Information Visualization

W14: Exercise - Implementation of Volume Rendering 1

Graduation School of System Informatics

Department of Computational Science

Naohisa Sakamoto, Akira Kageyama

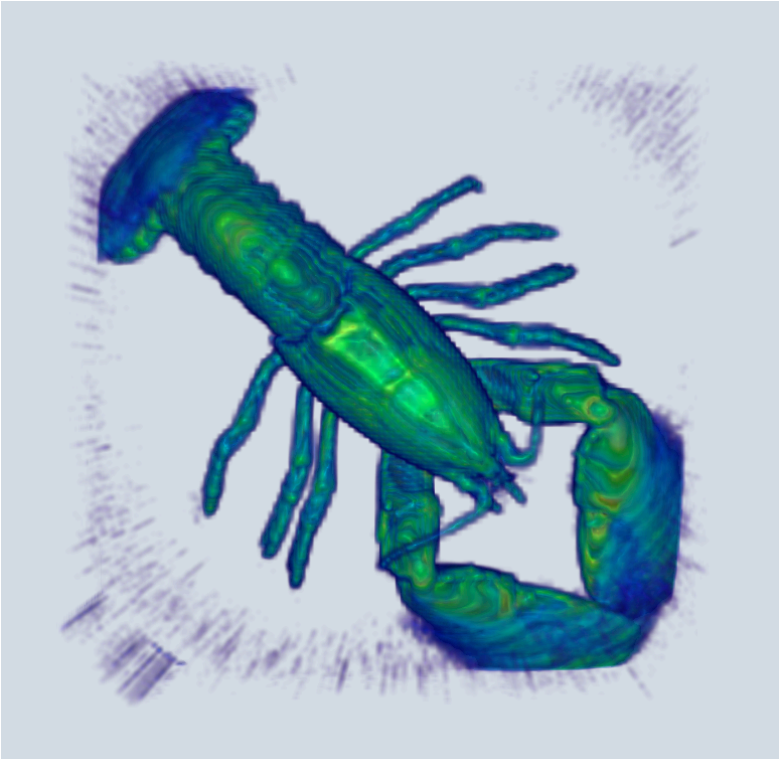
May. 31, 2017

Schedule

- W01 4/11 Guidance
- W02 4/12 Setup
- W03 4/18 Introduction to Data Visualization
- W04 4/19 CG Programming
- W05 4/25 Rendering Pipeline
- W06 4/26 Coordinate Systems and Transformations
- W07 5/09 Shading
- W08 5/10 Shader Programming
- W09 5/16 Visualization Pipeline
- W10 5/17 Data Model and Transfer Function
- W11 5/23 Scalar Data Visualization 1 (Isosurface Extraction)
- W12 5/24 Implementation of Isosurface Extraction
- W13 5/30 Scalar Data Visualization 2 (Volume Rendering)
- W14 5/31 Implementation of Volume Rendering 1
- W15 6/06 Implementation of Volume Rendering 2

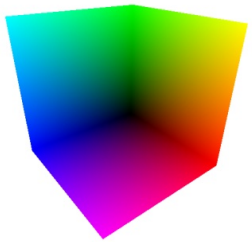
Ex01: Ray-casting Rendering

- Rendering a volume data by ray-casting approach.
  - Download
    - w14_main_ex01.js
    - w14_index_ex01.html
    - three.min.js
    - TrackballControl.js
  - Open
    - w14_index_ex01.html

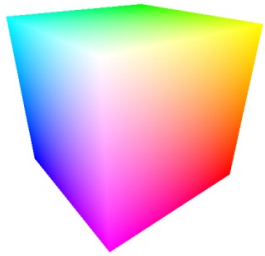


Ex01: Simple Ray-casting

- Two-pass rendering
  - First pass
    - Render the back faces of the bounding box to the off-screen buffer



  - Second pass
    - Render the front faces of the bounding box
    - Perform the ray-casting iterations
    - Draw the final image to the screen



Ex01: Simple Ray-casting

- First pass
  - Render target for the off-screen rendering

```
var exit_texture = new THREE.WebGLRenderTarget(
    screen.width, screen.height,
    {
        minFilter: THREE.LinearFilter,
        magFilter: THREE.LinearFilter,
        wrapS: THREE.ClampToEdgeWrapping,
        wrapT: THREE.ClampToEdgeWrapping,
        format: THREE.RGBFormat,
        type: THREE.FloatType,
        generateMipmaps: false
    }
);
```

Ex01: Simple Ray-casting

- First pass
  - Scene for the off-screen rendering

```
var exit_buffer = new THREE.Scene();
```

  - Off-screen rendering

```
function render() {
    ...
    renderer.render( exit_buffer, screen.camera, exit_texture, true);
    ...
}
```

Ex01: Simple Ray-casting

- First pass
  - ShaderMaterial for the off-screen rendering

```
var bounding_material= new THREE.ShaderMaterial( {
     vertexShader: document.getElementById(‘bounding.vert').textContent;
     fragmentShader: document.getElementById(‘bounding.frag').textContent,
     side: THREE.BackSide
});
```

Ex01: Simple Ray-casting

- First pass
  - Mesh for the off-screen rendering

```
var bounding_geometry = BoundingBoxGeometry( volume );

var bounding_mesh = new THREE.Mesh( bounding_geometry, bounding_material );
```

  - Add the mesh to the scene

```
exit_buffer.add( bounding_mesh );
```

Ex01: Simple Ray-casting

- Second pass
  - ShaderMaterial for the screen

```
var raycaster_metrial = new THREE.ShaderMaterial( {
     vertexShader: document.getElementById( ‘raycaster.vert' ).textContent,
     fragmentShader: document.getElementById( ‘raycaster.frag' ).textContent,
     side: THREE.FrontSide,
     uniforms: {
          volume_resolution:  { type: "v3", value: volume.resolution },
          exit_point: { type: "t", value: exit_texture },
          volume_data: { type: "t", value: volume_texture },
          transfer_function_data:{type: "t", value: transfer_function_texture},
          light_position: {type: “v3", value: screen.light.position }
          camera_position: {type: “v3", value: screen.camera.position }
          background_color: {type: “v3“, value: new THREE.Vector3().fromArray(
screen.renderer.getClearColor().toArray() )
     }
});
```

Ex01: Simple Ray-casting

- Second pass
  - Scene for the on-screen rendering

```
screen.scene
```

  - On-screen rendering

```
function render() {
    ...
    renderer.render( screen.scene, screen.camera );
    ...
}
```

Ex01: Simple Ray-casting

- Second pass
  - Mesh for the on-screen rendering

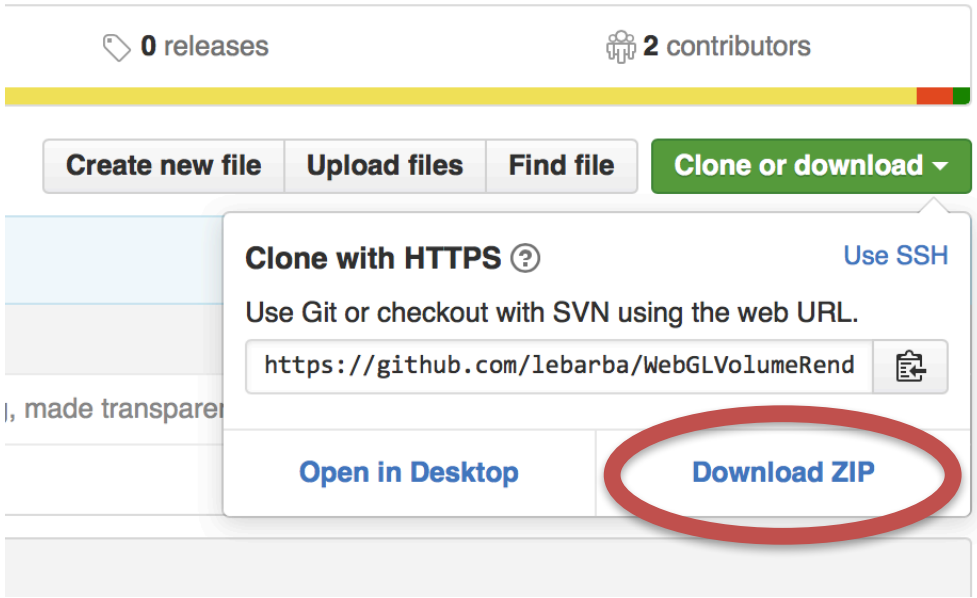
```
var raycaster_mesh = new THREE.Mesh( bounding_geometry, raycaster_material );
```

  - Add the mesh to the scene

```
screen.scene.add( raycaster_mesh );
```

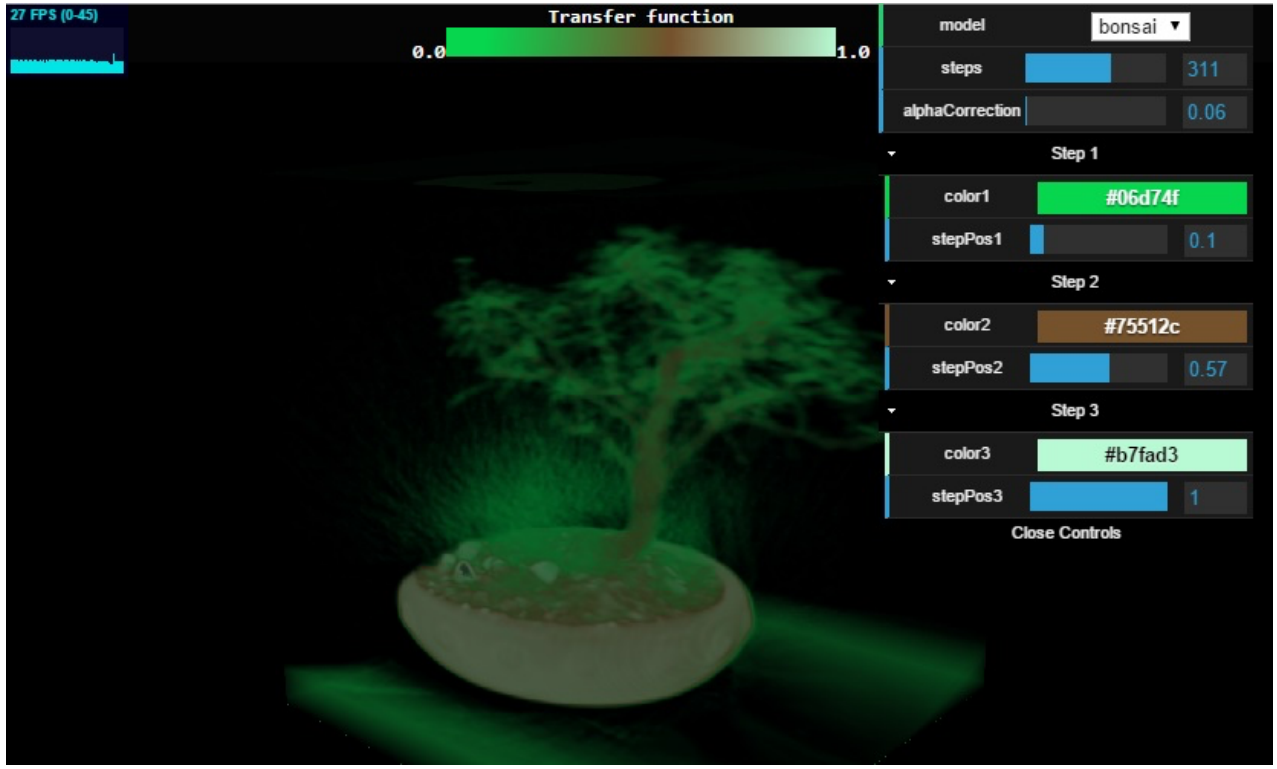
Ex02: Ray-casting with Simple UI

- Rendering a volume data by ray-casting approach.
  - WebGL Volume Rendering
    https://github.com/lebarba/WebGLVolumeRendering
  - Download
    - Clone or download
    - Download ZIP
  - Open
    - Web/Index.html



Ex02: Ray-casting with Simple UI

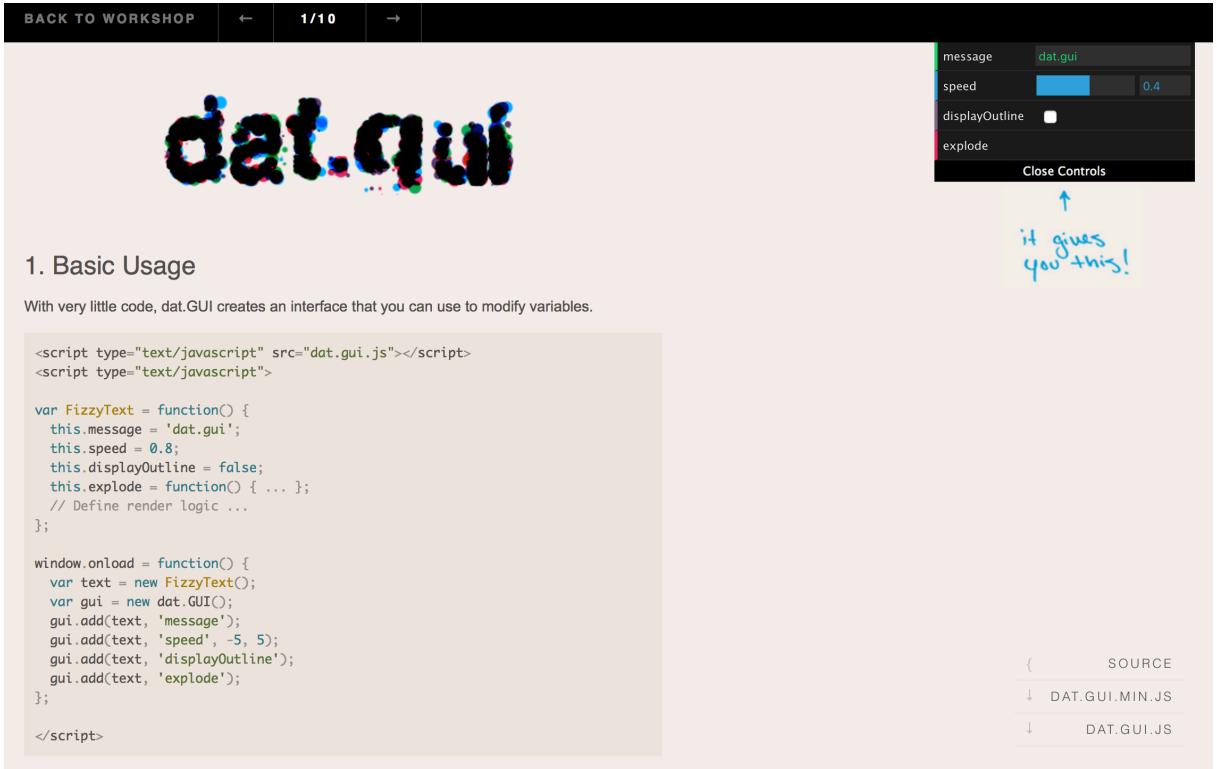
- Rendering a volume data by ray-casting approach.



Ex02: Ray-casting with Simple UI

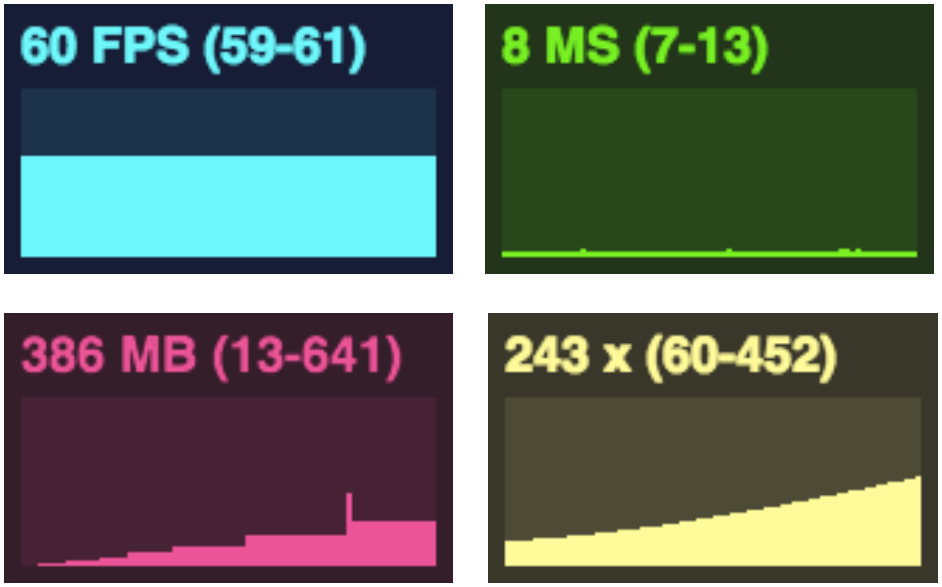
- User Interface
  - dat.UI

    https://workshop.chromeexperiments.com/examples/gui/#1--Basic-Usage



Ex02: Ray-casting with Simple UI

- User Interface
  - stats.js
    https://github.com/mrdoob/stats.js/



Polling

- Take the poll
  - Student ID Number
  - Name