Project MATVJII:

Visualizing 3D data

Functions

- Juan Hernández Almagro
- Àlex Melenchón Maza
- Oscar Pérez Martín

SpaceCoordsToVec3

This function gets the 2D Mouse coordinates and the sphere radius and returns a vector in the 3D surface of a Holroyd's arcball

Inputs:

- (x,y,r): Mouse coordinates and sphere radius

Outputs:

- A: Vector from 3D surface of a Holroyd's arcball

QuatFrom2Vec

this function extracts and returns the quaternion rotation from two vectors in 3D. We will use this to get the quaternion from the mouse position and the previous one.

Inputs:

- (vec1, vec2): Vectors 3D

Outputs:

- q: resulting quaternion

quaternionMultiplication

Computes the quaternion multiplication of the given quaternions

Inputs:

- qb: quaternion on the left
- qa: quaternion on the right

Outputs:

- q: computed quaternion

ReCalculateParametrization

This function sets all the new information in the other parametrizations. Here, we will call the following functions to get – Quaternions – Euler principal Angle and Axis – Euler angles – Rotation vector – Rotation matrix – of the equivalent attitude representation.

Inputs:

- R: The new rotation matrix.
- alreadyComp: is a flag that we set to not recalculate already computed parametrization:
- handles: This allows us to change and set new info to our figure

rotMat2Quaternion

Computes a quaternion given a rotation matrix. In our project we will use this to set the new info of the quaternion.

Inputs:

- R: Rotation matrix

Outputs:

- q: generated quaternion

rotMat2Eeaa

Computes the angle and principal axis of rotation given a rotation matrix R. In our project we will use this to set the new info of the angle and principal axis.

Inputs:

- R: Rotation matrix

Outputs:

- a: angle of rotation in degrees
- u: axis of rotation

rotM2eAngles

Computes the Euler angles (yaw, pitch, roll) given an input rotation matrix R. In our project we will use this to set the new info of the Euler angles

Inputs:

- R: Rotation matrix

Outputs:

- yaw: angle of rotation around the z axis
- pitch: angle of rotation around the y axis
- roll: angle of rotation around the x axis

RotMat2rotVec

Computes a rotation vector given a rotation matrix. In our project we will use this to set the new info of the Rotation Vector.

Inputs:

- R: Rotation matrix

Outputs:

- v: generated rotation vector

Eaa2rotMat

Computes the rotation matrix R given an angle and axis of rotation.

Inputs:

- a: angle of rotation
- u: axis of rotation

Outputs:

- R: generated rotation matrix

eAngles2rotM

Computes the rotation matrix R given the Euler angles (yaw, pitch, roll)

Inputs:

- yaw: angle of rotation around the z axis
- pitch: angle of rotation around the y axis
- roll: angle of rotation around the x axis

Outputs:

- R: rotation matrix

quaternion2rotM

Computes the rotation matrix R given a quaternion.

Inputs:

- q: quaternion.

Outputs:

- R: Rotation Matrix.

RotVec2RotMat

Computes the rotation matrix R given a rotation vector.

Inputs:

- r: rotation vector.

Outputs:

- R: generated rotation matrix.

updatePrevRot

This is used so when the user pushes a new param. we get back to the origin, so we don't drag errors & so the rotation doesn't cause any critical condition w/ the previous established rotation &, therefore, the input from the user start not matching the viewport

Inputs:

- toReset: is a flag that tells if we have to reset the prevRot or not.

updatePrevQuat

This is used so when the user pushes a new param. we get back to the origin, so we don't drag errors & so the rotation doesn't cause any critical condition w/ the previous established rotation &, therefore, the input from the user start not matching the viewport

Inputs:

- toReset: is a flag that tells if we have to reset the prevQuat or not.