[Volume 1 issue 9 Oct 2013]
Page No.238-244 ISSN :2320-7167

INTERNATIONAL JOURNAL OF MATHEMATICS AND COMPUTER RESEARCH

The role of point and ray picking shape Algorithm in picking object in Non Immersive Virtual world

K.Merriliance [1], Dr.M.Mohamed Sathik[2]

[1]Lecturer, Department of MCA,
Sarah Tucker College (Autonomous),
Tirunelveli– 627007, Tamilnadu, India
merriliance@gmail.com

[2]Principal, Sadakathulla Appa College (Autonomous),
Tirunelveli– 627011, Tamilnadu,India
mmdsadiq@gmail.com

Abstract: ***Picking is an essential task and an interaction technique in any interactive graphics application. In the field of VR-like systems, conventional point and ray-picking algorithm is still widely used. The algorithm gains its performance by utilizing ray intersection checks to minimize the number of costly nearest point calculations. To reduce these costly operations on the precise VR world, we introduce an algorithm that performs ray intersection test during user interactions. The algorithm is gaining its performance by reducing these checks dramatically compared to the naive approach where the checks are carried out with each 3D cursor movement. The user can also query the system to see how they can perform a specific task. In this paper I wish to illustrate how the point and ray picking shape algorithm works and helpful for the effective virtual object picking process.***

Keywords: object picking, ray picking shape, Non immersive VE

1. Introduction

Non Immersive VE calls 'artificial reality,' requires no personal hardware. It can combine quite 'real' objects - tree, whatever – with 'unreal' ones, such as animatronics figures, holograms, disguised objects, optical illusions, etc environment - a room, a simulator, etc. - where normal sensory cues are not cut off, but are supplemented by additional sounds, images, or other sensations. The advantage of this type of VR is that it can be a communal experience, with many people participating at once. In a sophisticated enough environment, it might be difficult to determine who the "real" participants are. In the virtual environment there are virtual objects whose attributes have to be represented and made interactive with the user by sophisticated software mechanisms. Different parameters define the perceptual shape, status and behaviour of the virtual objects which are manipulability by the user. Other attributes define the virtual environment itself and are responsible for light, atmospheric, and physical effects which can strongly differ from the user's existing real world conditions. One of the most basic elements in the complete taxonomy of three-dimensional interaction is that of object picking.

That is, indicating one specific object in the virtual world. Nearby objects can be directly picked by manually positioning the spatial interaction device precisely inside its observed graphical representation. The user can position the device to control the origin of direct interaction, while the device rotations control the direction of the ray for remote interaction. To pick an object we can use a direction picking or an item picking. In direction picking the user chooses a direction to pick an object which can be picked whether it is within reach or not. There are several ways. Item picking includes all ways to pick an item from an enumerated list and can be done by several methods: contact-pick, point to pick, 3d-cursor pick, aperture pick, menu-pick, pick in a virtual world name to pick, etc. Interaction with the virtual environment can occur by keyboards, mice and trackballs or may be enhanced by using 3D interaction devices environments, user interfaces and interaction, and important

technologies used in VEs. Interactivity consists of functions for navigation, manipulation, picking of objects or information, and communication with other users within the virtual world. A virtual environment is composed of objects, which may be brought to life through their behavior and interaction. Some objects will be static and have no behavior. Some will have behaviour driven from the real world, for example by a user. Alternatively, object behavior may be procedurally defined in a computer program. By replicating object behavior we reduce the dependency on the network and therefore make better use of increasing responsiveness.

2. Object picking process

The concept of non immersive Virtual Reality is a three-dimensional, computer-generated environment that allows for single or multiple users to interact, navigate, respond, and experience a synthesized world modeled from the real world. It incorporates the interaction within a VE or virtual world by the help of various peripheral devices in order to achieve a multi-sensory and multi-modal experience. The user interaction in the system is divided into two different parts. The first one implements the implicit interaction and is used for inspection of 3D models and for navigation through them. The second one implements other user interaction possibilities like object picking and movement or menu handling. For instance if the non immersive virtual environment contains a car, the car door can be given dynamic properties and constrained to allow them to rotate about their hinges through a specified angle. The user opens a door by moving the icon of the 3D mouse towards the position of the doors handle. When the user activates a button on the mouse, the picking of the door is confirmed. This is called picking. 3D objects, which are determined by the number of polygons, shading, lighting, texture resolution etc. Picking means objects can be picked from a Virtual world there could be thousands of objects. 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.

Early systems replicated object states but not their behavior. Each state change to any object was sent across the network to every replication. The development of object picking algorithms has become a popular area of research in recent years with many applications developed. Since, many picking algorithms are being proposed, objectively measuring the efficiency of a picking algorithm has become necessary for comparing the performance of object picking algorithms. 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 3D world using mouse. Clicking a mouse will create an appropriate picking bound at a 3D coordinate associated with the current mouse position. Object within a bound is picked. When no bounding box intersects with the picking ray, no object is picked. 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. The first thing we have to write code for setting up the framework for picking and we need to do is have some input from the mouse to play with and see if an object in our scene was clicked on. 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 we will create by taking a few settings. The method for finding out whether an object was hit by the ray is much simpler to implement. All we have to do is convert the ray into the local coordinates of the model we are checking and have the built in Mesh. Intersect function tells us whether we have hit home or not. Now our engine will successfully test any mesh based objects that we load and report back with a true or false whether that object was clicked on or not. It will also set the clicked on object to be active in the scene so we can access it and play with other things once we know what was clicked.

The shape of a 3D object is commonly represented by a polygonal mesh, in which the vertices of the polygons provide depth information on the object surface. While smooth surfaces are represented by a smaller number of vertices, surfaces with more detail are represented by a higher number of vertices. Given an adequate number of polygons, we are able to describe complex geometry with high precision. Nevertheless, complex models demand large storage, and long computing and rendering time, which may not be suitable for interactive visualization. The performance of graphics systems can thus be measured in terms of the number of polygons (or vertices) updated per second. In other words, the number of vertices in a model is a good evaluation metric.

3 Basics of point and ray picking shape algorithm

Picking the correct 3D objects is the direction of the picking shape and the virtual world coordinate perspective projection. This method of choosing an object from the screen with the mouse is called picking. The first thing we have to write code for setting up the framework for picking and we need to do is have some input from the mouse to play with and see if an object in our scene was clicked on. 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 we will create by taking a few settings. The method for finding out whether an object was hit by the ray is much simpler to implement because DirectX does a lot of this for us. All we have to do is convert the ray into the local coordinates of the model we are checking and have the built in Mesh. Intersect function tell us whether we have hit home or not. Now our engine will successfully test any mesh based objects that we load and report back with a true or false whether that object was clicked on or not. It will also set the clicked on object to be active in the scene so we can access it and play with other things once we know what was clicked. An important task of the interactive 3D environment component is to find geometric correlates for the semantic meaning of deictic

terms, such as “that,” “here,” and “there,” as well as to facilitate picking of objects. The ray is de-fined by the virtual camera position and the 2D mouse pointer on the image plane. By intersecting the objects in the scene with the ray, it is determined which one is picked? Not only objects but also their topological elements, i.e. faces, edges, vertices, can be picked, which is especially important in CAD. In virtual environments 3D picking and/or grabbing is typically performed by bounding box checks or collision detection taking the position of a virtual hand and the objects into account.

Pick Ray: It is the most basic picking shape and will pick an object in the same way as a penetrating a ray or radiation. The picking ray extends in the scene infinitely in a specific direction and an object intersected will be picked. Here we obtain local mouse and eye position from the view plane to 3D coordinate. The normalized ray direction that projects infinitely into the scene is then calculated. The closest intersected object is retrieved. The user clicks on the screen at point A. The object that is picked is to determine as there is only one along the projected vector indicated by the dashed line. A simple casting of the vector into the scene graph will reveal what object has been picked and our pick system will return a reference to it. When picking an object, typically we are not interested in the entire scene graph tree to that object, just the actual object that was picked. Since the object is a visible item, it has no children and the end of the path is the picked object. The picked item can never be an item in the middle of the path. Using the pick ray it picks many objects. How can we determine the exact object? Pick ray segment overcome this problem. This method maintain a certain degree by using only a ray with appropriate starting and ending point. We have to provide the eye direction, choose two end points of the pick segment. Picking the correct 3D object is to find the direction of the picking shape and the virtual world coordinates of the starting point.

4 ray picking shape algorithm

Getting target object is difficulty for users interacting with these environments. With ray picking we usually simplify a scene into bounding spheres or boxes. This makes the calculation a bit easier than testing all of the individual triangles. We don't need to create a 3d sphere out of triangles that can be rendered; we just represent the sphere as a simple function. The premise is that we have a mathematical formula for the points along a ray, and a mathematical formula for the points on a sphere. If we substitute the points on the sphere with the equation for points on a ray, then we get the intersection of points that are common to both. It's interesting to do this technique now because it shows us how we can use the transformation pipeline in reverse; from 2d screen to a 3d word space by using the inverse of our matrices. In a 3D environment, there may be more than one object under the mouse pointer when it is clicked. Normally, the user's intention is to pick the object which is visible at this point. The general approach will be to use the mouse coordinates to generate corresponding points on the near-plane and far-plane in world coordinates. These points will form a ray. The ray will be compared against every object. For intersection if more than one object is intersected, the object nearest the viewer is picked. 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 3D world using mouse. Clicking a mouse will create an appropriate picking bound at a 3D coordinate associated with the current mouse position.

Object within a bound is picked. When no bounding box intersects with the picking ray, no object is picked. We will be making it so that you can "pick up" and move objects after you have placed them. You can use the showBoundingBox method to create a box around objects. 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 3D object. 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. 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, and the basis for the enthusiasm for the technology becomes readily apparent. 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. The first thing we have to write code for setting up the framework for picking and we need to do is have some input from the mouse to play with and see if an object in our scene was clicked on. The first part of picking is simply getting the mouse clicks and sending them on to our scene. The second thing is to convert the 2D point into a 3D ray by projecting it using an inverse matrix we will create by taking a few settings. The method for finding out whether an object was hit by the ray is much simpler to implement. Intersect function tell us whether we have hit home or not. Now our engine will successfully test whether that object was clicked on or not. It will also set the clicked on object to be active in the scene so we can access it and play with other things once we know what. It can be used to discover the object at any X, Y, Z position on the screen. The user clicks the mouse .It Captures the X, Y and Z position of the click.

Mouse picking, as the most intuitive way to interact with 3D scenes, is in many interactive 3D graphics applications, such as mesh editing, geometry painting and 3D games. This method is used to calculate the exact intersection information, that is, the barycentric coordinate in the intersected triangle. The mouse picking operation can be performed by an ordinary ray-object intersection test and accelerated by lots of schemes for high efficiency. It is possible to triangulate the bounding boxes of objects as strips and to cull away objects that are positioned out of the view. mouse picking operation takes the screen coordinate of the cursor and the scene to be as input, and outputs the intersection information, such as object id, triangle id, and even the barycentric coordinate of the intersection point.

Algorithm

Step 1: Once the user clicks on the screen, compute the picking ray origin and direction in the view coordinate system.

Step 2: After the view the bounding boxes consist of the visible objects.

Step 3: The bounding boxes of all sub-objects whose corresponding query returns true. Again we pass query for each sub-object.

Step 4: For all the surfaces in the 3D part do

1. Determine whether the mouse lies on the plane by calling the module Pick_Pt_Lies On()
2. If yes, Set the found Flag and store its distance to eye in the buffer

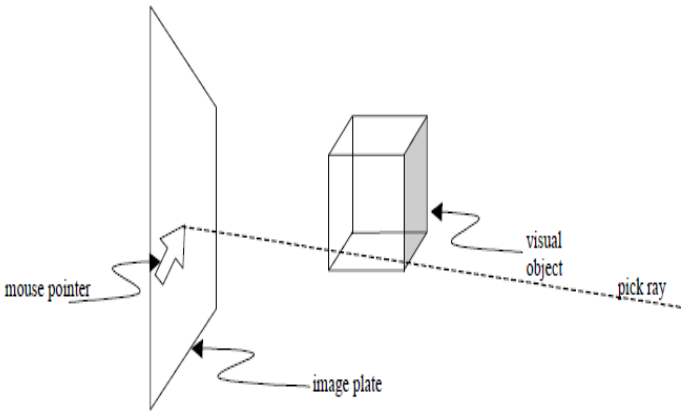
distance=sqrt((cx-bpx)2+(cy-bpy)2+(cz-bpz)2);cx,cy,cz-cursor point ;bpx,bpy,bpz-boundary point of an object.

3. Else go to the next surface

Step 5: Sort all the object by its distance

Step 6: Return the top most object from the buffer.

Lastly, if the occlusion query passes, the triangle with the minimal distance from the eye-point is picked and its intersection information can be retrieved from the target.



. Figure 2: ray picking in the non immersive Virtual World

This algorithm performs the object-space-based ray-triangle intersection test and output a point with picking information if the object 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. Then the point is passed and Output the picking information directly in the picked object as a result, the intersection information is obtained. The intersection test is conducted in the view space and if it is passed, the x- and y-components of its position coordinate are 0 because the render target used in our algorithm is only one-pixel in size. The z-component is the depth value which is obtained by

transforming the distance value into the projection space. Figure 2 show the projection of pick ray in the non immersive virtual world.Intersection of this ray with the objects of the virtual world is computed. The visual object intersecting closest to the image plate is picked for interaction while interacting with the closest visual object is the most common picking application; picking operations are not limited to picking the closest visual object. Some picking operations produce a list of all intersected objects. In some cases the interaction is not directly with the picked object, but with an object along the scene graph path to the object.

5 Conic volume picking algorithm

This algorithm uses bounding boxes to calculate conic volume intersections. Pick the object that corresponds to the pixel at the mouse cursor position. All pickable objects are rendered twice. The second render pass assigns unique object identifiers instead of colors. To adapt this method to 3D picking we change the projection and view matrices of the second render pass to correspond to the position and orientation of a pointing device. Hence, the perspective view frustum of the pointing device represents a conic picking volume. The opening angle of the picking volume can be adjusted by changing the field of view of the perspective projection. Visible pixels of the objects within the picking volume will be recorded by their identifiers in the off-screen buffer. Pixel based object statistics can then be applied to choose the active object. The user can pick objects that are located within spotlight's cone. To pick among more objects contained in the cone, it is possible to use a conic volume technique (Figure 3). It is a modification of ash light technique where the picking of objects within the cone is done using a collision plane.

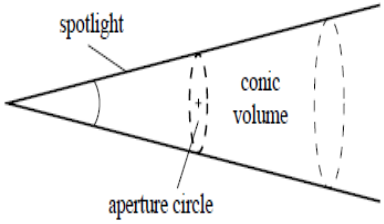


Figure 3: Conic volume technique in Object picking

The conic volume is determined by the position and orientation of the pointer's tip, its orientation and a spread angle. First, we determine which objects are within the range of this volume. For every object in the scene, every frame, we perform an "inside"-test of the object and the conic volume. In our current proof-of-concept we limit our test to the bounding box of an object, which is typical for the objects in our user tests. We transform the object's center point position and radius to the co-ordinate system of the pointer. In this co-ordinate system we can easily derive the projection distance on the ray (dr) and the distance from this projected point to the pointer's position (dp). Figure 3 shows a three-dimensional schematic overview of object P inside the conic volume. Point P lies on a two-dimensional cross section of the cone. This is a circular surface, with the ray defining both its center point and normal, while the radius is determined by the cone spread angle and distance dp.

dr = dist= sqrt((cx-bpx[i])2 + (cy-bpy[i])2 + (cz-bpz[i])2)

dp = z (2)

If the angle defined by these two distances, is smaller than the o to be inside of the conic volume. For these objects a scoring value will be calculated, while other objects will be ignored in the following stage. pening angle of our cone, we consider the object The Picking behavior enables picking objects in the scene. A pick in 3D is usually carried out as Ray Pick. The virtual camera position and the 2D mouse pointer on the image plane define the ray. By intersecting the objects in the scene with the ray, it is determined which one is picked.

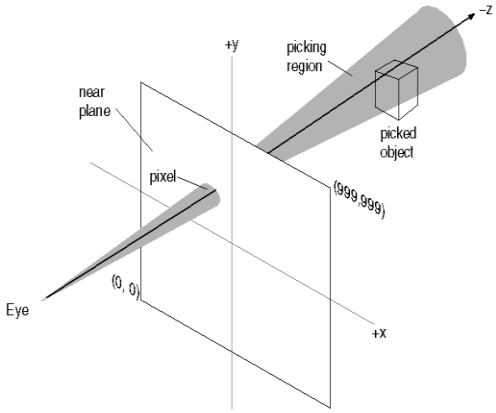


Figure 4: picking volume test

Objects available in the second phase is the basic interaction task corresponds to an indivisible unit of information that is meaningful in the context of the application. Typical composite tasks for interactive 3D applications are model construction and direct manipulations on this model. This component provides two basic functionalities: the maintenance of picking state and the identification mechanism. The identification occurs when a user "clicks" a pointing device over the output of a

camera. This process first detects whether that action occurred on top of some graphic primitive. If that happened, then the element that contains such primitive is located. The bounding box dragger allows direct interaction with its eight vertices, twelve edges, and six faces. In implementing our bounding box manipulator, we assigned scaling to vertices, rotation to edges, and translation to faces. Two consecutive positions of a vertex are mapped into a scaling factor. Two consecutive points over an edge are mapped into a rotation about an axis, being parallel to the picked edge and passing through the center of the bounding box. Each pixel position contains a value that is either zero or an integer number (ID) that corresponds to an object. Since we set up the projection and transformations according to the center pixel position c corresponds to the centre of the picking cone. A popular scoring metric for instance used by the projected distance d between the cone center and an object at pixel position p which is

$$d = \|c - p\|2.$$

Picking visual object using conic type of shape will be advantageous if the object has a cross section or size that is too small to be picked or intersected by a line. To illustrate the use of pick cone ray which has a picking shape that corresponds to a cone with infinite length. Smaller object to be easily picked. To make it easier to pick lines and points, the ray can be augmented to be a cone or a cylinder. Things that are picked must fall within this cone, as follows:

- For points and lines, if any part of the shape falls within this cone, it is picked.
- For all other shapes, the ray itself must intersect the shape for it to be picked.

The main benefits of this method are:

(1) It is easy to implement. Additional implementation code is kept to a minimum.

(2) It reduces picking ambiguity.

3) Bounding volumes can significantly reduce occlusion problem

4) This technique allows the easy integration of picking of non-transparent object

5. Ray picking shape algorithm vs conic volume picking algorithms

Ray-casting and cone-casting are by far the most popular distant picking algorithms. Attached to the user's virtual pointer there is a virtual ray or a small object. The closest object that intersects with this ray or cone becomes picked. This algorithm allows the user to easily pick objects at a distance, just by pointing at them. As the algorithm relates to a flashlight in real life, and since flashlights have no force feedback, to our opinion, introducing force feedback will not improve the interaction. The 'picking volume' is defined as the cone between the user's eye point and the aperture cursor. This algorithm in fact improves the cone-casting by reducing the rotation movements of the ray by simple translations of the aperture cursor. Another way a programmer can reduce the computation of picking is to use bounds intersection testing instead of geometric intersection testing. The determination of a pick using the bounds is significantly easier for all but the simplest geometric shapes and therefore, results in better performance. The measured time was analyzed with a paired t-test. Algorithms associated with this course, time complexity comparisons are more interesting than space complexity comparisons. A measure of the amount of time required to execute an algorithm. Feasibility determination of these algorithms is based on estimating the mean time. Time complexity expresses the relationship between the size of the input and the run time to pick an object. When comparing two algorithms that perform the same task, we often just concentrate on the differences between algorithms based on Average case for a successful search and pick the exact object from the virtual world. Detection Rate and Ray picking shape algorithm, which only measures the correspondence in the objects which are inside the bounding box and the objects detected by the picking algorithm with no reference on how the objects are picked in the non immersive virtual world. Thus, Ray picking shape algorithm provides information that allows for detection of errors that affect post-tracking processes Now suppose that we detect and measure features of the object U and represent them as Uj, j = 1, ... , N. where N is the number of objects. The distance may be either Euclidean or any weighted combination of features. In general, we compute the distance(dist) of the object from class j as given by

$$\text{dist} = \left[\sum_{j=1}^{N} (u_j - f_{ij})^2 \right]^{1/2},$$

Suppose that we have a scene g[i, j] and we wish to detect its an object f[i,j]. The impact of virtual object depends on the distance between the virtual scene and the viewer. Even though distance is an important factor in perceived quality, we eliminate this factor in the current version of our proposed evaluation metric by keeping a fixed distance. Then we scale objects to the largest possible size on a large monitor to allow observers to have better depth perception. An obvious thing to do is to place the scene at a location in an object and to detect its presence at that point by comparing intensity values in the scene with the corresponding values in the object. The sum of the squared errors is the most popular measure. In the case of object picking, this measure can be computed indirectly and computational cost can be reduced. MSE is a good predictor for view-independent perceptual quality of 3D objects. Mean Square Error (MSE) is commonly used as a quality predictor. MSE is defined as:

$$MSE = \sum_{i=0}^{M-1} \sum_{j=0}^{N-1} \frac{(P_0(i,j) - P_c(i,j))^2}{MN}$$

where p0 is the original virtual scene consist of set of 3D objects, Pc is the picked object, M and N are the width and height of the object respectively.

Matching type	MSE)	S.D
Ray picking shape algorithm	2.90	1.35
Conic volume picking algorithm	3.80	2.55

Table 1: Time needed to complete the task and precision of the resulting models.

The time to complete the task in the Ray picking shape algorithm found significantly lower compared to the time needed with a Conic volume picking algorithm in Object picking process. The mean square error was 3.80 (S.D. 2.55) in the conic volume picking algorithm compared to 2.90 (S.D. 1.35) at the Ray picking shape algorithm. Although Ray picking shape algorithm converges within even fewer iterations, the overall time is slightly more because one iteration takes more computation due to the additional parameter estimation. From this result, we recommend Ray picking shape algorithm for more robustness, and provides faster performance. SD is the standard deviation, of r at each picked object, where N is the number of objects.

$$SD = \sqrt{\frac{\sum_i r_i^2}{N-1}} \qquad \text{----------(1)}$$

The overall standard deviation is calculated as in Equation (1). Fortunately, evaluation errors in the results are small and do not change the overall property of the non immersive virtual scene. It is therefore important to note that Ray picking shape algorithm should only be used to estimate perceptual quality of objects for which geometry is an important component of perceived shape.

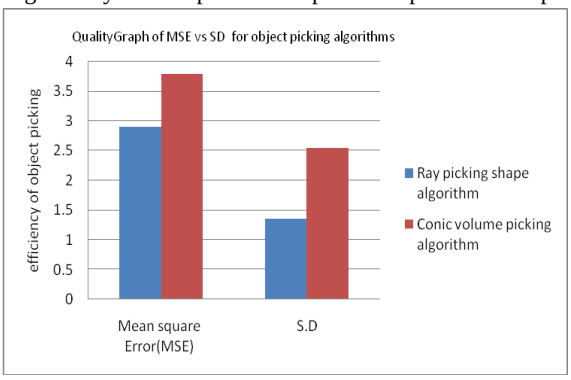


Figure 6: An analytical chart between Ray picking shape Algorithm Vs conic volume Algorithm

6. Conclusion

The performed user study implies that picking 3D object can be performed faster in ray picking shape algorithm without loss of precision. Furthermore the study has shown that even users without a strong virtual reality background can achieve good reliable results in non immersive virtual environments. Using the ray picking shape algorithm will further increase the robustness and reduce the computational costs. The results provide an initial understanding of how these factors affect selection performance. Furthermore, the results showed that our new techniques adequately allowed users to select targets which were not visible from their initial viewpoint. Our analysis indicated that our introduced visual feedback played the most critical role in aiding the picking task. Study results have shown the performance of the ray picking shape algorithm was to be slightly better than the performance of the conic volume technique. To efficiently evaluate the performance of the algorithm the features and behaviour of the object must be analyzed. From the results presented in the graph need to include information about object label, size, and location for performance evaluation of the algorithm is illustrated. The comparison algorithm performs maximization of the posterior parameters of all known objects. We plan to increase the strength of our findings by increasing the number of 3D objects in the model. We also want to check whethe ray picking shape algorithm was to be slightly better than the performance of the conic volume shape algorithm.

References

[1] D. A. Bowman and L. F. Hodges. An evaluation of techniques for grabbing and manipulating remote objects in immersive virtual environments. In Proc. SI3D 1997, page 35ff., 1997.

[2] Tobias Rick GPU Implementation of 3D Object Selection by Conic Volume Techniques in Virtual Environments

[3] Taylor II, R., et al., "The Nanomanipulator: A Virtual-RealityInterface for a Scanning Tunneling Microscope", ComputerGraphics Proceedings, ACM Siggraph, pp. 127-133, 1993.

[4] Sebastian Knodel." Navidget for Virtual Environments" Proceedings of the 2008 ACM symposium on Virtual reality software and technology.

[5] 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.

[6] Wu, 2002. shin - ting, marcel abrantes, daniel tost, and harlen costa batagelo "picking for 3d objects".

[7] 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.

[8] Foley, J.D., van Dam, A., Feiner, S.K., and Hughes, J.F. Computer Graphics: Principles and Practice.Addison-Wesley, 1990.

[9] Neider, J., Davis, T., and Woo, M. Open GL Programming Guide: The Official Guide to Learning OpenGL, release 1. Addison-Wesley, 1993.

[10] Mine."Virtual Environment Interaction Technique"

[11] Herman: "Virtual reality a new technology: A new tool for personal selection". Journal of neurosurgery, 2002.

[12] A. Steed. Towards a general model for selection in virtual environments. In 3DUI '06: Proceedings of the 3D User Interfaces, pages103–110, 2006.

[13] A. Ulinski, C. Zanbaka, Z. Wartell, P. Goolkasian, and L. Hodges.Two handed selection techniques for volumetric data. In 3DUI '07:Proceedings of the 3D