FUNCTIONS

Function [a] = GetMousePosition(x, y)

Receives the coordinates of the mouse in the plane and returns a vector(3) with the coordinates of the mouse on the sphere surface using Holroyd's arcball fuse.

Function [q] = GetQuaternionFromVectors(vec1, vec2)

Receives 2 vectors(3) that correspond to where the mouse was clicked and where the mouse is currently at in terms of the sphere and returns a quaternion using the last function shown in the following link.

http://lolengine.net/blog/2013/09/18/beautiful-maths-quaternion-from-vectors

Function [quaternion] = quaternionproduct(q, p)

Receives 2 quaternions and executes a product between them. Then it returns the resultant quaternion.

Function setGlogal( ) and Function getGlobal

Some of these functions have been created with the purpose of being able to share certain variables between other main functions.

Function [theta, phi, psi] = rotM2eAngles(mrotated)

Receives a rotation matrix and returns its corresponding Euler Angles.

Function [axis, angle] = rot2Mat2Eaa(mrotated)

Receives a rotation matrix and returns Euler Principal Angle and Axis.

Function [rmatrix] = eAngles2rotM(theta, phi, psi)

Receives 3 Euler Angles and returns their corresponding rotation matrix.

Function [m] = Eaa2rotMat(u, angle)

Receives Euler Principal Angle and Axis and returns its corresponding rotation matrix.

Function [MatrixR] = matrixfromquaternion(quaternion)

Receives a quaternion and returns its corresponding rotation matrix.

Function [quatm] = quatfrommat(matrix)

Receives a rotation matrix and returns its corresponding quaternion.

Function [rotvec] = rotationvectorfromepa(u, angle)

Receives Euler Principle Angle and Axis and returns the corresponding rotation vector.

USER ACTIONS

Mouse Click and Drag

-Registers where the mouse was clicked with **GetMousePosition(x, y)**

Calculates the initial quaternion using **GetQuaternionFromVectors(vec1, vec2)** *in this case with vec 1 and vec 1*

-Registers where the mouse is currently at with **GetMousePosition(x, y)**

-Uses these 2 vectors (mouse positions) to get a delta quaternion with **GetQuaternionFromVectors(vec1, vec2)**

-Calculates the product of the initial quaternion and the delta quaternion with **quaternionproduct(q, p)**

-Calculates the rotation matrix with the resulting quaternion with **matrixfromquaternion(quaternion)**

-Uses the matrix to rotate the cube

Quaternion Push Button

-Gets the quaternion values that the user has set

-Uses it to get a rotation matrix with **matrixfromquaternion(quaternion)**

-Uses the matrix to rotate the cube

Euler Angles Push Button

-Gets the Euler angles that the user has set

-Uses them to get a rotation matrix with **eAngles2rotM(theta, phi, psi)**

-Uses the matrix to rotate the cube

Rotation Vector Push Button

-Gets the rotation vector that the user has set

-Gets the angle by calculating its module

-Uses **Eaa2rotMat(u, angle)** *to get a rotation matrix*

-Uses the matrix to rotate the cube

Euler Principal Angle and Axis Push Button

-Gets the data that the user has set

-Uses it to get a rotation matrix with **Eaa2rotMat(u, angle)**

-Uses the matrix to rotate the cube

Reset Push Button

-Sets every editable text to 0

-Redraws cube in starting position