Medical Image Display
-- GPU Volume Rendering
Yun Teng & Yi Gong
Medical 3D Image Visualization

- Generate a data volume from 2D data slices
- Use rendering techniques to visualize data volume as 3D image.
- Resliceing from the 3D volume
Demo

http://www.youtube.com/watch?v=fJmZTREWbsY
Rendering Techniques

- Multi-planar rendering (MPR)
- Surface Rendering (SR)
- Volume Rendering (VR)
Rendering Techniques

- Multi-planar rendering (MPR)
  - Volume is originally built by stacking the axial slices
  - Cut slices through the volume in a different plane
  - Does not require too many calculations
  - Not really 3D
Rendering Techniques

- Surface Rendering (SR)
  - Visualize a 3D object as a set of surfaces called iso-surfaces (same intensity)
  - Advantages:
    - Once and for all computation
  - Disadvantages:
    - Only see one layer
    - Not applicable when the intensities for different materials are close
Rendering Techniques

● Volume Rendering (VR)
  ● Advantages:
    ● Preserve the integrity of the original data
    ● Give user more intuitive 3D visual results with inside contents
    ● Highly parallelable
  ● Disadvantages:
    ● High cost of computation
Volume Rendering Algorithm

- Ray-casting
  - Record each viewing ray’s front and back intersections with bounding cube by rendering the cube faces into textures
  - Along each ray direction, sample volume textures with equal intervals and integrate the luminance with transparency

Textures that record viewing ray intersections with cube’s front (left) & back (right) surfaces
Ray Casting (cont.)

* Rays are casted through the volume and is sampled along equally spaced intervals
* For a single voxel along a ray, the transparency formula is:

\[
\begin{align*}
C_{\text{out}.a} &= \text{decayFactor} \\
C_{\text{in}.\text{rgb}} &= C_{\text{in}.a} \\
C_{\text{out}} &= (1.0 - C_{\text{out}.a}) * C_{\text{in}} + C_{\text{out}}
\end{align*}
\]

$C_{\text{out}}$ is the outgoing intensity/color for voxel along the ray

$C_{\text{in}}$ is the incoming intensity for the voxel
Limitations of mobile GPU

- 3D textures are not fully implemented in OpenGL ES EXT
  - Flatten 3D textures to 2D ones and sample them with offset calculated based on z component value
- Limit of texture size are relative small compared with PC
  - Fully use the RGBA channels of one pixel
Transfer Functions

* Assign RGB and alpha values for every voxel in the volume
* 1D transfer function maps one RGBA value for every isovalue \([0,255]\)
Transfer Function UI

- User choose up to 5 control points (can be 0)
- Spline interpolation to get a smooth transition
- User can save and load desired profiles
Transfer Function UI (cont.)

Profile 140 94

Profile 86
Optimization

- Performance without acceleration
  - Without color transfer: 660~1000 ms
  - With color transfer: 780~1400 ms
- Can’t achieve real-time interaction
- If most part on a ray is black(empty), lots of iterations are wasted
Optimization (cont.)

- Empty Space Skipping
  - Instead of drawing one single cube’s front & back surfaces to record viewing ray intersections, break down into small cubes and only draw those that wrap up the content
## Optimization (cont.)

<table>
<thead>
<tr>
<th></th>
<th>No transfer</th>
<th>transfer</th>
<th>Draw cube</th>
</tr>
</thead>
<tbody>
<tr>
<td>No optimization</td>
<td>660~1200ms</td>
<td>780~1400ms</td>
<td>40ms</td>
</tr>
<tr>
<td>Empty space skipping</td>
<td>Around 420ms</td>
<td>Around 550ms</td>
<td>110ms</td>
</tr>
</tbody>
</table>

- **Number of iteration:**
  - no optimization: dynamically calculated, up to 100
  - Empty space skipping: fixed, 35
Reslicing & More UI
Thank you!