3D Object Picking in Web-based Design System

Dong-Hyun Kim, Bo-Yeul Yun and Eung-Kon Kim, *Member, KIMICS*

Abstract—We are able to work on the shared virtual space in Web-based Collaborative Design System using only Internet and Web browser. Then the users will share 3D objects and must be able to pick the objects effectively which they want to manipulate.

In this paper, picking is implemented not only by computing intersection of mouse pointer with the objects of the virtual world, but also by using capabilities and attributes of scene graph node, by setting bounds intersection testing instead of geometric intersection testing, by limiting the scope of the pick testing, using Java 3D. These methods can reduce the computation of picking and can pick 3D objects effectively and easily using the system of hierarchy.

Index Terms—Computer Graphics, 3D modeling, Solid modeler, Picking

I. INTRODUCTION

Traditional collaborative works were progressed by meeting each other and talking their opinions at an appointed hour and place. In these days, owing to development of computer and communication technology, new systems appear all around us, which can be worked effectively, interacting on the shared virtual space without any restrictions by time and space [1,2].

Collaborative Systems, which have been developed and used until now, include E-mail system, video conference, E-confirmation, cyberschool, remote medical diagnosis & examination, cooperative programming, and so on. However, most of the systems can work with some restriction which depends on platforms and softwares. Moreover the shared objects are the forms of window explorer such as folders, documents, memo[3].

We are developing a collaborative design system which will be able to work on web browser of client, approaching the design server through Internet. The clients share 3D objects because of designing in virtual space. Therefore the users who participate in collaborative design must be able to select the object which he or she want to manipulate. Picking gives the user a way to interact with individual visual objects in the scene.

Manuscript received April 17, 2003.

Dong-Hyun Kim is with the Department of Computer Information, Suncheon Cheong-am College, Suncheon-si, Korea.

Bo-Yeul Yun is with the Suncheon Maesan High shcool, Suncheon-si, Korea.

Eung-Kon Kim, corresponding author is with the Department of Computer Science, Sunchon University, Suncheon-si, Korea. (Phone: 061-750-3627, e-mail: kek@sunchon.ac.kr).

Mouse is widely used as a pointing device in GUI (Graphic User Interface) environment. It is easy to select the area of icon made with a fixed size and shape, but it's not easy to pick other objects. Intersection of a ray projecting from mouse pointer with the objects must be checked. To pick Objects in 3D space, computationally expensive is required.

In this paper, picking is implemented not only by computing intersection of mouse pointer with the objects of the virtual world, but also by considering the hierarchy of 3D scene graph. This method can reduce the computation of picking and can pick 3D objects effectively. The structure of this paper is as follows. In chapter 2, we will introduce a web-based collaborative design system. In chapter 3, we will describe the general picking method using the intersection. In chapter 4, we will explain the effective picking methods using the scene graph node which benefit 3D objects.

II. THE BRIEF OF WEB-BASED COLLABORATIVE DESIGN SYSTEM

The design system has Client/Server architecture. Clients can connect to the design server, then design solid model, store the data as the various file format, and display the data from other CAD systems. This system can support the detail design as 3D graphics features such as viewing, rendering, animating are available.

The system consist of connection manager which can control log-in, working manager which can keep viewing and shared working space, solid modeler which creates 3D objects. Fig.1 shows the entire structure of Web-based Collaborative Design System briefly.

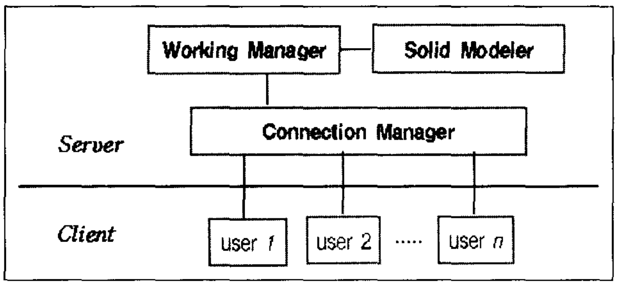


Fig. 1 Web-based Collaborative Design System

Connection Manager analyzes the request of client from the design server and sends the message to session processing module. It makes the user connect or disconnect according to session information, and keep connecting to server while working.

Working Manager processes the job request of client, maintains individual working space and shared working space with WYSIWIS(What You See Is What I See) viewing and concurrency control.

Solid Modeler consists of a set of Java classes to model and manipulate solids. The types of solid primitives are block, cylinder, cone, torus, pyramid, sphere, and so on. With these solid primitives, various solid models can be made by boolean operations such as INTERSECTION, DIFFERENCE, and UNION. And they cam be conversed by TRANSLATION, SCALING, REFLECTION, and ROTATION. The users can create objects and manipulate them in the modeler, and they can select any of them they want to operate. Picking gives the user a way to interact with individual visual objects in the scene.

III. PICKING USING THE INTERSECTION

Picking is to select an object or the mesh which consists of a part of object[4]. Picking is implemented by a behavior typically triggered by mouse button events. In picking a visual object, the user places the mouse pointer over the visual object of choice and presses the mouse button. The behavior object is triggered by this button press and begins the picking operation.

A ray is projected into the virtual world from the position of the mouse pointer parallel with the projection. Intersection of this ray with the objects of the virtual world is computed. The visual object intersecting closest to the image plate is selected for interaction. Fig. 2 shows a pick ray projected in a virtual world[5].

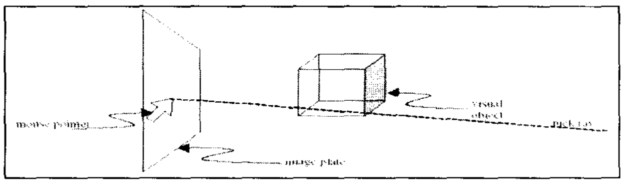


Fig. 2 Projection of Pick Ray in the Virtual World

Most of all we will explain the picking point. After the distance(D) between mouse pointer coordinate (x,y) and object coordinate (x1,y1) is computed, it is compared to find whether it is less than the range of APERTURE. Object coordinate(x1,y1) of the left side in Fig.3 is not picked, in the case of (x-x1)2 + (y-y1)2 > APERTURE2, but object coordinate(x1,y1) of the right side in Fig.3 is picked, in the case of (x-x1)2 + (y-y1)2 < APERTURE2.[6].

D2=(x-x1)2 + (y-y1)2
i) (x-x1)2 + (y-y1)2 > APERTURE2
ii) (x-x1)2 + (y-y1)2 < APERTURE2

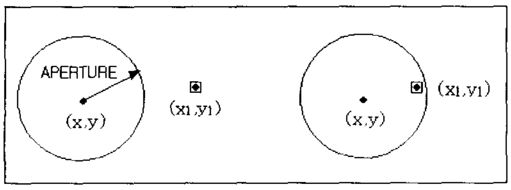


Fig. 3 Picking Point Object with Coordinate(x1,y1)

It takes much time to find D<APERTURE because of computing the value of square root. Therefore, it may as well to use square which sizes 2 APERTURE and has the center (x,y) as use circle like Fig. 4.

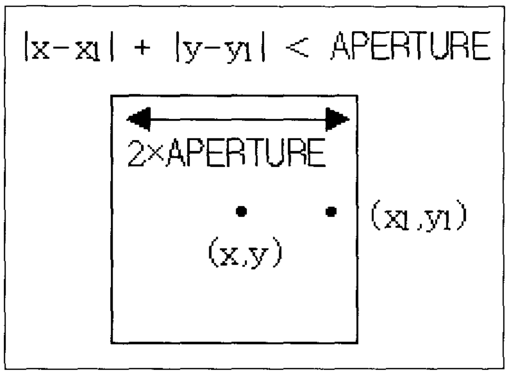


Fig. 4 Picking Test by Square for Point

In the case of a straight line, if a perpendicular line drawing from one point is less than the rage of APERTURE, the line is picked. Distance(D) from point coordinate (x,y) to straight line ax + by = 0 is as follows.

D={| ax sub 1 + by sub 1 + c |} OVER { sqrt {a sup2 + b sup2 }}

In the case of a definite straight line, Distance(D) from point coordinate (x,y) to the line with two coordinates (x1,y1), (x2,y2) is as follows.

D = {| (x-x sub1) (y sub2 - y sub1) - (y - y sub1) (x sub2 -x sub1) |} over {sqrt { { (x sub2 - x sub1) } sup2 + { (y sub2 - y sub1) } sup2 } }

Where the line comes into the circle like Fig. 5, it is picked. To pick the line, the following condition is needed.

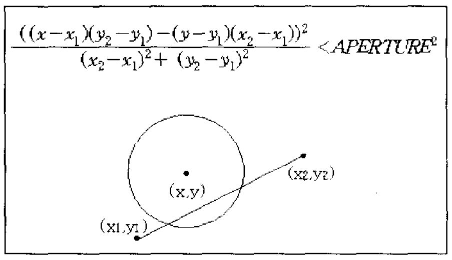

$\frac{((x-x_1)(y_2-y_1)-(y-y_1)(x_2-x_1))^2}{(x_2-x_1)^2+(y_2-y_1)^2} < APERTURE^2$
(x,y)
(x2,y2)
(x1,y1)

Fig. 5 Picking Line Object with two coordinates (x1,y1), (x2,y2)

In this time, to avoid a great number of arithmetic computation, the square is used as well like Fig. 6.

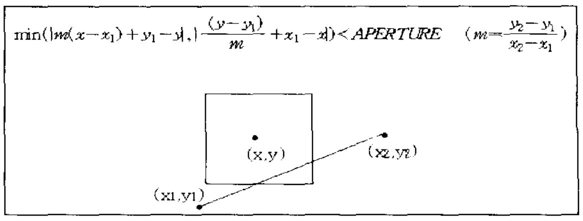

$\min(|m(x-x_1)+y_1-y|, |\frac{(y-y_1)}{m}+x_1-x|) < APERTURE \quad (m=\frac{y_2-y_1}{x_2-x_1})$
(x,y)
(x2,y2)
(x1,y1)

Fig. 6 Picking Test by Square for Line

Where the mouse pointer is placed in the area of the object, it cab be picked. When there are a few layers, the new problems which layer will be picked can occur. In this case, we have to make the object which is closest from mouse pointer picked. In Fig. 7 triangle B, which is closest to oblique side, is picked[4].

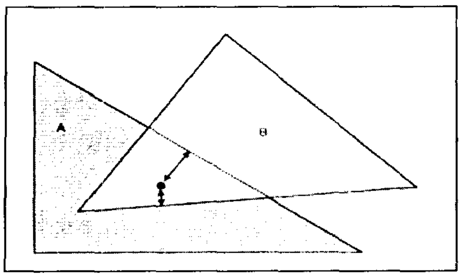


Fig. 7 Picking for Objects in Duplicated Layers

The object which wanted to pick can be shown easily as the pick window is presented around mouse pointer like Fig. 8.

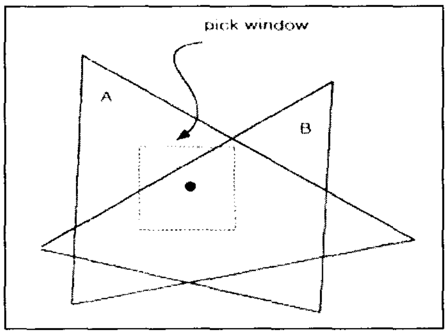


Fig. 8 Using Pick Window

IV. PICKING USING THE 3D SCENE GRAPH NODE

Picking occurs in several stages, with the earlier stages getting you close, and the later stages providing spatial precision to the hit determination. The followings are an overview of the full precision process[7]. Fig. 9 shows that mouse pointer hits the closest object in 3D world space according to the value of z-axis.

1. Throw a pick ray from the user's virtual eyeball, through the mouse position in the view's display plane, and on into the scene. This is shown in Fig. 9.
2. Intersect the pick ray with the bounds surrounding the shape objects in the scene, sorting the hits from closest to farthest.
3. Test each hit candidate, starting with the closest, for intersection between the pick ray and the candidate's actual geometric shape.
4. If no shape is hit, or the hit object owning the shape is not a designated target object, quit the process without a real hit.
5. Determine the hit point on the hit object using the distance

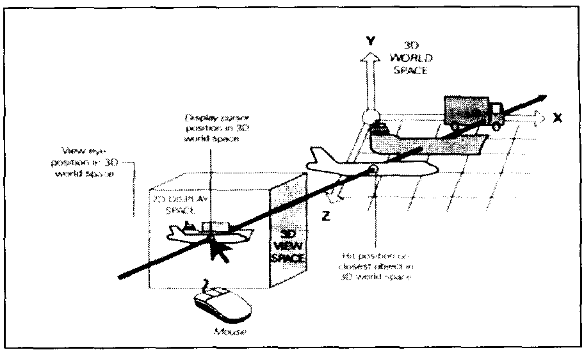

Y
3D WORLD SPACE
X
Display cursor position in 3D world space
View eye position in 3D world space
2D DISPLAY SPACE
3D VIEW SPACE
Z
Hit position on closest object in 3D world space
Mouse

Fig. 9 The Pick Ray in the 3D World

In some cases the interaction is not directly with the selected object, but with an object along the scene graph path to the object. If a visual object is composed of six individual Shape3D objects, as in the scene graph diagram of Fig. 10, it is not the TransformGroup object above the intersected Shape3D object that needs to be modified. The object is modified by manipulation of the TransformGroup object that is the child of the BranchGroup object in the scene graph. The result of some picking operations is to return the scene graph path for further processing.

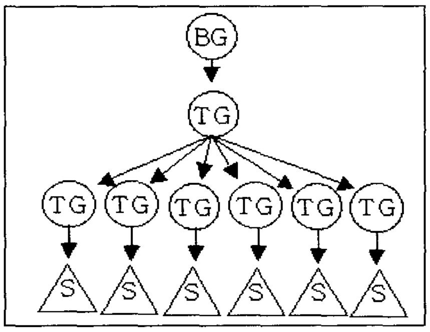


Fig. 10 Scene Graph Diagram for a Cube

Moreover above mentioned methods are needed a great number of arithmetic computation. Therefore we must pay attention to the node of 3D scene graph. Especially 3D objects which use the tree structure of scene graph have a lot of advantage. 3D object may be disassembled by its tree hierarchy like Fig. 11, if upper node is picked, the child node is also picked as well[8].

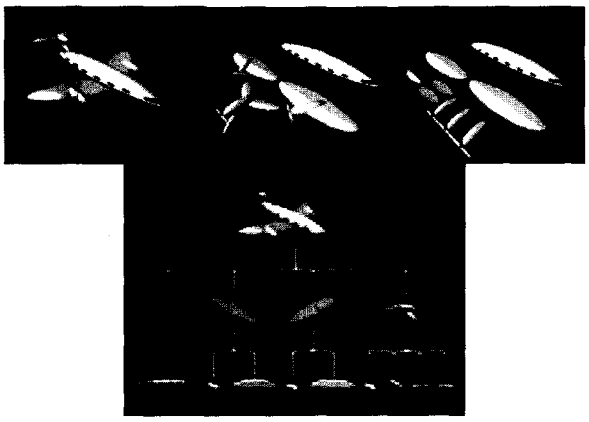


Fig. 11 Example of Hierarchy for Airplane

Intersection testing is computationally expensive. Therefore, picking is computationally expensive and is more expensive with increasing complexity of the scene. The Java 3D API provides a number of ways that a programmer can limit the amount of computation done in picking[5].

One important way is through the capabilities and attributes of scene graph nodes. Whether or not a scene graph node is pickable is set with setPickable() method of the class. A node with setPickable() set to false is not pickable and neither are any of its children. Consequently, these nodes are not considered when calculation intersections. The following reference block list Node method.

setPickable(boolean pickable)

Another way of picking related feature of the Node class is the ENABLE_PICK_REPORTING capability. This capability applies only to Group nodes. When set for a Group, that group object will always be included in the scene graph path returned by a pick operation. The following reference block list Node capabilities.

ENABLE_PICK_REPORTING
ALLOW_BOUNDS_READ | WRITE
ALLOW_PICKABLE_READ | WRITE

To reduce the computation of picking, there is another way to use bounds intersection testing instead of geometric intersection testing. Several pf the pick related classes have constructor or methods have a parameter : USE_BOUNDS or USE_GEOMETRY. When the USE_BOUNDS option is selected, the pick is determined using the bounds of the visual objects, not the actual geometry. The determination of a pick using the bounds is significantly easier for all but the simplest geometric shapes and therefore, results in better performance. Of course, the drawback is the picking is not as precise when using bounds pick determination.

USE_BOUNDS
USE_GEOMETRY
PickBounds(Bounds boundsObject)

The next technique for reducing the computational cost of picking is to limit the scope of the pick testing to the relevant portion of the scene graph. In each picking utility class a node of the scene graph is set as the root of the graph for picking testing. This node is not necessarily the root of the content branch graph. On the contrary, the node passed should be the root of the content subgraph that only constrains pickable objects, if possible.

The followings are the methods which implemented in solid modeler using Java 3D API according to the above-mentioned three methods.

Node Method
extends: SceneGraphObject
subclasses: Group, Leaf
The Node class provides an abstract class for all Group and Leaf Nodes. It provides a common framework for constructing a Java 3D scene graph, specifically bounding volumes, picking and collision capabilities.
void setBounds(Bounds bounds)
Sets the geometric bounds of a node.
void setBoundsAutoCompute(boolean autoCompute)
Turns the automatic calcuation of geometric bounds of a node on/off.
setPickable(boolean pickable)
When set to true this Node can be Picked. Setting to false indicates that this node and it's children are ALL unpickable.

Node Capabilities
ENABLE_PICK_REPORTING
Specifies that this Node will be reported in the pick SceneGraphPath if a pick occurs. This capability is only specifiable for Group nodes; it is ignored for leaf nodes. The default for Group nodes is false. Interior nodes not needed for uniqueness in a SceneGraphPath that don't have ENABLE_PICK_REPORTING set will be excluded from the SceneGraphPath.
ALLOW_BOUNDS_READ | WRITE
Specifies that this Node allows read (write) access to its bounds information.
ALLOW_PICKABLE_READ | WRITE
Specifies that this Node allows reading (writing) its pickability state.

PickBounds
extends: PickShape
A bounds to supply to the BranchGroup and Locale pick methods
PickBounds()
Create a PickBounds.
PickBounds(Bounds boundsObject)
Create a PickBounds with the specified bounds.
Bounds get()
Get the boundsObject from this PickBounds.
void set(Bounds boundsObject)
Set the boundsObject into this PickBounds.

Code fragment in Fig. 13 shows the createSceneGraph method of MousePickApp. This program uses a PickRotate object to provide interaction. Since the construction of the picking object requires the Canvas3D object, the createSceneGraph method differs from earlier versions by the inclusion of the canvas parameter[5]. This code is annotated with the step from the recipe of followings which uses the mouse picking utility class. This program allows the user to pick an object to interact with.

1. **Create the scene graph (line 13-15)**
2. **Create a picking behavior object with root, canvas, and bounds specification (line 20)**
3. **Add the behavior object to the scene graph (line 21)**
4. **Enable the appropriate capabilities for scene graph objects (line 26-28)**

```
1. public BranchGroup
   createSceneGraph(Canvas3D canvas) {
2. // Create the root of branch graph
3. BranchGroup objRoot = new BranchGroup();
4.
5. TransformGroup objRotate = null;
6. PickRotateBehavior pickRotate = null;
7. Transform3D transform = new Transform3D();
8. BoundingSphere behaveBounds = new
   BoundingSphere();
9.
10. // Create ColorCube PickRotateBehavior
    objects
11. transform.setTranslation(new Vector3f(-
    0.6f, 0.0f, -0.6f));
12. objRotate = new TransformGroup(transform);
13. objRotate.setCapability(TransformGroup.ALLOW
    _TRANSFORM_WRITE);
14. objRotate.setCapability(TransformGroup.ALL
    OW_TRANSFORM_READ);
15. objRotate.setCapability(TransformGroup.EN
    ABLE_PICK_REPORTING);
16.
17. objRoot.addChild(objRotate);
18. objRotate.addChild(new ColorCube(0.4));
19.
20. pickRotate = new
    PickRotateBehavior(objRoot,canvas,
    behaveBounds);
21. objRoot.addChild(pickRotate)
22.
23. // Add a second ColorCube object to the scene
    graph
24. transform.setTranslation(new Vector3f( 0.6f,
    0.0f, -0.6f));
25. objRotate = new TransformGroup(transform);
26. objRotate.setCapability(TransformGroup.ALLOW_T
    RANSFORM_WRITE);
27. objRotate.setCapability(TransformGroup.ALLOW_
    TRANSFORM_READ);
28. objRotate.setCapability(TransformGroup.ENABL
    E_ PICK_REPORTING);
29.
30. objRoot.addChild(objRotate);
31. objRotate.addChild(new ColorCube(0.4));
32.
33. // Java3D perform optimizations on this scene graph
34. objRoot.compile();
35.
36. return objRoot;
37. }
```

V. SUMMARY AND CONCLUSION

The users will share 3D objects on the virtual space in Web-based Collaborative Design System and they must be able to pick the objects which they want to manipulate.

In this paper, picking is implemented not only by computing intersection of mouse pointer with the objects of the virtual world, but also by using capabilities and attributes of scene graph node, by setting bounds intersection testing instead of geometric intersection testing, by limiting the scope of the pick testing, using Java 3D.

Node-based picking methods can not need the arithmetic computation of picking comparing with ray-based picking methods, and bounds-based picking can reduce much more load of computation than geometry-based picking. Moreover, we can pick 3D objects effectively and easily considering the scene graph with hierarchy.

REFERENCES

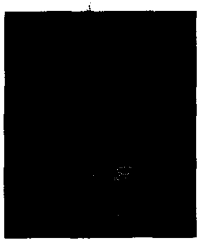
[1] F. Faure, C.Faisstnauer, G.Hesina, "Collaborative animation over the net", IEEE 1999, p107-p116.

[2] Icgiro Hagiwara and Shinsuke Noda, "Homotopical Modeling as the Basis of New CAD Standard Homotopy CAD for Collaboration Engineering", IEEE, 1999, p231-p237.

[3] Bo-yul Yoon, Eung-kon Kim, "A Study on Web-based Collaborative CAD System", The Journal of KIMICS (Korean Institute of Maritime Information & Communication sciences), Vol.4. No.4, 2000.

[4] Woo-suk Joo, "3D Computer Graphics", Green Press, 1999, p63-p65.

[5] Denis J Bouvier, "Getting started with the Java 3D API", Sun microsystems, 2000, http://www.javasoft.com/products/java-media/3D/collateral/



Dong-Hyun Kim
Received B.S. degree in computer engineering from Chosun University, Kwang-ju, Korea, in 1986, the M.S. degree in Computer engineering from Gwang-woon Univ. in 1992, and the Ph.D. degree in Computer Science from Chosun Univ. in 2002.

He worked for Honam Branch, HyunDae Electronics Co. as a chief executive from 1989~1996.

Since 1996, he is currently an associate Prof. of Dept. of Computer Information, Suncheon Cheong-am College, Suncheon, Korea.

He is interested in Multimedia, EC, Digital Contents & its applications.

Dr. Kim is a member of the KIMICS.

Bo-Yul Yoon
Received the B.A. degree in English Linguistics from Chonnam University in 1984, the M.E. degree in Computer Engineering from Chosun University in 1988, and the D.S. degree in Computer Science from Sunchon University in 2002.

He taught the students English in Sunchon Maesan Middle School from 1984 to 1996. Since 1997, he has been teaching the students in Sunchon Maesan High School. He has also managed the computer work as the chief of the department of Educational Information.

He is interested in Computer Graphics and CBI(Computer Based Instruction). Dr. Yoon is a member of the KIMICS.

Eung-Kon Kim
Received the B.S. degree in electionic engineering from Chosun University, Gwangju, Korea, in 1980, the M.S. degree in electronic engineering from Hanyang University, Seoul, Korea in 1987 and the Ph.D. degree in Computer engineering from Chosun University in 1992.

He is currently a professor of Department of Computer Science, Sunchon National University.

He is interested in Computer graphics and its applications.