Project MATVJII ARCBALL

Arnau Falgueras **Josep Sánchez** **Jordi Pardo**

Documentación

Lista de funciones:

- **Function** [rotmat] = **EulerAnglesToRotationMatrix**(roll,pitch,yaw)
  Pide como parámetros 3 ángulos y te devuelve una matriz de rotación.
  Y sirve para obtener la matriz de rotación a partir de 3 ángulos.

```
function [rotmat] = EulerAnglesToRotationMatrix(roll, pitch, yaw)
%EulerAxisToQuaternion Transform an Euler Angles to Rotation Matrix
Ryaw=[cos(yaw),sin(yaw),0;-sin(yaw),cos(yaw),0;0,0,1];
Rpitch=[cos(pitch),0,-sin(pitch);0,1,0;sin(pitch),0,cos(pitch)];
Rroll=[1,0,0;0,cos(roll),sin(roll);0,-sin(roll),cos(roll)];
rotmat=Ryaw'*Rpitch'*Rroll';
end
```

- **Function** [quat] = **EulerAxisToQuaternion**(axis,angle)
  Pide como parámetros el vector del Euler principal y un ángulo.
  Transforma el vector del Euler a Quaternion.

```
function [quat] = EulerAxisToQuaternion(axis,angle)
%EulerAxisToQuaternion Transform an Euler Axis to Quaternion
quat=[0;0;0;0];
quat(1)=cos(angle/2);
quat(2:4)=sin(angle/2)*axis;
end
```

- **Function** [rotmat] = **EulerAxisToRotationMatrix**(axis,angle)
  Pide como parámetros el vector del Euler principal y un ángulo.
  Transforma el vector del Euler a una matriz de rotación.

```
function [rotmat] = EulerAxisToRotationMatrix(axis,angle)
%EulerAxisToQuaternion Transform an Euler Axis to Rotation Matrix
ux=[0,-axis(3),axis(2);axis(3),0,-axis(1);-axis(2),axis(1),0];
rotmat1=eye(3)*cos(angle);
rotmat2=(1-cos(angle))*(axis*axis');
rotmat3=ux*sin(angle);
rotmat=rotmat1+rotmat2+rotmat3;
end
```

- **Function** [rotvec] = **EulerAxisToRotationVector**(axis,angle)
  Pide como parámetros el vector del Euler principal y un ángulo.
  Transforma el vector del Euler a un vector de rotación.

```
function [rotvec] = EulerAxisToRotationVector(axis,angle)
%EulerAxisToQuaternion Transform an Euler Axis to Rotation Vector
rotvec=axis*angle;
end
```

- **Function** [q] = **GetQuaternionFrom2Vectors**(vec1, vec2)
  Pide como parámetros 2 vectores y devuelve un Quaternion.
  A partir de dos vectores obtienes un Quaternion.

```
function [q] = GetQuaternionFrom2Vectors(vec1,vec2)
%GetQuaternionFrom2Vectors Get a Quaternion from two vectors
modulevec1=power(vec1,2);
modulevec1=sqrt(modulevec1(1)+modulevec1(2)+modulevec1(3));
modulevec2=power(vec2,2);
modulevec2=sqrt(modulevec2(1)+modulevec2(2)+modulevec2(3));
vec1=vec1/modulevec1;
vec2=vec2/modulevec2;
w = cross(vec1, vec2);
u=vec1'*vec2;
q = [(u); w(1); w(2); w(3)];
moduleq=power(q,2);
moduleq=sqrt(moduleq(1)+moduleq(2)+moduleq(3)+moduleq(4));
q=q/moduleq;
%normalize(q);
end
```

- **Function** [q] = **Multiply2Quaternions**(q1, q2)
  Pide como parámetros 2 Quaternion y te devuelve un Quaternion.
  Multiplica dos Quaternion para devolver su resultado.

```
function [q] = Multiply2Quaternions(q1,q2)
%Multiply2Quaternions Does the multiplication of two Quaternions
q=[0;0;0;0];
q(1)=(q1(1)*q2(1))-(q1(2:4)'*q2(2:4));
q(2:4)=(q1(1)*q2(2:4))+(q2(1)*q1(2:4))+(cross(q1(2:4),q2(2:4)));
end
```

- **Function** [axis,angle] = **QuaternionToEulerAxis**(quat)
  Pide como parámetro un Quaternion y devuelve un vector de Euler y un ángulo.
  A partir del Quaternion devuelve un vector Euler y un ángulo.

```
function [axis,angle] = QuaternionToEulerAxis(quat)
%QuaternionToEulerAxis Transform a Quaternion to Euler Axis
angle=acos(quat(1))*2;
if(angle==0)
    axis=[1;0;0];
else
    axis=quat(2:4)/sin(angle/2);
end
end
```

- **Function** [roll,pitch,yaw] = **RotationMatrixToEulerAngles**(rotmat)
  Pide como parámetro una matriz de rotación.
  Transforma la matriz de rotación en ángulos Euler.

```
function [roll, pitch, yaw] = RotationMatrixToEulerAngles(rotmat)
%EulerAxisToQuaternion Transform a Rotation Matrix to Euler Angles
pitch=asin(-rotmat(3,1));
if sin(pitch) == -1
    roll=0;
    yaw=atan2(rotmat(1,2)+sin(roll),rotmat(2,2)-cos(roll));
elseif sin(pitch) == 1
    roll=0;
    yaw=atan2(-(rotmat(1,2)-sin(roll)),-(rotmat(2,2)-cos(roll)));
else
    roll=atan2(rotmat(3,2)/cos(pitch),rotmat(3,3)/cos(pitch));
    yaw=atan2(rotmat(2,1)/cos(pitch),rotmat(1,1)/cos(pitch));
end
end
```

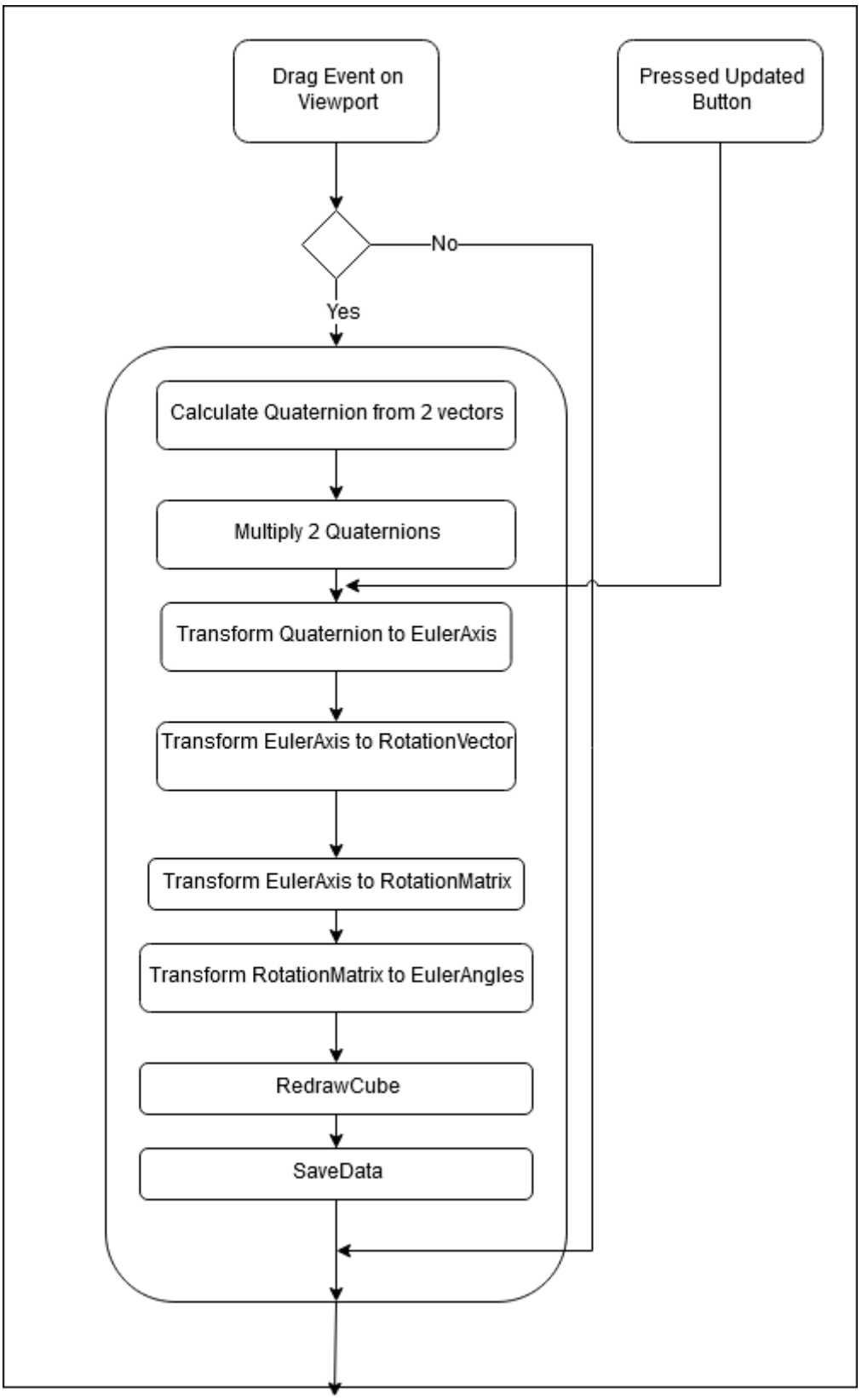
- **Function** [axis,angle] = **RotationMatrixToEulerAxis**(rotmat)
  Pide como parámetro una matriz de rotación y te devuelve un ángulo y el vector Euler.
  Transforma la matriz de rotación en un vector Euler.

```
function [axis,angle] = RotationMatrixToEulerAxis(rotmat)
%EulerAxisToQuaternion Transform a Rotation Matrix to Euler Axis
angle=acos((trace(rotmat)-1)/2);
if sin(angle) == 0
    if cos(angle) == 1
        axis=[1;0;0];
    else
        R=(rotmat+eye(3))/2;
        axis=[0;0;0];
        if axis(1)~=0
            axis(1)=sqrt(R(1,1));
            axis(2)=R(1,2)/axis(1);
            axis(3)=R(1,3)/axis(1);
        elseif axis(2)~=0
            axis(2)=sqrt(R(2,2));
            axis(1)=R(2,1)/axis(2);
            axis(3)=R(2,3)/axis(2);
        else
            axis(3)=sqrt(R(3,3));
            axis(1)=R(3,1)/axis(3);
            axis(2)=R(3,2)/axis(3);
        end
    end
else
    ux=(rotmat-rotmat')/(2*sin(angle));
    axis=[ux(3,2);ux(1,3);ux(2,1)];
end
end
```

- **Function** [axis,angle] = **RotationVectorToEulerAxis**(rotvec)
  Pide como parámetro un vector de rortación y te devuelve en vector Euler y un ángulo.
  Transforma un vector de rotación en un vector Euler.

```
function [axis,angle] = RotationVectorToEulerAxis(rotvec)
%EulerAxisToQuaternion Transform a Rotation Vector to Euler Axis
angle=norm(rotvec);
if(angle==0)
    axis=[1;0;0];
else
    axis=rotvec/angle;
end
end
```

Diagrama


Drag Event on Viewport
Pressed Updated Button
No
Yes
Calculate Quaternion from 2 vectors
Multiply 2 Quaternions
Transform Quaternion to EulerAxis
Transform EulerAxis to RotationVector
Transform EulerAxis to RotationMatrix
Transform RotationMatrix to EulerAngles
RedrawCube
SaveData