ISSN No. 0976-5697

Volume 2, No. 4, July-August 2011

International Journal of Advanced Research in Computer Science

CASE STUDY AND REPORTS

Available Online at www.ijarcs.info

Various object picking algorithms in Non Immersive Virtual World-A Report

K.Merriliance*
Lecturer in MCA Department,
Sarah Tucker College,
Tirunelveli, India.
merriliance@gmail.com

Dr M.Mohamed Sathik
Associate Professor in Computer Science Dept,
Sadakathullah Appa College,
Tirunelveli, India.
mmdsadiq@gmail.com

Abstract: This paper presents a method for picking up an indicated object in a Non Immersive Virtual environment. First the user should indicate a target object and provides the system with a task instruction on how to get it. The system acquires geometric information about the target object and constructs a Virtual environment model and the system finds a main point based on evaluation using the acquired information. An important and advantageous feature of this scheme is to perform the object picking task through simple clicking operations, and the user can execute the task without exact models of the target object and the environment being available in advance. A user study of these was performed which revealed their characteristics and deficiencies of objects in the non immersive virtual world, and led to the development of a new class of techniques. Object selection is a primary interaction technique which must be supported by any interactive three-dimensional virtual reality application. Although numerous techniques exist, few have been designed to support the selection of objects in dense target environments, or the selection of objects which are occluded from the user's viewpoint. We presented a limited understanding on how these important factors will affect selection performance hence to realize these operations it is necessary to use interaction techniques that would allow us to accomplish given type of interaction better and faster.

Keywords: Non Immersive Virtual Environment, object picking, Pick ray, positioning, Pick sphere.

I. INTRODUCTION

The advance of computer graphics knowledge and technology, itself tied to the enormous increase in processing power and decrease in cost, together with the development of relatively efficient and unobtrusive sensing devices, has led to the emergence of participatory immersive virtual environments, commonly referred to as "virtual reality" Non-immersive virtual environment, as the name suggests, are the least immersive implementation of VR techniques. Using the system, the virtual environment is viewed through a portal or window by utilizing a standard high resolution monitor. The manipulations of virtual objects have been proposed as special human computer interfaces for the scientific visualization process. In the earliest stage physical objects are tracked on the table by using markers attached to them. The physical objects can then be used for interactions on the table. Picking is a simple technique for capturing which 3D object the user clicked on with the mouse. It can be used to discover the object at any X, Y position on the screen, when the user clicks the mouse. Picking means objects can be selected from the Virtual world. In a Virtual world there could be thousands of objects. Here we have explained as how the virtual object to be picked. The transformation of the 2D mouse position to a 3D location in the virtual world is an important process in picking. Because computer display is really a regenerated 2D view of the underlying 3D world.

Picking the correct 3D object is to find the direction of the picking shape and the virtual world coordinates of the starting point. It is the process of determining which object in the world is selected through user interaction, usually a mouse click. Typically, picking involves the selection of a 3D object from its 2D projection on the screen by pointing and clicking the mouse. The process involves casting a ray into the world from a selected point on the virtual Environment. Any objects the ray intersects after passing the VE are potential objects for selection. A ray is a line with a start point and a direction. Picking makes extensive use of Bounding Volume's intersection method. The object position follows the mouse cursor position closely, while the object always stays in contact with other surfaces in the scene. When the object moves in the virtual world its system of coordinates(x, y, z) moves with it. Therefore the position and orientation of object vertices in the object. System of coordinate remains invariant, regardless of the object position in the scene.

A. Object Picking

The ability to navigate through a world seen only on your computer screen or through a special headset opens the door for an incredible variety of experiences. It adds the ability to navigate through a virtual environment or the capability of picking up objects, or otherwise interacting with objects found in the virtual environment. Now that we have the ability to put objects in virtual world and move around them, it would also be nice to be able to choose which object we are focused on. One method of doing this would be to click on the object on the screen and have the camera refocus itself around that object. This method of choosing an object from the screen with the mouse is called picking. Picking is an important interaction technique in graphics applications. Things which are inside the Virtual Environment are known as objects. The objects computer-generated stereo images are projected onto the surface of the workbench. Interaction means objects in the scene can be manipulated. we implemented operations on objects' topology and geometry, and choosing and moving around objects. In the Virtual world it is important that the pick generates exact information using the accurate model and supports the identification of topological entities such as faces, edges and vertices. Using a mouse to select objects in virtual world is a little tricky because the mouse gives only 2D pixel coordinates which must be some how converted to 3D coordinates. Often the third dimension brings along an extra complexity in terms of completing the task in an

efficient and comfortable manner. In fact, the mouse location on screen represents an infinite number of points in world space which are projected on to a single point in screen space. In a Virtual environment, there may be more than one object under the mouse pointer when it is clicked. Normally, the user's intention is to select the object which is visible at this point. The ray will be compared against every object. If more than one object is intersected, the object nearest the viewer is selected. We may pick object within a specific bound which can be updated dynamically depending on changes in the view point of a user with in the virtual world using mouse. Clicking a mouse will create an appropriate picking bound at a 3D coordinate associated with the current mouse position.

We would like to have a way for the user to know which object is currently manipulating. Our basic idea is to disable the bounding box on the old current object when the mouse is first clicked, then enable the bounding box as soon as we have the new object. This approach, which utilizes other structures in the scene, typically uses a ray from the eye point through the current pixel to identify the first intersection point with the scene. This intersection is then used to compute the position of the object. These behaviors are then used to constrain objects to particular places in a scene. A ray along the current mouse position is then used to find the places in the scene where the constraints are fulfilled and the object is close to the cursor position. Therefore, we keep the pick ray connected to the object, but gradually straighten the ray every time the movement of the user's hand decreases the angle to the object, whereas the object's position is unchanged. The first part of picking is simply getting the mouse clicks and sending them on to our scene. The second thing is for getting the Scene to pick all of our objects and to write the next part of the picking function, converting the 2D point into a 3D ray by projecting it using an inverse matrix. It will also set the clicked on object to be active in the scene so we can access it and know what was clicked.

II. PICK SPHERE

Selecting an object is easy for the user to reach out until the ray intersects the target object. An object is selected when this box intersects the object's bounding box. Here we have considered the shape of the box as sphere which is referred here as pick sphere. A pick in virtual world is normally carried out as pick ray. The computer projects a ray from the extended finger; if the ray intersects the object it is captured. 2Dmouse pointer defines the ray. The normalized ray direction that projects infinitely into the scene is then calculated. The closest intersected object is retrieved. In cases where the hand is used as a mouse, the data glove serves as an input source for a real-time, animated computer graphics model of the hand. The virtual hand moves around in response to the user moving his or her hand and may intersect with objects or may project a selection ray from an extended finger.

The objective is to grab an object by pointing a finger at it. To solve this problem, this paper presents a new system for picking up an indicated object in a virtual environment. Due to its extension it is easy to pick topological entities and the transparency of the sphere renders intersections between the model and the pick sphere obviously. First the user should indicate a target object and provides the system with a task instruction on how to get it. The system acquires geometric information about the target object and constructs a Virtual environment model and the system finds a main point based on evaluation using the acquired information. Because the object picking depends on the object's shape, situation inside the virtual environment.

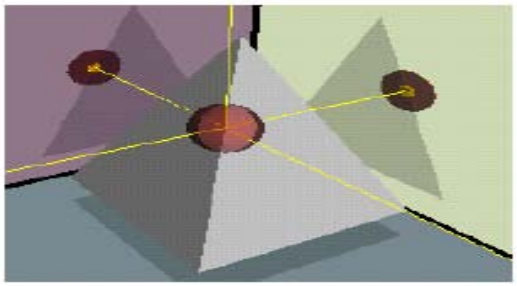

Figure 1: 3D cursor and its pick sphere

It is the most basic picking shape and will pick an object in the same way as a penetrating a ray or radiation. Here we obtain local mouse and eye position from the view plane to 3D coordinate. In this paper, we perform the exact object-space-based ray-sphere intersection test in a geometry shape by taking advantage of its geometric processing capability. Ray is projected into the virtual world from the position of the mouse pointer. Intersection of this ray with the objects of the virtual world is computed. The visual object intersecting closest to the virtual environment is selected for interaction.

The object picking is defined as the sphere between the user's eye point and the cursor. This fact improves the sphere-casting by rotating movements of the ray by simple translations of the cursor. Selecting objects via ray-based approaches have many ambiguities. If the ray does not hit any object the selection is underspecified. If several objects are intersected by the selection is over specified. Both cases should be modified. Our goal was to find out if these methods allow users to interact better and faster with the virtual environment. We also wanted to discover what problems they bring along and how we can handle them.

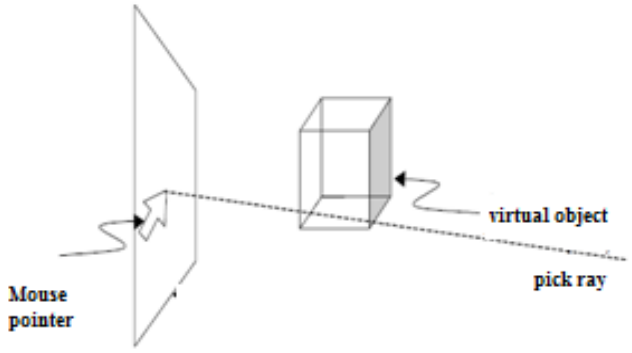

Figure: 2 Projection of Pick Ray in the Virtual World

A. Pick Ray

Creates a Pick Ray with origin and direction of (0, 0, 0) and also specifies the 3D origin and a vector 3D direction. Intersecting an Object to test for intersection of an object, the ray needs to be compared with the bounding volume of the object. This would necessitate applying the object's world transforms to the bounding volumes. It is usually more efficient to apply the inverse of the world transform to the near and far points. Ray tracing gives some of the most

realistic results of any rendering technique. Instead of building a scene polygon by polygon, ray tracing passes a ray from the eye through each pixel into the scene. The path of the ray is traced to see what object is strikes first and at which point .From this point further rays are traced to all the light sources in the scene in order to determine how this point is illuminated. Then the contribution to surface of each of these rays is calculated. The view port is sub-divided by a grid of rows and columns, the number of which is determined by the required image resolution.

For each square on the grid, a ray is built which extends from the eye through the centre of the square and into the scene. This ray is tested against all objects in the scene to determine which objects are intersected by the ray and gives a list of points of intersection. The list is sorted and the intersection nearest the eye is determined. This point will be used for further processing. In order to find the point of intersection of a ray with an object, we reverse the transformation that created the object from a primitive and apply that reversed transformation to the ray. Then find the points of intersection of the transformed ray and the original primitive. If we have a scene with many objects, for each ray from the eye we need to check for intersections with all objects. However each ray will usually strike a small percentage of the objects in a scene. An extent defines a simple area around each object which can be quickly tested for intersections. Each object is projected onto the view plane and the minimum bounding rectangle for each object is calculated before ray tracing .The minimum bounding rectangle is determined by finding the maximum and minimum x and y co-ordinates for the virtual environment. Each ray is then tested to see if it intersects the minimum bounding rectangle, if it does not; the object cannot possibly be intersected by the ray and is ignored.

III. RAY INTERSECTION TEST IN THE SPHERE

The ray tracing is the calculation of intersection between a ray and objects in the scene. All of this equation begins with the parametric equation for a line. A line from the point A=(x1, y1, z1) to point B=(x2, y2, z2). The two end points of our ray need to be converted to world coordinates by applying the inverse of the view and projection matrices.

Consider a Ray: Ray (t) = o +td, t>=0

Sphere: |P - C|2 - r 2 = 0

To perform a ray-sphere intersection test we need a ray with a known point of origin O, and direction vector d. A sphere with a known centre at a point C and a known radius r Given the above mentioned sphere, a point P lies on the surface of the sphere if

(P – c). (P-c)=r2 ---------- (1)

Given a ray with a point of origin O, and a direction vector d

Ray (t) = o +td, t>=0

We can find the t at which the ray intersects the sphere by setting ray (t) equal to P

(o +td-c). (o +td-c)= r2---------- (2)

To solve for t we first expand the above into a more recognizable quadratic equation form

(d.d)t2+2(0-c).dt+(o-c).(o-c)-r2=0 or At2 + Bt + C = 0

where A = d.d ;

B = 2(o-c).d;

C = (o-c).(0-c)-r2;

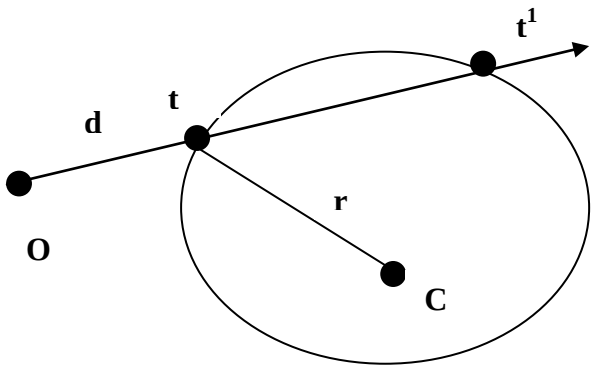

Figure 3: Ray-Sphere Intersection

This can be solved using a standard quadratic formula. Note that in the absence of positive, real, roots, the ray does not intersect the sphere. In the object space of a sphere it is centered at origin, meaning that if we first transform the ray from world space into object space, the mathematical solution presented above can be simplified significantly. The following Java code is an example how the simplified version of the ray/sphere intersection described above might be implemented.

```
Public class Ray
{
float o,d,c;
Ray(float x, float y, float z)
{
o=x;
d=y;
c=z;
}
Boolean  Sphereintersect(const Ray& r1, float* t)
{
   float a = dot(r1.d, r1.d);
   float b = 2 * dot((r1.o-r1.c),r1.d);
   float c = dot((r1.o-r1.c), (r1.o-r1.c)) - (r * r);
   float d = b * b - 4 * a * c;
   float dis, t0,t1,q;
      if (d < 0)
      return false;
else
   dis = sqrtf(d);
      if (b < 0)
      q = (-b - dis)/2.0;
   else
      q = (-b + dis)/2.0;
     t0 = q / a;
     t1 = c / q;
     if (t1 < 0)
      return false;
     if (t0 < 0)
   {
      t = t1;
      return true;
   }
      else
   {
      t = t0;
```

```
      return true;
   }
}
Class demo
{
Public static void main(String args[1])
{
Ray r1=new Ray();
Ray obj=new Ray(1.0,5.0,10.0);
Boolean b=r1.sphereintersect (obj,5.0);
{
System.out.println (“Intersection point=”,+b)
If (b==1)
  System.out.println (“ray intersects the sphere”);
  Else
System.out.println (“ray misses the sphere”);
}
}
```

If B2-4AC<0, then ray and sphere do not intersect because the intersections are imaginary numbers .If B2-4AC=0, then the ray intersects the sphere at one point. If B2-4AC>0, then the ray intersects the sphere at exactly two points Finally we can find the point of intersection then check If the center of the sphere is inside the enlarged object bounding box all object faces are checked. Again, the procedure starts with enlarging the face bounding boxes by the pick radius and checks if the cursor center is inside the enlarged box. The vertices are checked by simply comparing there position with the cursor position and the pick radius.

Finally, if all these checks failed, an is-inside check is performed to see if the cursor is in the interior of an object without intersecting any of its topological elements. The shape of the virtual object is determined by their 3D surface, which can be described in many the object picking depends on the object’s shape, situation in the environments.

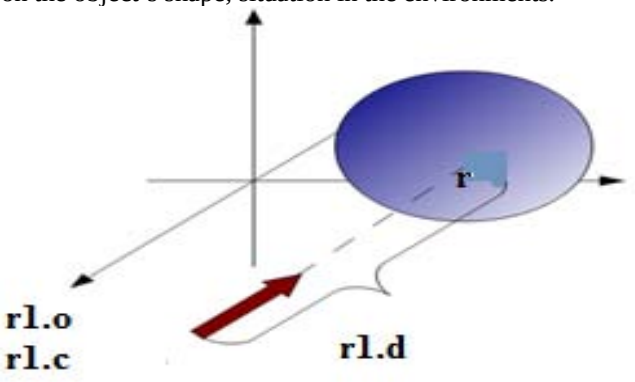

Figure: 4 Pick Ray test to find the depth state among visible objects.

The solutions to the quadratic equation give the time of intersection. If this gives two real solutions, then the ray has intersected the sphere twice. If only one solution is found, then the ray is a tangent to the sphere. If no real solutions are found, then the ray does not intersect the sphere. Once the intersection times are known, they can be inserted into the original equation of the ray to calculate the co-ordinates of intersection. A sphere is a common shape to use and is centered on the object and has the smallest radius which allows it to enclose the entire object. To determine if a ray passes through an object, first test for intersections with its bounding sphere. This is usually a lot simpler than testing for intersections with complex objects. However, some objects, are not suitable for sphere bounding, also, calculating the minimum bounding sphere can be difficult. The object space based ray-sphere intersection test is implemented in a highly parallelized geometry shader. When we applying the occlusion queries, only a small number of objects are rendered in subsequent layers, which accelerates the picking efficiency.

The algorithm is as follows

Input: Cursor point and Virtual scene

Output: Ray intersection point, object id

Step 1: Compute the picking ray intersection point and the direction in the view coordinate system.

Step 2: Set the depth states to false. The bounding boxes consist of the visible objects. We pass a query for each object.

Step 3: The bounding boxes of all sub-objects whose corresponding query returns true. Again we issue a Boolean query for each sub-object during the pass. Then set the depth state to true.

Step 4: Consider the actual spheres whose queries have been returns true then pass query for all sphere shaped objects.

Step 5: If the occlusion query returns true, the picking information returns the one-pixel-sized target data; otherwise, no object is picked. Sphere outside the view ray is discarded, and only the closest sphere is needed.

Step 6: If the query passes, the sphere with the minimal distance from the eye-point is picked and its intersection information can be retrieved from the target.

After this algorithm performs the object-space-based ray intersection test in the geometry shader, output a point with picking information if the sphere is intersected. The x and y-components of the intersection point are set to 0, and the z-component is assigned as the depth value of the point.

Output the picking information directly in the pixel shader. When the user clicks the mouse, the screen coordinates of the cursor are transformed through the projection matrix into a view-space ray that goes from the eye-point through the point clicked on the screen and into the screen. A 3D picking problem can be reduced to a problem of determining the object that intersects at a given point the eye-ray fired from the center of projection through the pixel’s center into the un projected scene. This problem can be solved by determining all the objects that intersect the ray and performing point-in-solid inclusion tests to find out which object contains the specified 3D point. It may reduce the problem to a one-dimensional problem by determining the intersection intervals along the eye-ray for each object.

Then, the object that is pointed at can be obtained with a point-in-segment inclusion test this is because when no bounding box intersects with the picking ray, our approach will not render the actual spheres and return false directly. It may reduce the problem to a one-dimensional problem by determining the intersection intervals along the eye-ray for each object. Then, the object that is pointed at can be obtained with a point-in-segment inclusion test. For handling a sphere shape, the problem is reduced to the computation of the nearest point on the shape, most of which require potentially time consuming point-and-shape or the ray-and-shape intersection algorithms. It allows

accurate fine positioning in 3D-space.Reduced to a simple rounding of the device position to a point of a specified grid.

A. *Advantages*

a. Cost of performing intersection test will be totally application-independent and very low.
b. Selection Supports both visible and occluded target.
c. Accurate fine positioning in 3D space
d. It is possible to minimize the costly nearest point calculations.
e. Results can be easy reused if these have already acquired at once.
f. No additional information on the object and environment are needed.
g. Ray intersection test is performed through searching the best solution from many other feasible solutions.

When we analysis these concepts in the view of selection and cost of performing an intersection test (see Fig:5), picking using Ray intersection test algorithm is always greater than the average cost among other algorithms.

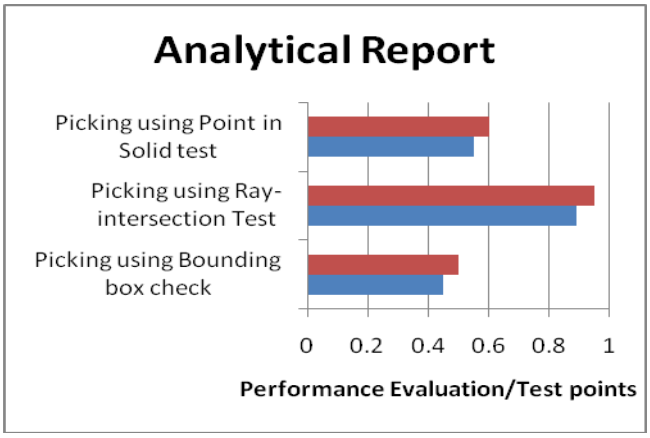

Figure 5. Average cost of performing an intersection test

IV. CONCLUSION

We have presented an efficient algorithm for intersection tests between a picking ray and multiple objects in non immersive environment. Furthermore, the Analytical Report provides some new areas for future developments of usability evaluation methods. We discussed an implementation of the proposed algorithms, based on these guidelines, we present new forms of the pick ray casting and 3D picking using sphere shape algorithm techniques, which are augmented with positioning, selection and the average cost of performing intersection test feedback, to support selection within the Virtual environments. We have demonstrated the performance of the three methods, with retrieval results. Furthermore, the results showed that our new techniques adequately allowed users to select targets which were not visible from their initial viewpoint. In this paper, we perform the exact object-space-based ray-sphere intersection test in a geometry shader by taking advantage of its sphere shaped object. The overall approach makes no assumptions about the object's motion and can be directly applied to all sphere models. This paper has given a good starting point for designers and can be directly applied to all sphere models.

V. REFERENCES

[1] Cleber S.Ughini, Fausto R.Blahco "3D interaction Technique for accurate object selection in Immersive Environment". Proceedings of the 1997 symposium on Interactive 3D
[2] Foley, J.D., van Dam, A., Feiner, S.K., and Hughes, J.F. Computer Graphics: Principles and Practice. Addison-Wesley, 1990.
[3] Sebastian Knodel." Navidget for Virtual Environments" Proceedings of the 2008 ACM symposium on Virtual reality software and technology.
[4] Williams, G., McDowell, I. E. and M. T. Bolas. Human scale interaction for virtual model displays: A clear case for real tools. In Proc. Of The Engineering Reality of Virtual Reality.
[5] Wu, 2002. shin - ting, marcel abrantes, daniel tost, and harlen costa batagelo "picking for 3d objects".
[6] Wu, X. 1992. A linear time simple bounding volume algorithm. In Graphics Gems III, David Kirk, Ed., chapter VI, pages 301–306.Academic Press, San Diego, CA.
[7] Antony Sted."Evaluating Effectiveness of Interactions technique across immersive virtual Environment Systems". October 2005, Vol. 14, No. 5, Pages 511-527 Posted Online March 13, 2006.
[8] Jesper Kjeldskov."Combining Interaction technique and Display types of Virtual Reality". J Kjeldskov - Proceedings of OzCHI, 2001 - cs.aau.dk
[9] Cleber S.Ughini, Fausto R.Blahco "3D interaction Technique for accurate object selection in Immersive Environment". Proceedings of the 1997 symposium on Interactive 3D.
[10] Foley, J.D., van Dam, A., Feiner, S.K., and Hughes, J.F. Computer Graphics: Principles and Practice.Addison-Wesley, 1990.
[11] Neider, J., Davis, T., and Woo, M. Open GL Programming Guide: The Official Guide to Learning OpenGL, release 1. Addison-Wesley, 1993.
[12] Mine."Virtual Environment Interaction Technique" 1995.