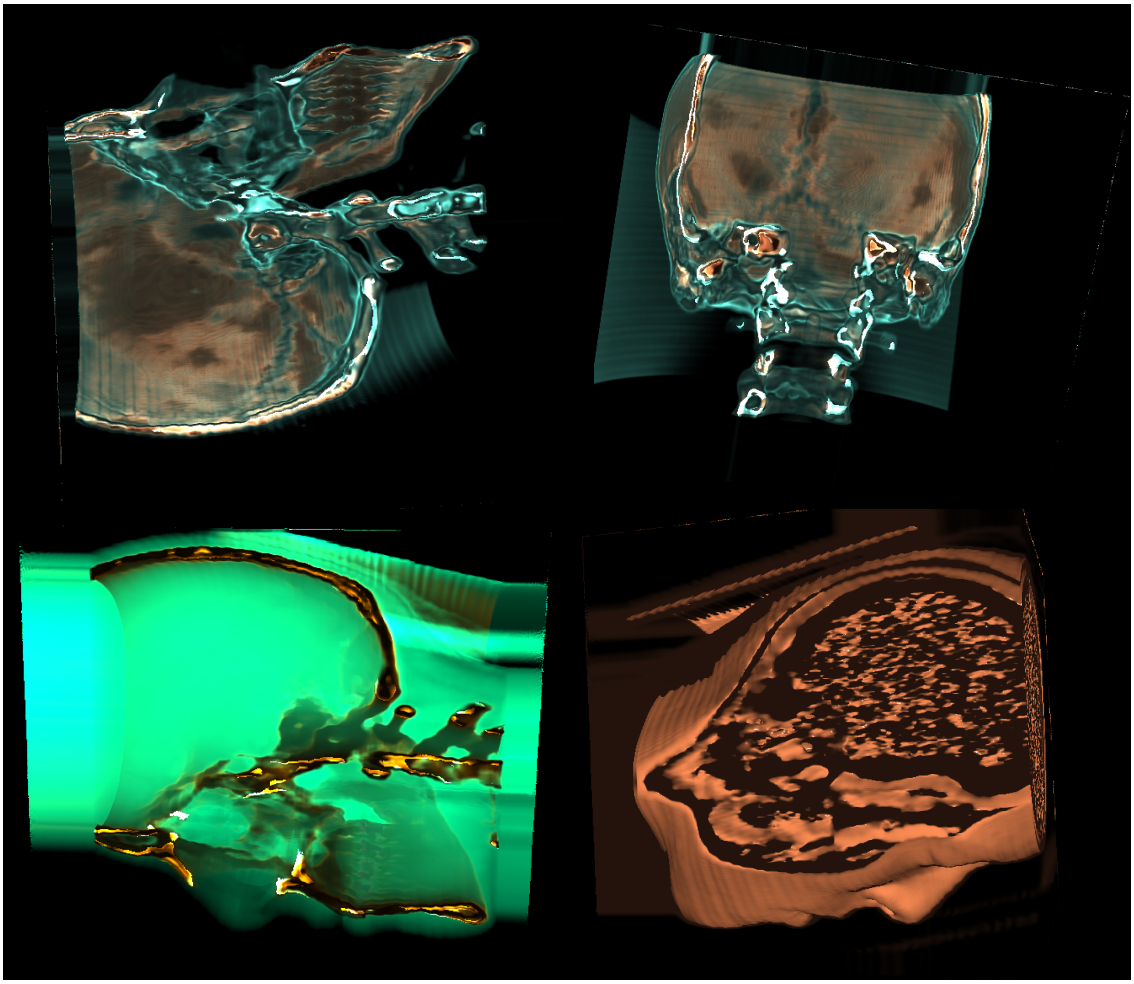


5 Bonus

1. Axis-aligned Clipping planes

In order to clip plane, I don't track the ray direction from the entry position, instead using modified position in the plane.

```
//clip plane
position = position + 200 * raydir * step_size;
```



2. maximum Intensity Projection

Instead of transfer integration, maximum intensity projection means only choose the maximum intensity along each ray direction.

```
//Maximum intensity projection
if (scalar > max_scalar)
{
        MIP = value;
        max_scalar = scalar;
}
...
gl_FragColor = (vec4(ambi + diffuse + spec, 1.0)) * MIP;
```