**Closest Points**

By John Ying

**Introduction**

Through various different types of video games in the world, many include the need for predicting the best path to a location. In many cases, this most commonly refers to AI (artificial intelligence). If a computer character needs to find the quickest route to his designated position (whether that be a turret gun, safe house, alarm, etc.), the closest points program will help in predicting the quickest route to the said path.

**Methods**

Many, many, many methods were used in the process of calculating the closest point to an object based upon a line aka a possible trajectory or a plane of an object.

The first to be talked about is the calculation used to figure out the distance between points. Although this piece had to be created anew, the .getMagnitude() function created in a previous section was the key to figure out the distance.

x = cPointHolder1.getX() - ship.getX();

distance1.setRectGivenRect(x, y, z);

distance1.getMagnitude()

Using the projection() function from a previous lab as well, the finding of the projection of vectors upon one another was key. Upon the top half of the division, the coordinates of the U and V vectors are multiplied by corresponding coordinates and then added together based upon the separating directions. On the bottom, it was the addition of the different v Vector axis AFTER the squaring of each one. Finally, it solved the final projection by finding the division of all of that multiplied by the v Vector coordinates

topAdd = (u.x \* v.x) + (u.y \* v.y) + (u.z \* v.z);

botAdd = (v.x \* v.x) + (v.y \* v.y) + (v.z \* v.z);

ans = topAdd / botAdd;

proj.x = ans \* v.x;

proj.y = ans \* v.y;

proj.z = ans \* v.z;

The cross product was a simple calculation of a given mathematical equation in basic vector projections. This equation was turned into code form.

x = (v2.y \* v3.z) - (v2.z \* v3.y);

y = (v2.z \* v3.x) - (v2.x \* v3.z);

z = (v2.x \* v3.y) - (v2.y \* v3.x);

Below solves for PQ in the standard projection equation by taking Q subtracted by P which is ship minus point.

pq.x = ship.x - point.x;

pq.y = ship.y - point.y;

pq.z = ship.z - point.z;

Next is the projection of d vector onto PQ. Using the predetermined function projection with the parameters v and u.

proj = projection(direction, pq);

Finally, for the closestPointLine function, the solution of ‘S’ is P – projection of d onto PQ.

x = point.x + proj.x;

y = point.y + proj.y;

z = point.z + proj.z;

In finding the closest point based upon a point and plane, two vectors need to be established by subtracting the position of any of the three given points.

v1.x = planeP1.x - planeP2.x;

v2.x = planeP1.x - planeP3.x;

Next is finding the cross product from the earlier explained function based upon the two vectors found previously.

normal.crossProduct (v1, v2);

In finding P, a random point that was given for the plane must be subtracted from Q, the ship.

point.x = ship.x - planeP1.x;

Now that we have point, we can find the projection based on the normal found from the cross product and the point found from subtraction.

proj = projection(normal, point);

Finally, the function solves for ‘S’ by subtracting the projection found previously from Q aka the ship.

x = ship.x - proj.x;

**Results**

Finally, the results are presented based on user defined input so that any values that the user wishes to input may be calculated.

----------------------------------------------------------------

*Enter the components of the ship in space: {x y z}*

*282 791 456*

*Enter the point of a moving object: {x y z}*

*151 366 965*

*Enter the direction of the moving object: {x y z}*

*12 31 -46*

*The closest point to the ship is:*

*x: 293.171*

*y: 733.274*

*z: 420.012*

*The distance from that ship is: 68.9359km.*

*Enter 3 points on a surface: {x y z}*

*377 912 598*

*Enter the second point: {x y z}*

*385 920 593*

*Enter the third point: {x y z}*

*388 914 604*

*The closest point on the surface to the ship is:*

*x: 229.995*

*y: 883.354*

*z: 520.558*

*The distance from that ship is: 124.103*

*Press any key to continue . . .*

----------------------------------------------------------------

**Conclusion**

I am proud say that after many, many hours, this program functions flawlessly for the desired need of the situation. There were many problems to start with but with adding more hours, some requested help and understanding, the closest points works well.