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**~~

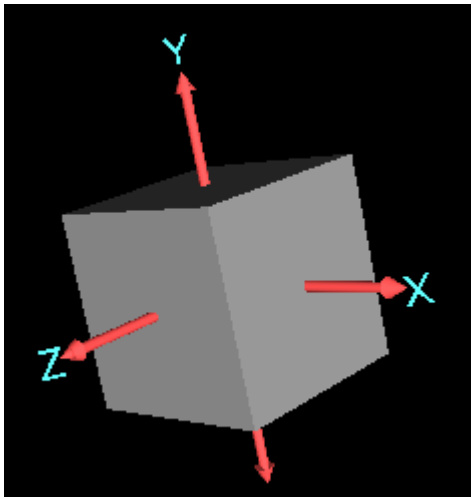
~~with~~

~~**Plugins**~~

Browsers

without

Plugins by SVG

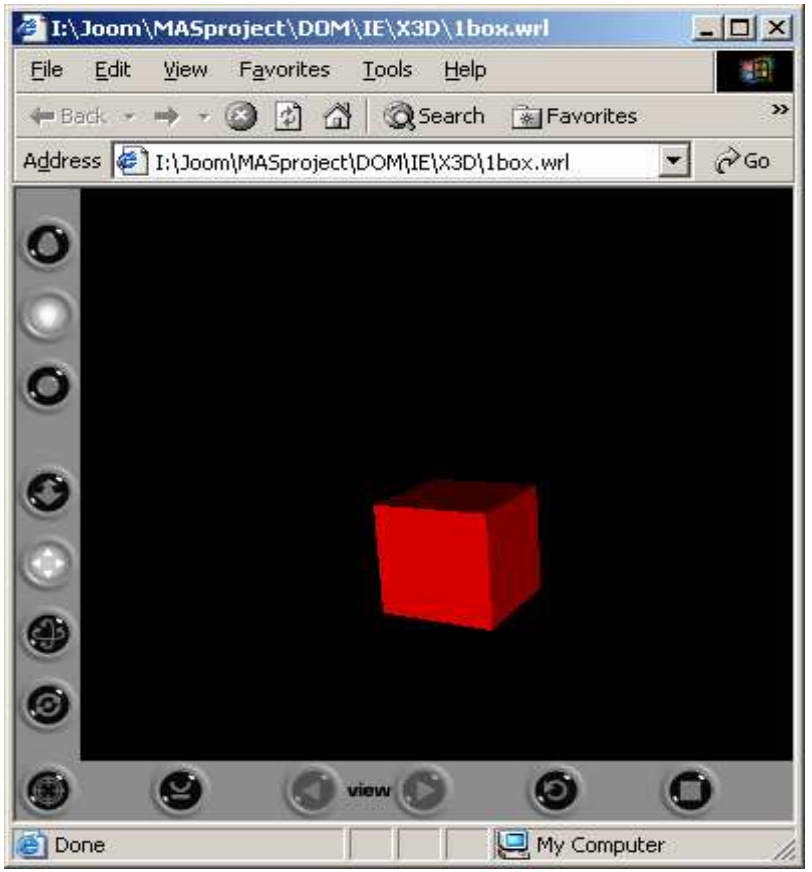


Introduction (cont.)

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



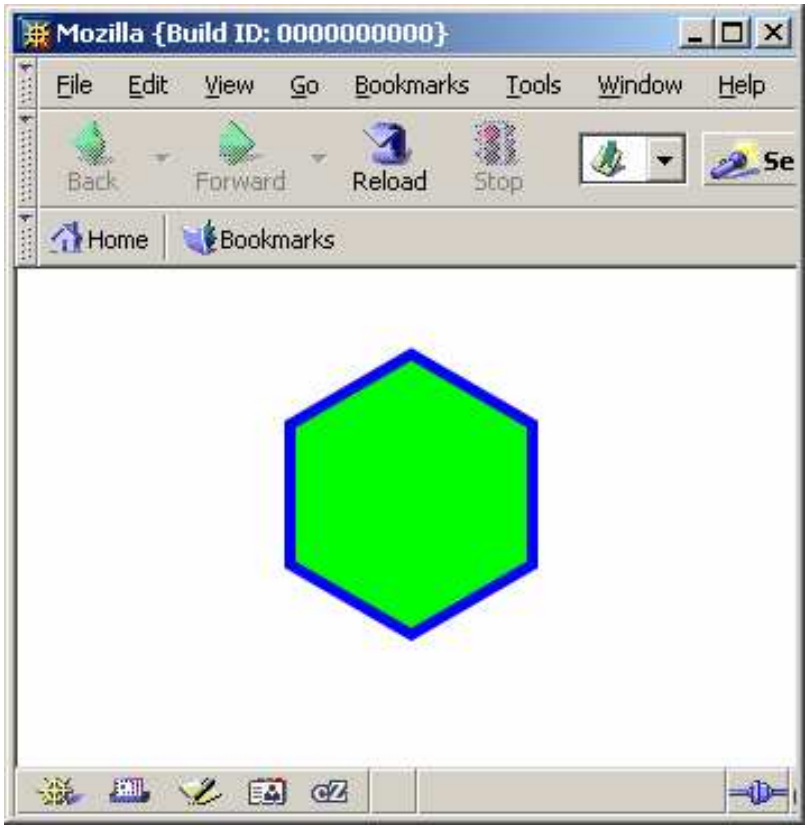
Introduction (cont.)

SVG document

```
<?xml 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



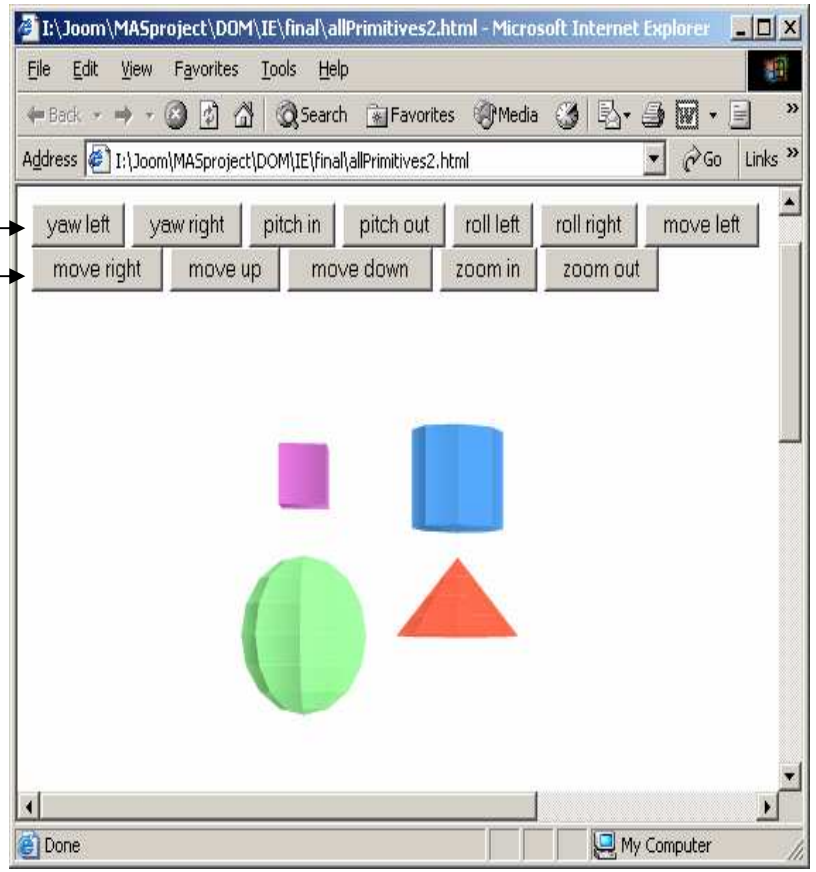
Introduction (cont.)

X3D source data

```
<?xml 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

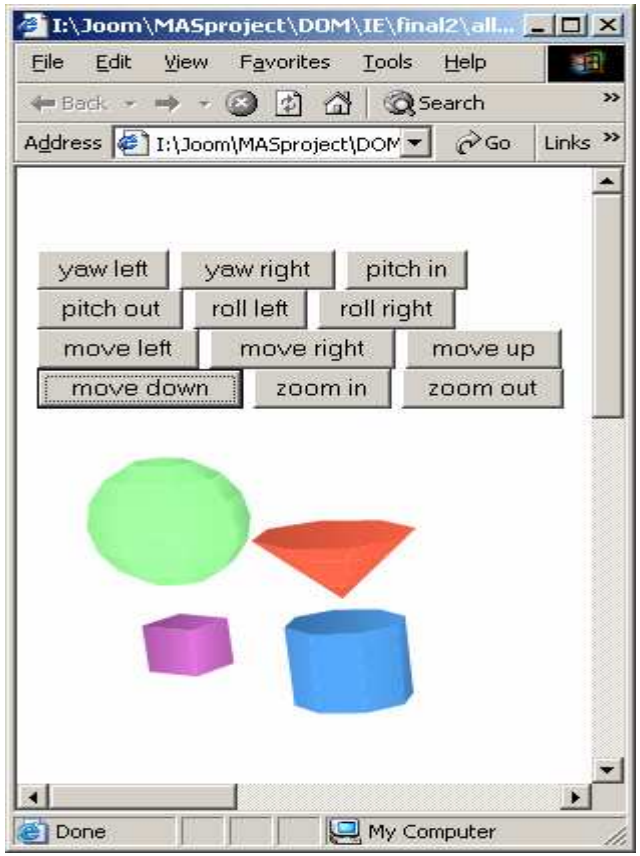
Buttons for changing perspectives

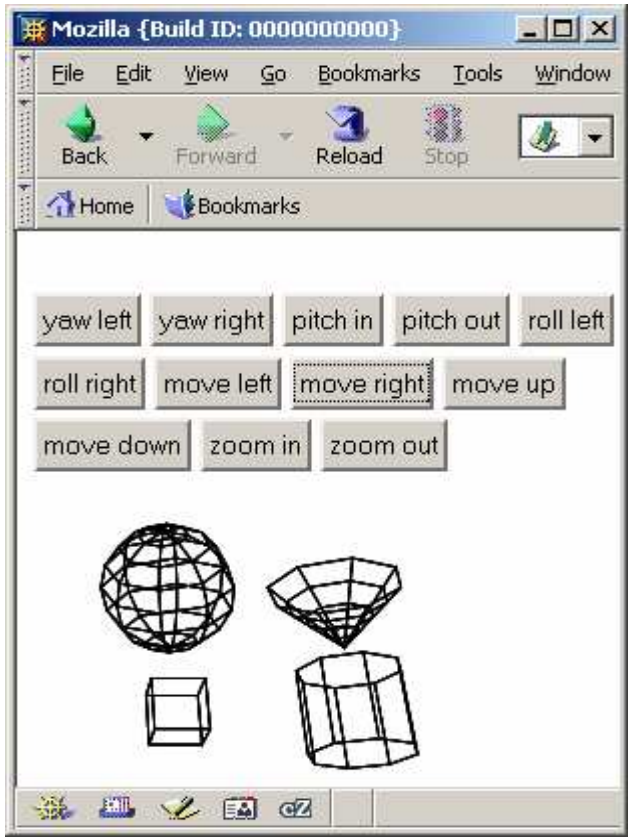


Requirements

- 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 AthlonXP or Pentium IV processor running 1300 –1400 MHz and 512 MB Memory).

Requirements (cont.)

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

A processing-time table of different test cases

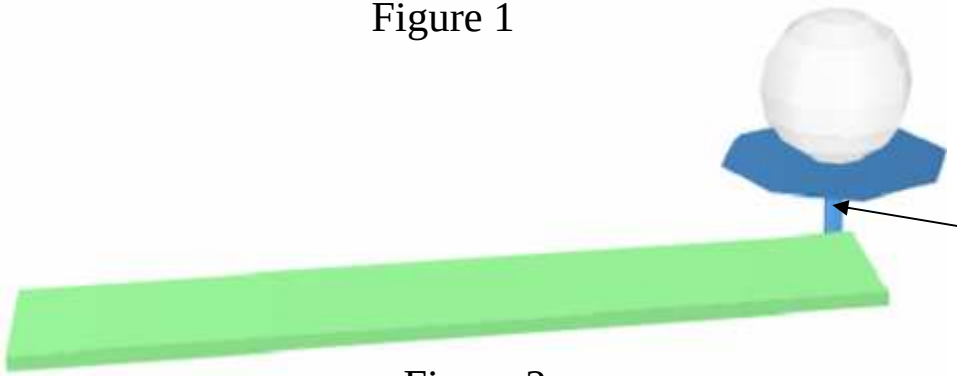
Requirements (cont.)

- 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

Figure 1

An incorrect view



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

Figure 2

A correct view

Users have to subdivide the polygons to get a correct view of this scene. The program cannot subdivide polygons

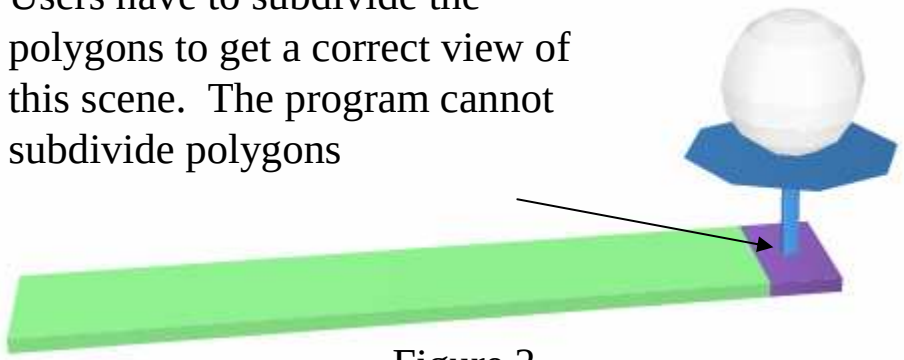
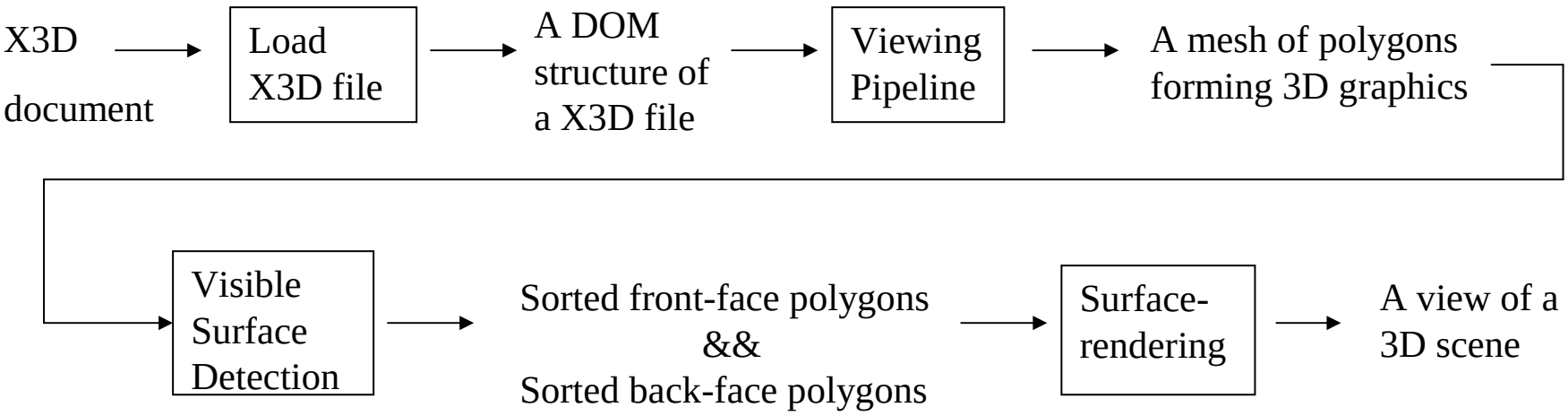


Figure 3

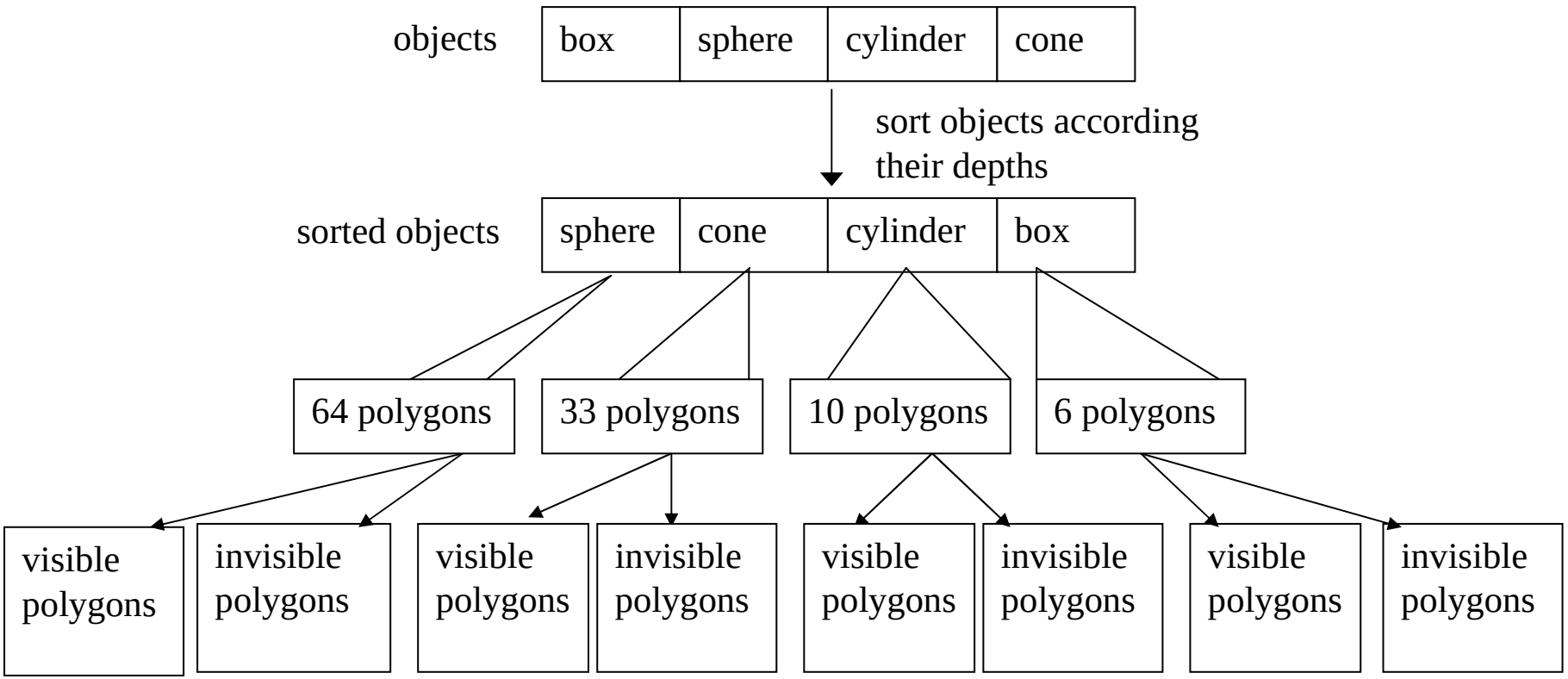
Project’s Design & Implementation



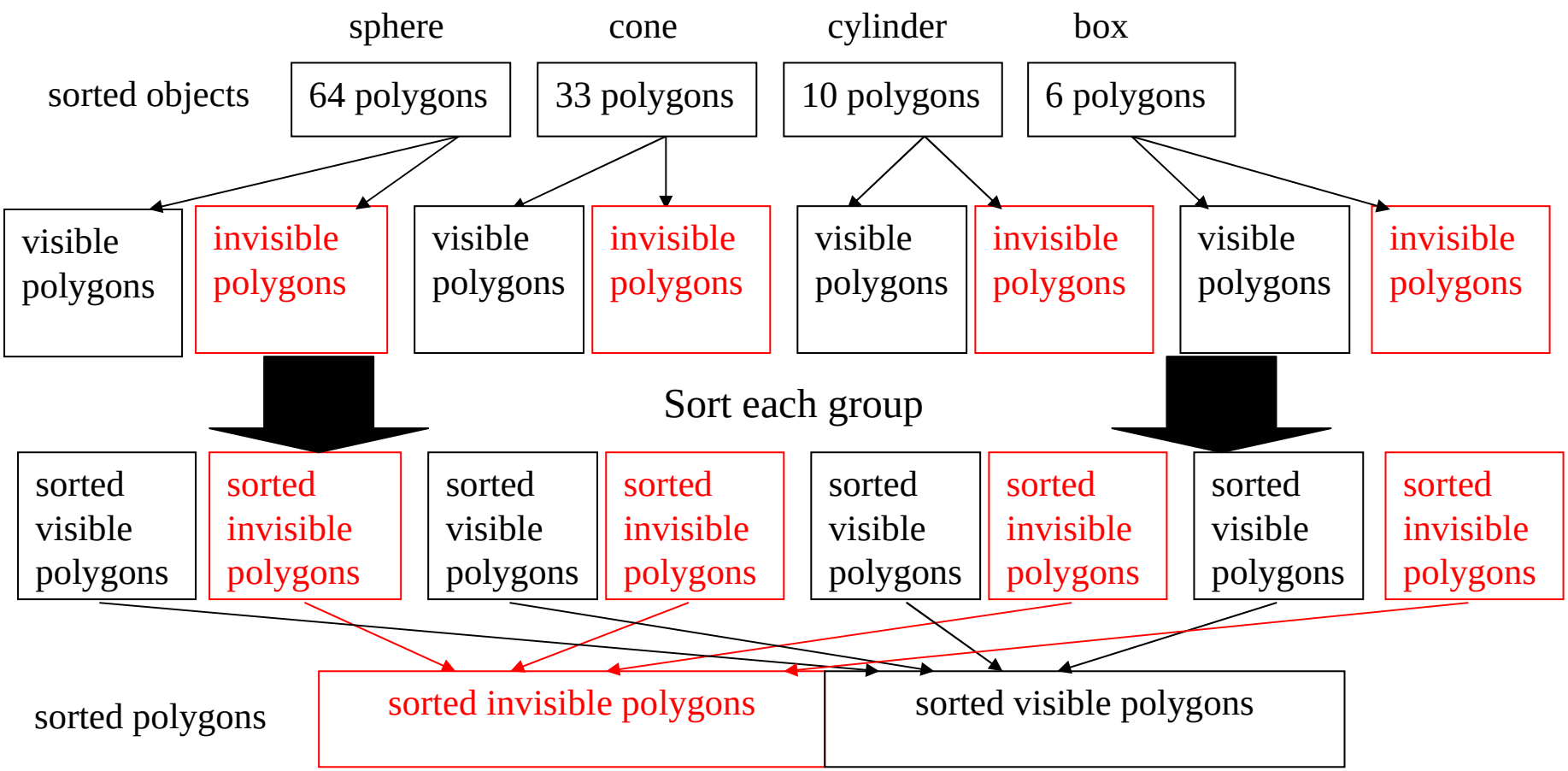
Project's Design & Implementation (cont.)

- 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.)



Project's Design & Implementation (cont.)



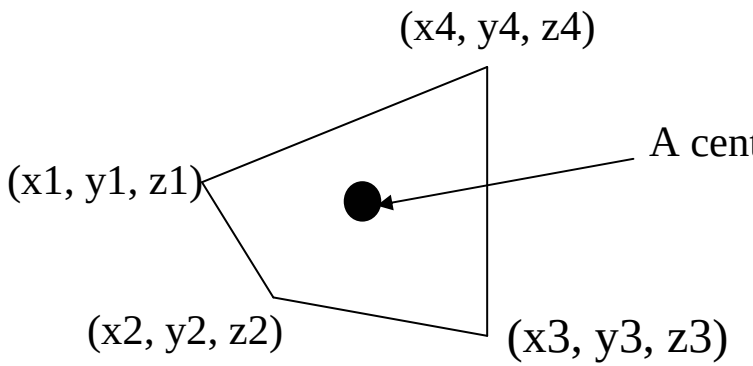
Project’s Design & Implementation (cont.)

The polygon’s depth

x = (x1+x2+x3+x4)/4

y = (y1+y2+y3+y4)/4

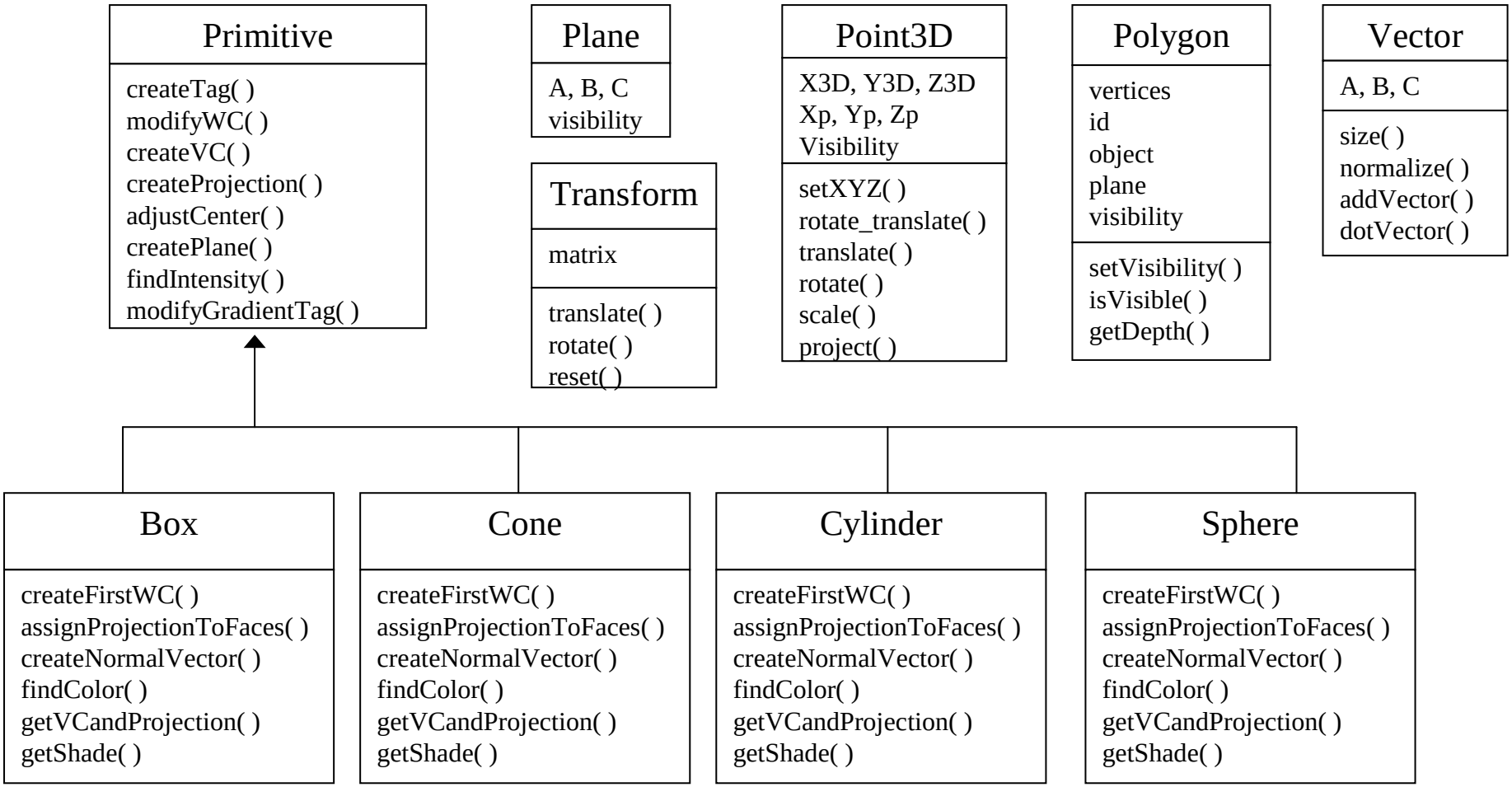
z = (z1+z2+z3+z4)/4



A center point (x, y, z)

The depth of the polygon

Project's Design & Implementation (cont.)



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.)

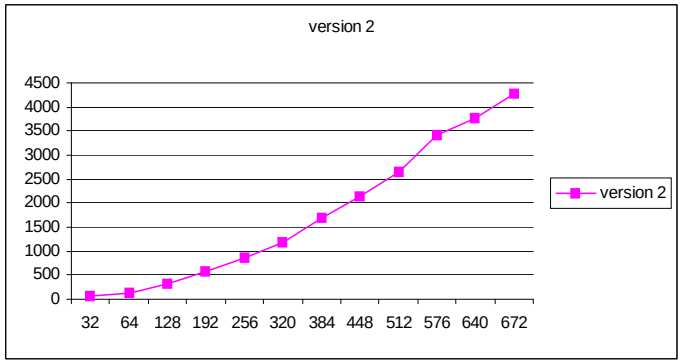
# of Polygons	System processing time	Surface-rendering processing time
36	~70-80 ms	~ 30-40 ms
113	~ 240-290 ms	~ 110-140 ms
192	~ 560-590 ms	~ 280-310 ms
309	~ 1000-1400 ms	~ 700-930 ms

Comparing between the system processing time
and the surface-rendering processing time.

Optimization (cont.)

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

The processing time of our program version 2

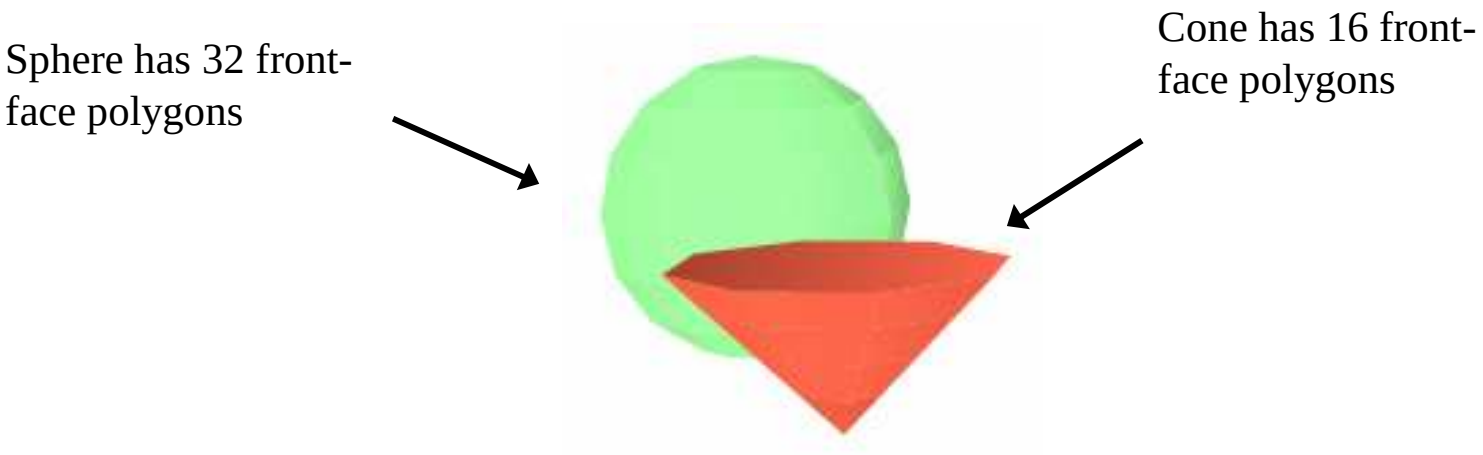


A graph associated with results shown in the table

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.)



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

Version	Processing Time	% Performance Increases or Drops (compared with Version I)
I	~90-110 ms	-
II	~60-80 ms	Increases 20-30 %
III	~230-270 ms	Drops > 100%

Test Case II: 192 polygons

Version	Processing Time	% Performance Increases or Drops (compared with Version I)
I	~ 830-850 ms	-
II	~ 560-590 ms	Increases 32-36 %
III	~ 3895-6690 ms	Drops > 100 %

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.)

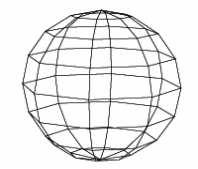
- Future Work
  - Subdivide polygons so that the translator can display overlapping and intersecting objects correctly.
  - Clip a part of the scene outside the viewing limit.

Conclusion (cont.)

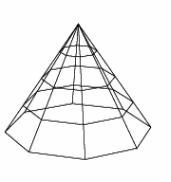
– Optimize a hidden-surface removal algorithm. Reduce a number of front-face polygons to be checked by using a hidden-object removal approach instead.

Conclusion (cont.)

A hidden-surface removal method

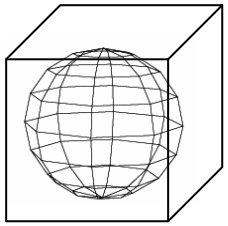


thirty two front-face polygons → verified with →

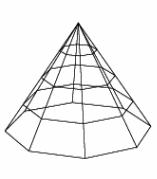


sixteen front-face polygons

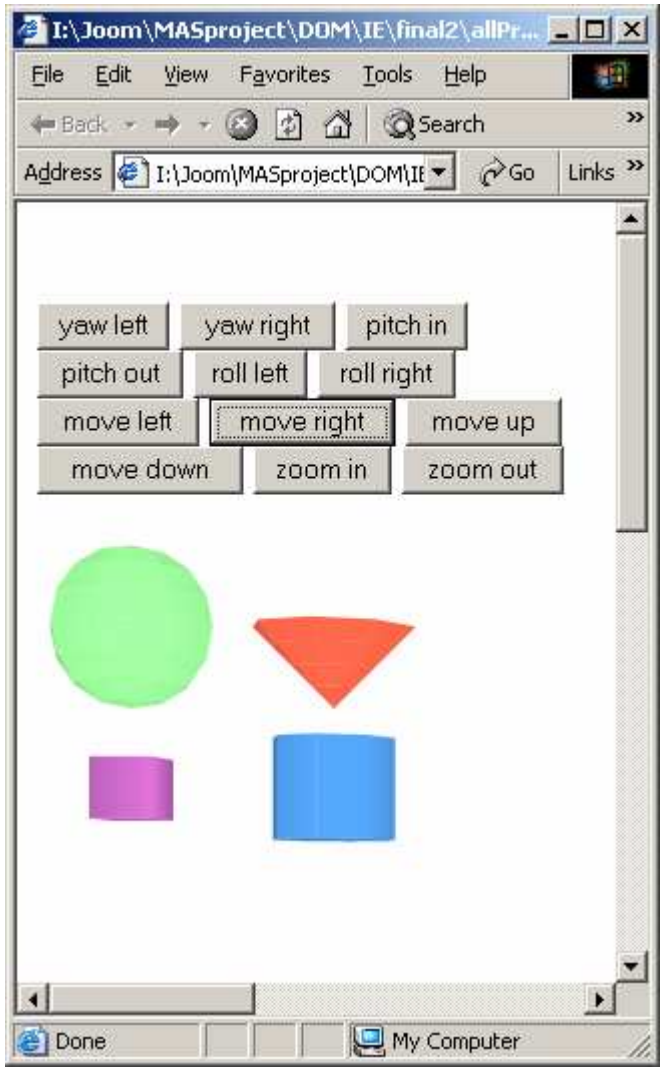
A hidden-object removal method



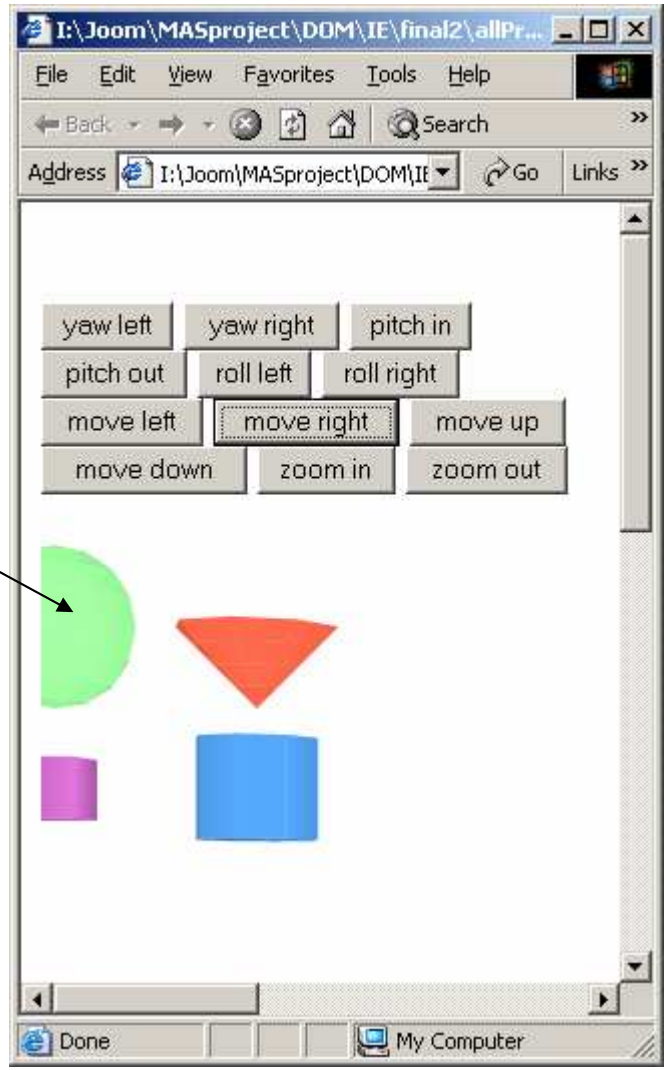
a boundary box (only six polygons will be verified) → verified with →



sixteen front-face polygons

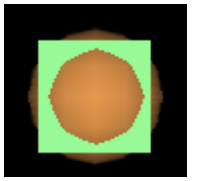


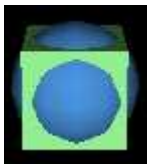
Some polygons of sphere are outside the viewing limit but they are still rendered by our program.

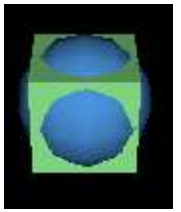


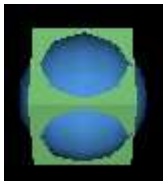
Polygons are sorted ONLY with the other polygons of the same collection (object).

Polygons are sorted with every other polygon in the scene









Correct views from an X3D browser