3D Graphics without Plugins using SVG

Paungkaew Sangtrakulcharoen
Department of Computer Science
San Jose State University
San Jose, CA 95192-0103
Tel. 408-223-1980
paungkaew@hotmail.com

Advisor: Dr. Chris Pollett
Committee members: Dr. Cay Horstmann
Dr. Ho Kuen Ng
Outline

• Introduction
• Requirements
• Project Design & Implementation
• Optimization
• Conclusion
Introduction

X3D

Browsers with Plugins

Browsers without Plugins by SVG

X3D
Introduction (cont.)

X3D document

<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE X3D SYSTEM "latest.dtd">
<X3D>
  <Scene>
    <Transform>
      <Shape>
        <Appearance>
          <Material diffuseColor="1 0 0"/>
        </Appearance>
        <Box size="1.0 1.0 1.0"/>
      </Shape>
    </Transform>
  </Scene>
</X3D>

X3D Browser
Introduction (cont.)

SVG document

```xml
<?xml version="1.0"?>
<svg xmlns="http://www.w3.org/2000/svg">
  <g transform="scale(0.5)">
    <polygon style="fill:lime; stroke:blue; stroke-width:10"
             points="350,75 458,137.5
                    458,262.5 350,325 242,262.6
                    242,137.5"/>
  </g>
</svg>
```

Mozilla-SVG browser
Introduction (cont.)

X3D source data

```xml
<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE X3D SYSTEM "latest.dtd"/>
<X3D>
  <Scene>
    <Transform translation="-2 -1 0">
      <Shape> <Sphere radius="2"/> </Shape>
    </Transform>
    <Transform translation="3 0.15 0">
      <Shape> <Cone bottomRadius="2" height="2"/> </Shape>
    </Transform>
    <Transform translation="-2 3 0">
      <Shape> <Box size="1.5 1.5 1.5"/> </Shape>
    </Transform>
    <Transform translation="3 3 0">
      <Shape> <Cylinder radius="1.5" height="2.5"/> </Shape>
    </Transform>
  </Scene>
</X3D>
```

The output document

Buttons for changing perspectives
Requirements

- **DTD file**
  - A source X3D file must be validated with X3DToSVG.dtd.

- **X3D tags supported by this application**
  - <Appearance>, <Box>, <Cone>, <Cylinder>, <Group>, <Material>, <Transform>, <Shape>, <Sphere>, <Scene>, <X3D>

- **Features and User Interface**
  - Have features for the user to navigate a scene as it can be done in an X3D browser.
Requirements (cont.)

The sample of the output document
Requirements (cont.)

• Maximum Load
  – The program responds within a second when we change perspectives of the scene that renders up to 300-320 polygons (tested on machines having AMD Athlon XP or Pentium IV processor running 1300-1400 MHz and 512 MB Memory).
Requirements (cont.)

<table>
<thead>
<tr>
<th># of Polygons</th>
<th>Processing Time (milliseconds)</th>
</tr>
</thead>
<tbody>
<tr>
<td>32 (a cone)</td>
<td>~30-60</td>
</tr>
<tr>
<td>64 (a sphere)</td>
<td>~110-130</td>
</tr>
<tr>
<td>128 (two spheres)</td>
<td>~290-320</td>
</tr>
<tr>
<td>192 (three spheres)</td>
<td>~560-590</td>
</tr>
<tr>
<td>256 (four spheres)</td>
<td>~750-870</td>
</tr>
<tr>
<td>320 (five spheres)</td>
<td>~1090-1190</td>
</tr>
<tr>
<td>384 (six spheres)</td>
<td>~1590-1680</td>
</tr>
<tr>
<td>448 (seven spheres)</td>
<td>~1972-2133</td>
</tr>
<tr>
<td>512 (eight spheres)</td>
<td>~2444-2654</td>
</tr>
<tr>
<td>576 (nine spheres)</td>
<td>~3024-3405</td>
</tr>
<tr>
<td>640 (ten spheres)</td>
<td>~3225-3755</td>
</tr>
<tr>
<td>672 (ten spheres + one cone)</td>
<td>~3625-4266</td>
</tr>
</tbody>
</table>

A processing-time table of different test cases
Requirements (cont.)

• Limitations
  – Our program cannot guarantee to display some scenes of overlapping and intersecting objects correctly.
A golf ball with a tee sits on a green

An incorrect view

A leg of this tee should appear on the green, not behind the green.

A correct view

Users have to subdivide the polygons to get a correct view of this scene. The program cannot subdivide polygons.
Project’s Design & Implementation

X3D document → Load X3D file → A DOM structure of a X3D file → Viewing Pipeline → A mesh of polygons forming 3D graphics

Visible Surface Detection → Sorted front-face polygons & Sorted back-face polygons → Surface-rendering → A view of a 3D scene
Project’s Design & Implementation (cont.)

• Visible-Surface Detection
  – Can be classified in two categories, an object-space method and an image-space method.
  – Most visible-surface detection algorithms in computer graphics use image-space methods such as depth-buffer method or z-buffer method.
  – Only object-space methods can be used in our project.
Project’s Design & Implementation (cont.)

objects

0 1 2 3

box sphere cylinder cone

sort objects according their depths

sphere cone cylinder box

64 polygons 33 polygons 10 polygons 6 polygons

visible polygons invisible polygons visible polygons invisible polygons visible polygons invisible polygons visible polygons invisible polygons
Project’s Design & Implementation (cont.)

Sorted objects:
- **64 polygons** (sphere)
- **33 polygons** (cone)
- **10 polygons** (cylinder)
- **6 polygons** (box)

Visible polygons:
- **Invisible polygons**

Sorted visible polygons:
- **Sorted visible polygons**

Sorted invisible polygons:
- **Sorted invisible polygons**

Sorted polygons:
- **Sorted polygons**
Project’s Design & Implementation (cont.)

The polygon’s depth

A center point \((x, y, z)\)

\[
\begin{align*}
x &= (x_1 + x_2 + x_3 + x_4)/4 \\
y &= (y_1 + y_2 + y_3 + y_4)/4 \\
z &= (z_1 + z_2 + z_3 + z_4)/4
\end{align*}
\]
Project’s Design & Implementation (cont.)

<table>
<thead>
<tr>
<th>Primitive</th>
<th>Plane</th>
<th>Point3D</th>
<th>Polygon</th>
<th>Vector</th>
</tr>
</thead>
<tbody>
<tr>
<td>createTag()</td>
<td>A, B, C visibility</td>
<td>XYZ, Y3D, Z3D</td>
<td>vertices</td>
<td>A, B, C</td>
</tr>
<tr>
<td>modifyWC()</td>
<td>Xp, Yp, Zp</td>
<td>setXYZ()</td>
<td>id</td>
<td>size( )</td>
</tr>
<tr>
<td>createVC()</td>
<td></td>
<td>rotate_translate()</td>
<td>object</td>
<td>normalize( )</td>
</tr>
<tr>
<td>createProjection()</td>
<td></td>
<td>translate()</td>
<td>plane</td>
<td>addVector( )</td>
</tr>
<tr>
<td>adjustCenter()</td>
<td></td>
<td>rotate()</td>
<td>visibility</td>
<td>dotVector( )</td>
</tr>
<tr>
<td>createPlane()</td>
<td></td>
<td>scale()</td>
<td></td>
<td></td>
</tr>
<tr>
<td>findIntensity()</td>
<td></td>
<td>project()</td>
<td></td>
<td></td>
</tr>
<tr>
<td>modifyGradientTag()</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- Box
  - createFirstWC()
  - assignProjectionToFaces()
  - createNormalVector()
  - findColor()
  - getVCAndProjection()
  - getShade()

- Cone
  - createFirstWC()
  - assignProjectionToFaces()
  - createNormalVector()
  - findColor()
  - getVCAndProjection()
  - getShade()

- Cylinder
  - createFirstWC()
  - assignProjectionToFaces()
  - createNormalVector()
  - findColor()
  - getVCAndProjection()
  - getShade()

- Sphere
  - createFirstWC()
  - assignProjectionToFaces()
  - createNormalVector()
  - findColor()
  - getVCAndProjection()
  - getShade()
Optimization

• Optimize the rendering algorithms
  – The first version (non-optimized version):
    • Renders every polygon on the scene (both front-face and back-face polygons).
  – The second version:
    • About 50% of the system processing time is coming from the surface-rendering procedure.
    • Want to reduce the bottleneck surface-rendering processing time.
    • Selectively renders only front-face polygons.
    • Improve the performance by 30-40%.
    • Has time complexity $O(n^2 \ln n)$; n is the number of polygons
## Optimization (cont.)

<table>
<thead>
<tr>
<th># of Polygons</th>
<th>System processing time</th>
<th>Surface-rendering processing time</th>
</tr>
</thead>
<tbody>
<tr>
<td>36</td>
<td>~70-80 ms</td>
<td>~ 30-40 ms</td>
</tr>
<tr>
<td>113</td>
<td>~240-290 ms</td>
<td>~110-140 ms</td>
</tr>
<tr>
<td>192</td>
<td>~560-590 ms</td>
<td>~280-310 ms</td>
</tr>
<tr>
<td>309</td>
<td>~1000-1400 ms</td>
<td>~700-930 ms</td>
</tr>
</tbody>
</table>

Comparing between the system processing time and the surface-rendering processing time.
Optimization (cont.)

<table>
<thead>
<tr>
<th># of Polygons</th>
<th>Processing time</th>
</tr>
</thead>
<tbody>
<tr>
<td>32 (a cone)</td>
<td>~30-60 ms</td>
</tr>
<tr>
<td>64 (a sphere)</td>
<td>~110-130 ms</td>
</tr>
<tr>
<td>128 (two spheres)</td>
<td>~290-320 ms</td>
</tr>
<tr>
<td>192 (three spheres)</td>
<td>~560-590 ms</td>
</tr>
<tr>
<td>256 (four spheres)</td>
<td>~750-870 ms</td>
</tr>
<tr>
<td>320 (five spheres)</td>
<td>~1090-1190 ms</td>
</tr>
<tr>
<td>384 (six spheres)</td>
<td>~1590-1680 ms</td>
</tr>
<tr>
<td>448 (seven spheres)</td>
<td>~1972-2133 ms</td>
</tr>
<tr>
<td>512 (eight spheres)</td>
<td>~2444-2654 ms</td>
</tr>
<tr>
<td>576 (nine spheres)</td>
<td>~3024-3405 ms</td>
</tr>
<tr>
<td>640 (ten spheres)</td>
<td>~3225-3755 ms</td>
</tr>
<tr>
<td>672 (ten spheres + one cone)</td>
<td>~3625-4266 ms</td>
</tr>
</tbody>
</table>

The processing time of our program version 2

A graph associated with results shown in the table
Optimization (cont.)

– The third version:

• Apply the hidden-surface removal algorithm to reduce a number of rendered polygons so that the surface-rendering time can decrease.

• This algorithm is to check and remove any UNSEEN or HIDDEN front-face polygons.

• The algorithm for detecting a hidden surface is every front-face polygons must be checked with the other CLOSER polygons that belong to DIFFERENT objects.
Optimization (cont.)

Sphere has 32 front-face polygons

Cone has 16 front-face polygons

The second version renders $= 32 + 16$ polygons

The third version renders $< 32 + 16$ polygons

Because some front faces of the sphere are hidden by some front faces of the cone.
Optimization (cont.)

- Ironically, it drops the performance more than 100% because the overhead of the hidden-surface removal process is more than the time we save from the surface-rendering process.
Optimization (cont.)

Test Case I: 36 polygons

<table>
<thead>
<tr>
<th>Version</th>
<th>Processing Time</th>
<th>% Performance Increases or Drops (compared with Version I)</th>
</tr>
</thead>
<tbody>
<tr>
<td>I</td>
<td>~90-110 ms</td>
<td>-</td>
</tr>
<tr>
<td>II</td>
<td>~60-80 ms</td>
<td>Increases 20-30 %</td>
</tr>
<tr>
<td>III</td>
<td>~230-270 ms</td>
<td>Drops &gt; 100%</td>
</tr>
</tbody>
</table>

Test Case II: 192 polygons

<table>
<thead>
<tr>
<th>Version</th>
<th>Processing Time</th>
<th>% Performance Increases or Drops (compared with Version I)</th>
</tr>
</thead>
<tbody>
<tr>
<td>I</td>
<td>~ 830-850 ms</td>
<td>-</td>
</tr>
<tr>
<td>II</td>
<td>~ 560-590 ms</td>
<td>Increases 32-36 %</td>
</tr>
<tr>
<td>III</td>
<td>~ 3895-6690 ms</td>
<td>Drops &gt; 100 %</td>
</tr>
</tbody>
</table>
Optimization (cont.)

- Of all three versions, the second version has the best performance.
- The third version, which we thought would be the best, has the worst performance because of the overhead of the hidden-surface removal process.
Optimization (cont.)

- Performance hints to speed up JavaScript
  - Avoid using a dot operator because it requires time to dereference. Locally cache values if we need to access them very often. IMPROVE
  - Use a `var` statement when possible. IMPROVE
  - Try to reuse the object when possible because a constructor `new` can be slow. IMPROVE
  - It is recommended to use the forms of the Array constructor that specify a size or take a list of initial elements. It will speed up the code. NOT IMPROVE
Conclusion

• Project Achievements
  – Succeeded in using SVG (originally designed for 2D graphics) to render 3D graphics.
  – Deliver an application which translates X3D to SVG, produces a view of smooth shaded 3D objects, and has features for the user to navigate the scene.
  – Optimized the application and achieved a performance improvement around 30-40%.
Conclusion (cont.)

– Discover that not all of performance hints which claimed to boost JavaScript code can speed up the program.

– Successfully used JavaScript in a nonstandard way to develop a nontrivial and complicated task like a rendering program.
Conclusion (cont.)

• Future Work
  – Subdivide polygons so that the translator can display overlapping and intersecting objects correctly.
  – Clip a part of the scene outside the viewing limit.
Conclusion (cont.)

– Optimize a hidden-surface removal algorithm. Reduce a number of front-face polygons to be checked by using a hidden-object removal approach instead.
Conclusion (cont.)

A hidden-surface removal method

A hidden-object removal method

[Diagrams and text describing the methods]
Some polygons of sphere are outside the viewing limit but they are still rendered by our program.
Polygons are sorted ONLY with the other polygons of the same collection (object).

Polygons are sorted with every other polygon in the scene.

Correct views from an X3D browser.