ISSN: 2277-3754
International Journal of Engineering and Innovative Technology (IJEIT)
Volume 1, Issue 2, February 2012

Analysis of Object Picking Algorithms Using Bounding Box in Non Immersive Virtual World

K.Merriliance, Dr. M.Mohamed Sathik
merriliance@gmail.com, mmdsadiq@gmail.com

Abstract*— Most computer applications depend heavily on user input. Within many games and virtual environments, for example, user input is essential to create and/or direct actions within a virtual environment. Much of this input comes through direct interaction with the virtual world's content, usually using a mouse. Most applications provide low level programs for 3D object selection, which makes the process of building interactive virtual applications a challenge for programmers. This paper describes a high level 3D object picking using bounding box method from non immersive virtual world which greatly simplifies that process. 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. 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. Object picking is a interaction technique which must be supported by any interactive three-dimensional virtual reality application. We presented a limited understanding on how these important factors will affect picking 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.

***Index Terms*—3D Object Selection, Non immersive Virtual Environment, Pick ray, Bounding box.**

I. INTRODUCTION

Virtual reality is the presentation of a 3D scene of sufficient quality to evoke a perceptual response similar to a real scene. The 3D scene is often called a virtual world, and the term "virtual reality" is commonly abbreviated "VR". Non-immersive virtual environment that allows users to view and interact with stereoscopic objects displayed on a workspace similar to a tabletop workspace used in day-to-day life. 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 behavior driven from the real world, for example by a user [1]. Alternatively, object behavior may be procedurally defined in a computer program.

In Non-Immersive VR Systems, the user will navigate through the environment using a joystick or space ball. The workstation will be operating like an interactive television. Object Picking is performed in VR world using a 3D mouse or an interactive glove to explore and interact with any of the objects. The user opens a door by moving the icon of the 3D mouse towards the position of the door's handle. When a collision is detected, the user confirms the selection of the door by activating a button on the mouse [2]. In virtual environments, Object picking is typically performed by bounding box checks or collision detection, taking the position of a virtual hand and the objects in account. Since 3D Picking is a costly task, especially when performed on a precise 3D CAD model, hence an algorithm is needed that accelerates Picking. In the field of 3D CAD it is important that the pick generates precise information using the accurate CAD model and supports the identification of topological entities such as faces, edges and vertices. Thus, a 'real' 3D pick requires the 'is-inside-model check'. To reduce these costly operations on the precise CAD model, we introduce an algorithm that performs multi-level bounding box checks and uses coherence during user interactions. This 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 it is essential to incorporate adequate feedback [2]. Object Picking is a method in which a 3D mouse or an interactive glove can be used to explore and interact with any of the objects that have been assigned dynamic properties. 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.

The user must know when he has chosen an object for selection, must know when he has successfully performed a selection action and must be able to determine the current selection state of all objects. The selection of small or distant objects can be facilitated by several enhancements. Figure 1 shows an example of how to interact an object model

1) The user indicates a target object and provides the system with a task instruction on how to pick it.

2) The system acquires geometric information about the target object and constructs a bounding box around the target by stereo vision using the information obtained from the task instruction.

3) The system finds a grasp point based on grasp evaluation using the acquired information movement.

ISSN: 2277-3754
International Journal of Engineering and Innovative Technology (IJEIT)
Volume 1, Issue 2, February 2012

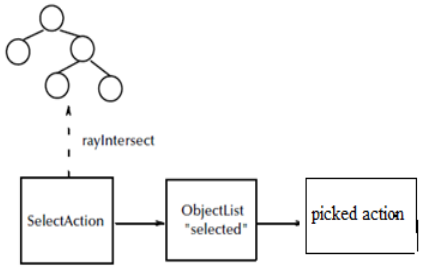

Fig. 1. Communication between objects

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. In some cases the interaction is not directly with the selected object, but with an object along the scene graph path to the object [3]. All objects have the same basic structure. Each object defined in the input file has a name and may be followed by a list of properties and property values. Property values that are not explicitly defined are set to a default value. The container methods invoke the methods of all the children nodes. A 3D selection interface object, controlled by the tracked wand, allows selecting, moving and touching objects in the VE. The 3D selection object consists of a selection bounding box and two perpendicular vectors, indicating the current spatial orientation of the wand .The 3D selection object can be used in absolute or incremental mode. In the absolute mode, the physical absolute position of the wand is set to the position of the selection tool bounding box in the VE. Once the selection bounding box touches an object, a touch event for that object is triggered and the associated sub-routine is executed [4].

A.*Picking Objects from Virtual World:* Selection techniques in 3D environments revealed that the environment density and visibility of the goal target are factors which are not well understood. As such, it is our goal to design and evaluate techniques which can adequately account for these two variables [5]. In addition to some standard design guidelines for 3D selection techniques and Picking design should

- Allow for accurate selections.
- Be easy to understand and use.
- Reducing their "empty space" so their collisions become more representative of the collisions of the contained objects.
- Produce low levels of fatigue.
- Allow fast selections
- Satisfy the above for sparse and dense target environments.
- Support selections for both visible and occluded targets.
- Our strategy for reducing the effects of these constraints will be to increase the activation areas of the selection techniques.

Use Object picking to identify the objects on the screen that appear near the cursor. To use Object picking, the software must be structured so that the picture can be regenerated on the screen whenever picking is required. As there was nothing special about the notation used for 2D shape picking, the same approach can be used for 3D object picking. Bounding Volume is a 3D object that encloses an object [5]. Different types of bounding volumes may be considered, each of them having their own strong points and weaknesses. There are two types of Bounding Volumes. They are

1. Bounding Boxes (Fig. 2 (a)).
2. Bounding Spheres. (Fig. 2 (b)).

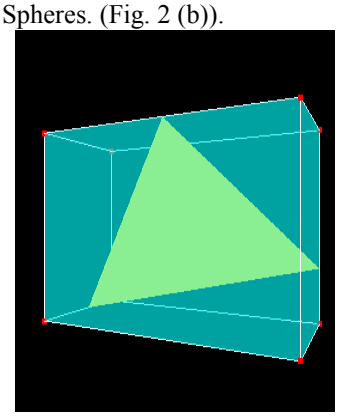

Fig .2(a). A bounding box dragger

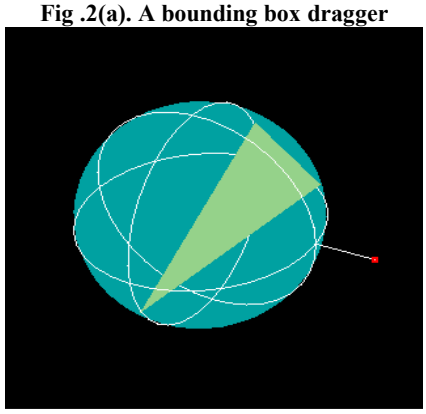

Fig 2(b): A bounding sphere

The bounding boxes are usually axis-oriented, described by two opposite corner vertices, and the bounding spheres are described by the center and the radius. A Bounding Box for an object is just a rectangular box in three dimensional spaces, with sides parallel to the coordinate planes, that contains the object. A bounding box is an orthogonal, rectangular volume that bounds an object. More complicated bounding volumes may be considered for efficient bounding when a small number of bounding primitives are required [6]. Such volumes use more parameters in their description, allowing a wider range of shapes in optimizing their filling efficiency and trading away some of their computational simplicity. The choice is highly dependent of the shape of the objects to be bounded. For elongated objects, possible solutions include bounding ellipsoids and cylinders. Clicking a mouse will create an appropriate picking bound at a 3D coordinate associated with the current mouse position. Object within a bound is selected. When no bounding box intersects with the picking ray, no object is selected. Using a mouse to select objects in 3D is a little tricky because the mouse gives only 2D pixel coordinates which must be somehow converted

ISSN: 2277-3754
International Journal of Engineering and Innovative Technology (IJEIT)
Volume 1, Issue 2, February 2012

to 3D coordinates. 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 3D environment, there may be more than one object under the mouse pointer when it is clicked [6].

Normally, the user's intention is to select 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 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 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 selected. When no bounding box intersects with the picking ray, no object is selected [6]. We will be making it so that you can "pick up" and move objects after you have placed them. We would like to have a way for the user to know which object it's 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. The 3D input devices adopted by these systems allow for direct 3D interaction, thus to completely support 3D interaction.

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. As an example for a heuristic approach we list the idea of using a library of predefined objects with predefined movement behaviors. 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. Hand movements not decreasing that angle drag the selected object as in single-user manipulation [7].

The ability to navigate through a world seen only on your computer screen, or through a special headset or visor, 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, 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 [6]. 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 [7]. 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. It will also set the clicked on object to be active in the scene so we can access it and work with other things once we know what was clicked.

II. SELECTION

When a tip of a 3-D cursor is positioned in a block, it is selected. To make the selection clear, the stabbed primitive is highlighted and the bounding box of the whole block appears. Selection is the process of identifying one object, usually to manipulate them in some way. It may also be used to identify a region of space and/or the objects inside. Here the user must select the object from a distance using a tool of some sort. The most intuitive and easy way for the user to select an object is to simply reach out until his hand intersects the desired object [6]. After some trial, we decided that the most precise way to support this was to put a tracker on the forefinger of each Pinch Glove and define a small, invisible box around it. An object is selected when this box intersects the object's bounding box. This technique is always active, even when one of the other tools described next inactive as well. This component provides two basic functionalities

- The maintenance of selection state
- The identification mechanism.

The selection is a binary state associated with all elements in the toolkit data structure. The meaning of a selected state is given by the application. Typically it indicates which selected elements can be manipulated. The identification occurs when a user "clicks" a pointing device over the output of a camera or mice. This process first detects whether that action occurred on top of some graphic primitive [7]. If that happened, then the element that contains such primitive is located. Finally, the state of this element is changed accordingly to its previous state. We identify object picking using pick ray, which are shown in Figure 3.

B. 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.
- Ray is the path of photons that successfully reach the eye.
- Ray is a sampling probe that gathers color/visibility information

ISSN: 2277-3754
International Journal of Engineering and Innovative Technology (IJEIT)
Volume 1, Issue 2, February 2012

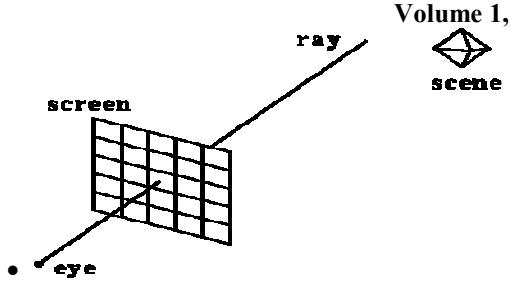

Fig.3. object picking using pick ray

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 can activate a ray that shoots out from the tip of his index finger [8]. To select an object the user has to intersect it with the ray. This method is often used in VE applications to select distant objects. If multiple objects are intersected, the first object is chosen. Ray can be expressed in the parametric form

E + t (P-E) ---------- (1)

Where E is the eye point and P is the pixel point. This technique allows the user to select an object by pointing at it with his forefinger so that its tip overlaps the object in the user's field of view. To determine where the user is pointing, an invisible ray shoots out from between the user's eyes to the fingertip. The object closest to the user's eyes is chosen. They can serve as a reference for the cameras, or can be used to improve the depth perception of the scene. Additionally, guides can provide a visual feedback for constraints applied to draggers [8].

C. Advantages of This Method:

- It is fairly fast compared to all picking methods.
- The selection primitives are no longer bound to rectangular shapes, even textures and alpha channels can be used as selection data.
- It also works flawlessly with all transformations, fragment tests and even in 3D.

III. BOUNDING BOX CHECKING

With rectangular objects, it can be difficult to tell the difference between the object's bounding box and the path of the object itself. A bounding box always displays eight large hollow anchor points. A rectangular path always displays four small anchor points. For any object, you can select its bounding box—a rectangle that represents the object's horizontal and vertical dimensions [9]. The bounding box is also called a container. The bounding box makes it possible to quickly move, duplicate, and scale the object without having to use any other tool. For paths, the bounding box makes it easy to work with an entire object without accidentally altering the anchor points that determine its shape.

When you select one or more objects with the Selection tool, you see a bounding box that indicates the size of each object. If you don't see a bounding box when an object is selected, you may have selected the object using the direct method we will learn here is something called "Bounding Volumes". A bounding volume some sort of volume, usually a cube or sphere, which tightly surrounds a mesh. When we check for collisions or picking, we first check the bounding volumes of the objects, and if the bounding volume has been intersected, then we can move on to check the actual object the bounding volume is bound to. Scenes becoming more complex usually means more triangles to test against when checking for collision or picking [9]. So, we can learn here how to minimize the amount of triangles that need to be tested each frame. Figure 4 illustrates this picking phase using bounding box. To compute a bounding sphere is not that difficult, in fact, all it really is a radius with a position, usually the center of the object.

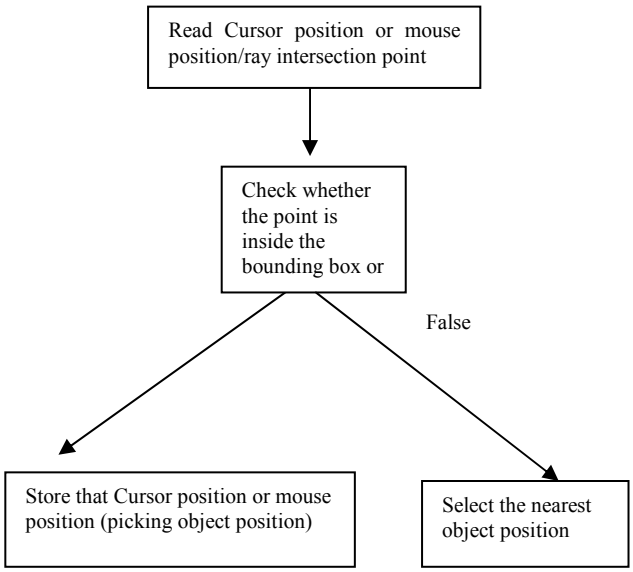

Fig .4. General model for picking a 3D object using bounding box

To find it, we would iterate through all the vertices in the object, and set the radius to the distance of the vertex furthest from the center of the object. To find the bounding box, we will need to create two points, the minimum and the maximum, which we will use to find the size of our bounding box, to make sure it tightly surrounds our object. To find the minimum and maximum points, we will iterate through each vertex in the object. We will find the maximum and minimum +X, -X, +Y, -Y, +Z, and -Z points, and set the distance to the most positive x y and z points to the maximum point, and the negative x y and z points to the minimum point. To make a bounding box, we have four points: P1(-1,0),P2(+1,0),P3(0,-1),P4(0,+1).We will iterate through each point, and set the maximum and minimum points to the maximum and minimum distances, to make sure we surround all points[10]. First we test P1(-1, 0) - Since this is the first one, it will set the Min point to (-1,0) and the Max point to (-1, 0)Next is P2(+1, 0) - Since The X value of Min is smaller, Min will stay (-1, 0), but since the X value of Max is smaller, it will set Max to (+1, 0)Next, P3(0, -1) - The X value of the min and max are both higher and lower, so it will not set the X value of either, but will set the Y value of Min to -1, making

ISSN: 2277-3754
International Journal of Engineering and Innovative Technology (IJEIT)
Volume 1, Issue 2, February 2012

Min (-1, -1), and Max (+1, 0)Finally 4(0,+1) Min will stay the same, since both values of Min are already smaller than the values of P4, but Max will change to (+1, +1) So the final result is: Min(-1, -1) and Max(+1, +1). Figure 5 illustrates the visual representation of a bounding box. Now we need to use the Min and Max points to create a square that surrounds our points. We can create each vertex like this V1 (Min.x, Min.y), V2 (Min.x, Max.y), V3 (Max.x, Max.y), V4 (Max.x, Min.y)

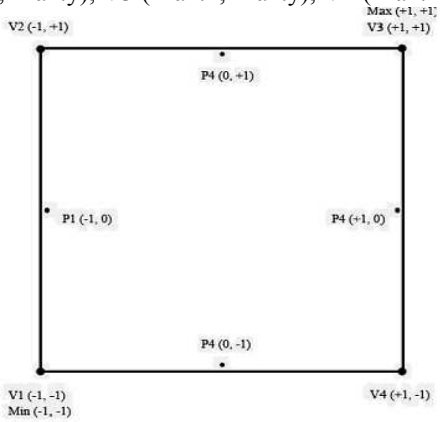

Fig .5. Visual Representation of a Bounding Box

A description of the algorithm is given below.

Select Object: When the user selects an object

Step1. Shoot a ray from p0 or any input device to intersect with the object to get the 3D position of the pixel at the intersection point p1.

Step2. Align the virtual object to the axes that the object movement has been constrained to the diagram.

Step 3.Move the height of the 3D object to the height of p1.

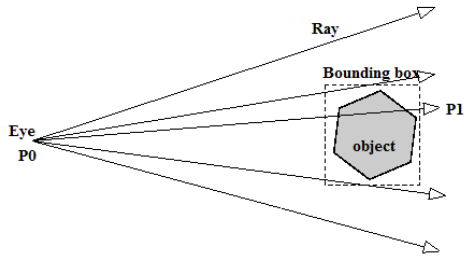

Fig .6.object picking using Bounding box

If an object has been picked, the position on the object that the ray hit the object is stored. The parameters passed to the function when it is called, are

- The current mouse position Mouse2DPosition->X
- The current Mouse2DPosition->Y
- The 3D position on the object that had just been picked

The 2D mouse position on the screen is passed to this function. The mouse ray in propagated [8]; the intersection point of the mouse ray with the virtual object is found; the intersection point is modified based on axis constraints.

IV. RAY INTERSECTION TEST

- Ray: defined by 2 points p0 and p1.
- An object is defined by a normal vector n and a point in the plane V0.

A vector can be thought of in two ways: either a point at <x, y, z> or a line going from the origin <0, 0, 0> to the point <x, y, z>.

w =P0-V0 ---------- (2)

//distance between the ray and the intersected object

v = V1-V0//width of the object ---------- (3)

u = P1-P0 ---------- (4)

w + su = v ----------(5)

The following expression is used as the final equation to solve:P0 + su = V1---------- (6)

- This can be thought of as 'move the start point P0 by the vector su to get to the end point V1, where s is described as some scalar value. Equation(6) is expressed as the intersection Point For the code version of this algorithm presented below:

Input: p = an object

Output: I = the intersection point

If i= 0 (no intersection occurs) or i= 1(intersection in the unique point I) else the segment lies in the object

```
int Intersects(object^ p, Vector^ I)
    {
      Vector^ intersect Point;
      Vector^ u = p1 - p0;
       Vector^ w = p0 - p->Position;
        Float    D = p->Normal->Dot (u);
        Float    N = -p->Normal->Dot (w);
       If (Math: Abs (D) < ZERO)     // segment is parallel to
the object
            {
                 if (N == 0)       // segment lies in the object
                   return 2;
                 else
                      return 0;       // no intersection
            }
            float s = N / D;
            if (s < 0 OR s > 1)
               return 0;              // no intersection
 Intersect Point = p0 + u*s;    // compute segment intersection
point
            I.X = intersectPoint->X;
            I.Y = intersectPoint->Y;
            I.Z = intersectPoint->Z;
            Return 1;
            }
```

It is often the case in 3D environments that users need to select objects which are obscured from their viewpoint. This is generally the case when the target of interest lies behind another object in the scene and is thus occluded. If cursor is inside the enlarged face bounding box store distance and current position. If the environment is densely populated with targets, the chance of such an occlusion occurring increases [10]. With a target being invisible to the user, it is generally impossible to select, as the user will have no visual feedback as to the location of the cursor relative to the intended target.

ISSN: 2277-3754
International Journal of Engineering and Innovative Technology (IJEIT)
Volume 1, Issue 2, February 2012

A. ***Advantages:***

- Ray tracing is slow.
- Ray intersect object is often expensive.
- Improving speed in two ways.
- Reduce the cost of ray intersect object.
- Cost of performing intersection test will be totally application-independent and very low.
- Selection Supports both visible and occluded target.
- Accurate fine positioning in 3D space.
- It is possible to minimize the costly nearest point calculations.
- Results can be easy reused if these have already acquired at once.
- No additional information on the object and environment are needed.

Construction of good bounding box can be difficult. One way to determine which object a user has picked, be it via mouse click or finger tap, is to generate a pick ray relative to the user's eye, and transform it by the inverse of the matrix. 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 [11]. To show that the current object has been deselected by removing the bounding box visual

```
if (Current Object)
{
   Current Object->showBoundingBox (false);
Else
Current Object->showBoundingBox (true); // show the bounding box so the user can see that which object is selected
}
```

Now the Current Object is always highlighted on the screen

V. OBJECT PICKING USING SPATIAL SUBDIVISION

A. ALGORITHM

Step 1. Divide space in to subregions

Step2.Place objects from scene in to appropriate subregions

Step 3.When tracing ray, only intersect with objects in sub-regions through which the ray passes

Step 4. Select an object by positioning mouse over it and clicking.

Step 5.When hit occurs, copy entire contents of stack to output buffer.

The environment consisted of Object picking using Spatial Subdivision as shown in Figure 7.

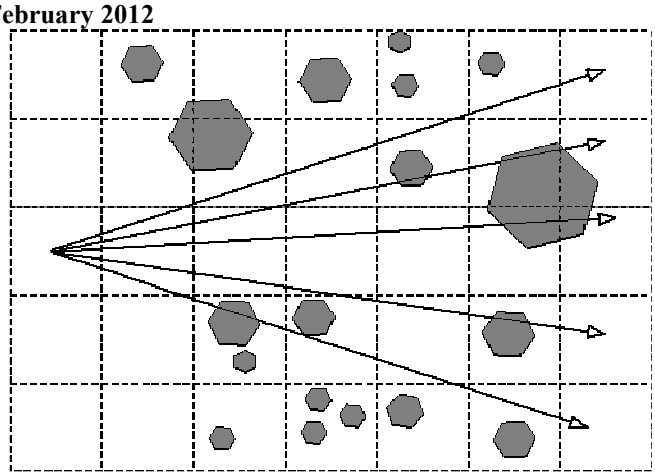

Fig. 7.Object picking using Spatial Subdivision

B. **Advantages:**

- useful when lots of small objects in scene
- Map selection point to a ray is simple.
- Intersect with all objects in scene.
- very easy access to 3D object selection

This will allow us to access the outside of the scene and manipulate some of our camera variables to change whenever a new is clicked on [11]. This algorithm takes as input the matched key points of the observed object. It first calculates a trimmed mean over the motion values, considering the upper and lower bounds as false matching. The last step consists in positioning the bounding box on the observed object in order to have the gravity centre at the same position as for the model [12]. Since the bounding box positioning is done in relation to the previous object, errors will propagate during the sequence. In order to avoid most of these disadvantages, we have chosen a bounding box using spatial subdivision algorithm. Knowing the position (x,y) of the key points for the object model and the observed object we use the this method to compute the position of the object correctly.

Table 1.Comparative study of various PICKING ALGORITHMS

Main features	**picking using Bounding box**	**picking Algorithm using spatial subdivision**
Cost of performing intersection test	High(expensive)	Low(Determining subdivision is difficult)
Ray Tracing	slow	fast
Selection	Visible only	All s in the scene
speed	high	low

ISSN: 2277-3754
International Journal of Engineering and Innovative Technology (IJEIT)
Volume 1, Issue 2, February 2012

Pointing	Hard	Accurate fine positioning in 3D space

Based on the results of a series of user's studies, we presented a list of guidelines for techniques to pick objects in 3D scenes. Depending on the application, it can be advantageous to provide visual feedback to the user on the state of the semantic pointing functionality. When we analysis these two algorithms in the view of selection and cost of performing an intersection test , picking using bounding box algorithm is always greater than the average cost along the picking algorithm using spatial subdivision Figure 8 shows the estimation of the intersection test.

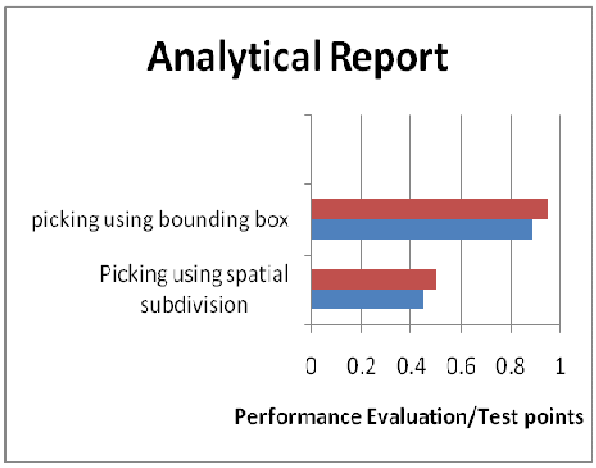

Fig .8. Average cost of performing an intersection test

VI. CONCLUSION

Although this has not been an exhaustive depiction of all possible interaction techniques, it provides a good exposure to many of the more common techniques currently in use. It is important to examine the interaction techniques that are available and determine those that are best suited for the tasks that need to be accomplished. The method I described in this paper is better one to pick an object from the virtual world. The investigation resulted in new design guidelines that will allow for more usable design of non-immersive desktop, photo-realistic virtual environments. Furthermore, the study provides some new areas for future developments of usability evaluation methods for non-immersive virtual world. Picking is a primary interaction technique which must be supported by any interactive three-dimensional virtual reality application. Thus, a limited understanding on how these important factors will affect selection performance. In this paper, we present a set of design guidelines and strategies to aid the development of selection techniques using bounding box. We discussed an implementation of the proposed algorithms. Based on these guidelines, we present new forms of the picking using bounding box and 3D picking Algorithm techniques, which are augmented with positioning, selection and average cost of performing intersection test feedback, to support selection within dense and occluded 3D target environments. Our analysis indicated that our introduced visual feedback played the most critical role in aiding the selection task. We have presented an efficient algorithm for intersection tests between a picking ray and multiple s in non immersive environment. Furthermore, the Analytical Report provides some new areas for future developments of usability evaluation methods. We have compared the performance of the two methods, with retrieval results. Furthermore, the results showed that our Handle the hits, and get the picked new techniques adequately allowed users to select targets which were not visible from their initial viewpoint. This paper has given a good starting point for designers and can be directly applied to all objects. Thus the performance of the picking process increase while using the bounding box. Thus this algorithm is too simple, more efficient and it helps in easy interaction with the virtual world, hence it makes the Virtual world user friendly.

REFERENCES

[1] AMO-VILLANI N., WRIGHT K.: SMILE: "Effects of platform (immersive versus non-immersive) on usability and enjoyment of a virtual learning environment for deaf and hearing children". ACM Proceedings of SIGGRAPH 2007.

[2] Ivan Poupyrev Interaction Lab, Sony CSL: Beyond VR: "3D interfaces in Non immersive Environment". (2001).

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

[4] George C.Robertson: "Non Immersive Virtual reality".(2005)

[5] "Interaction for VR". Year of Publication: 2003. ISBN: 1-58113-578-5.

[6] ISSAC"A META CAD system for virtual Environment" Computer-Aided Design, Volume 29, Issue 8, August 1997, Pages 547-553 Mark R Mine.

[7] "The simple virtual Environment library User's Guide"-www.cc.getech.edu.

[8] Elsivier."Special issue on Virtual Environment interaction". Journal of Visual Languages & Computing · Volume 10, Issue 1, February 1999.

[9] M.Mine."Virtual Environment Interaction Technique" (1995).

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

[11] 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.

[12] Jesper Kjeldskov."Combining Interaction technique and Display types of Virtual Reality". J Kjeldskov - Proceedings of OzCHI, 2001 - cs.aau.dk.

[13] Cleber S.Ughini, Fausto R.Blahco "3D interaction Technique for accurate object selection in Immersive Environment". Proceedings of the 1997 symposium on Interactive 3D.

[14] Y. Makihara, M. Takizawa, K. Ninokata, Y. Shirai, J. Miura and N.Shimada, "A Service Robot Acting by occasional Dialog – Object Recognition Using Dialog with User and

ISSN: 2277-3754
International Journal of Engineering and Innovative Technology (IJEIT)
Volume 1, Issue 2, February 2012

Sensor-Based Manipulation-," Journal of Robotics and Mechatronics, Vol. 14, No. 2, pp. 124–132,2002.

AUTHOR BIOGRAPHY

K.Merriliance received her MSc degree in Computer science from Madurai Kamaraj University, Madurai, and M.Phil. degree in Computer Science from Mother Teresa Women's University, Kodaikanal and pursuing part-time Ph.D in computer science in Mother Teresa Women's University, Kodaikanal. Currently she is working as a Lecturer in the Department of MCA, Sarah Tucker College, Tirunelveli Under the guidance of Dr.M.Mohamed Sathik, she has presented and published many papers in National and International journals. Her areas of specialization are Image Processing and Virtual Reality.

Dr.M.Mohamed Sathik received his Ph.D degree, M.Tech. and M.Phil., degree in Computer Science from Manonmaniam Sundaranar University, Tirunelveli, India ,Master Degree in Business Administration from Alagappa University, Karaikudi, India and M.S in Counseling And Psycho Therapy from Institute Of Psychotherapy and Management Mumbai, India. Currently he is working as Principal and Associate Professor in the Department of Computer Science, Sadakathullah Appa College, Tirunelveli.. His experience started from 1987. He has guided more than 30 research scholars and also published more books. He is a member of curriculum development committee of various universities and autonomous colleges of Tamilnadu and his area of specialization are Virtual Reality, Image Processing and Sensor Networks.