CIS 462/CIS562 Computer Animation (Fall 2016)
Homework Assignment 3
(Rotations)

Due: Mon, Oct. 10, 2016 (must be submitted to Canvas before midnight)

Please note the following:

- No late submissions or extensions for this homework.
- This is a programming assignment. You will need to use the *AnimationToolkit* code framework from the previous CIS462/562 CurveEditor assignment
- Unzip the HW3-AnimationToolkit.rar file and copy the files into the AnimationToolkit folder on your local machine.
- Double click on the file " Rotations-Demo.exe" in the AnimationToolkit\bin directory to see a demo of the Rotation assignment functionality
- Commit and push the updated project files to your CIS462-562 GitHub repository.
- When you are finished with this assignment, update your GitHub project files. Also, after doing a "Clean", zip your code under the *AnimationToolkit* directory along with a *Readme.txt* file and name it *hw02_name.zip,* then upload your solution to Canvas before the deadline.
- Work within the *AnimationToolkit* code framework provided. Feel free to enhance the GUI interface if you desire.
- You only need to implement the functions in the aRotation.cpp and aSplineQuat.cpp files marked with "TODO" to complete this assignment.
- **NOTE: THIS IS AN INDIVIDUAL, NOT A GROUP ASSIGNMENT. That means all code written by you for this assignment should be original! Although you are permitted to consult with each other while working on this assignment, code that is substantially the same as that submitted by another student will be considered cheating.**

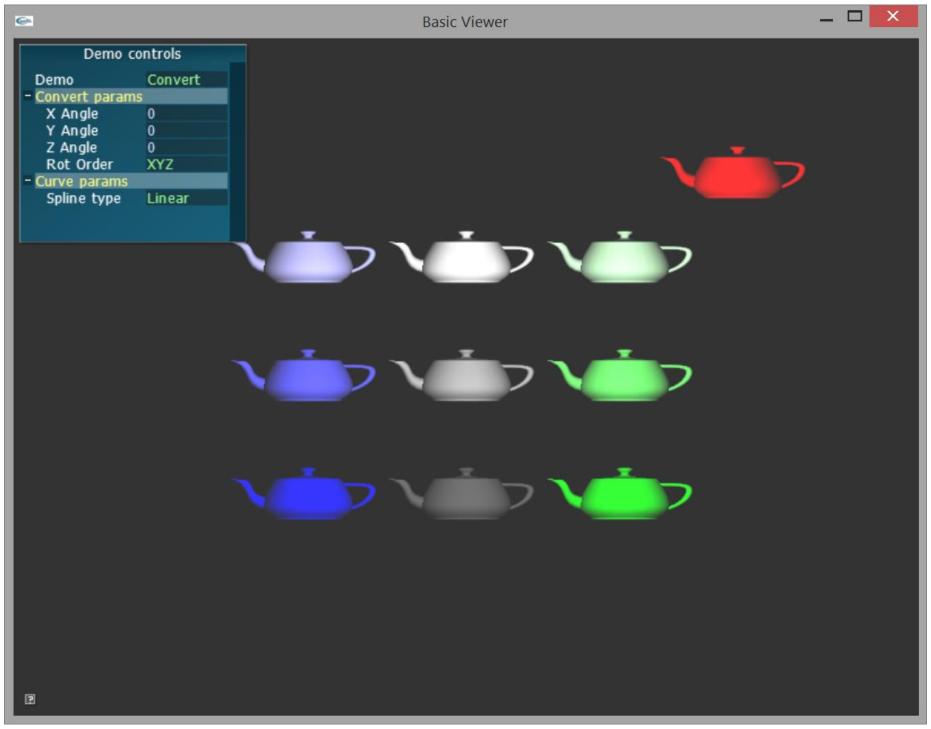
ANIMATION TOOLKIT – ORIENTATIONS

This is the first of a three-part assignment. In part 1of the assignment you will implement the following:

- conversions between the three different (but equivalent) orientation representations:
  - Euler Angles
  - Rotation matrix
  - Quaternions
- Interpolation of quaternions using linear and cubic splines

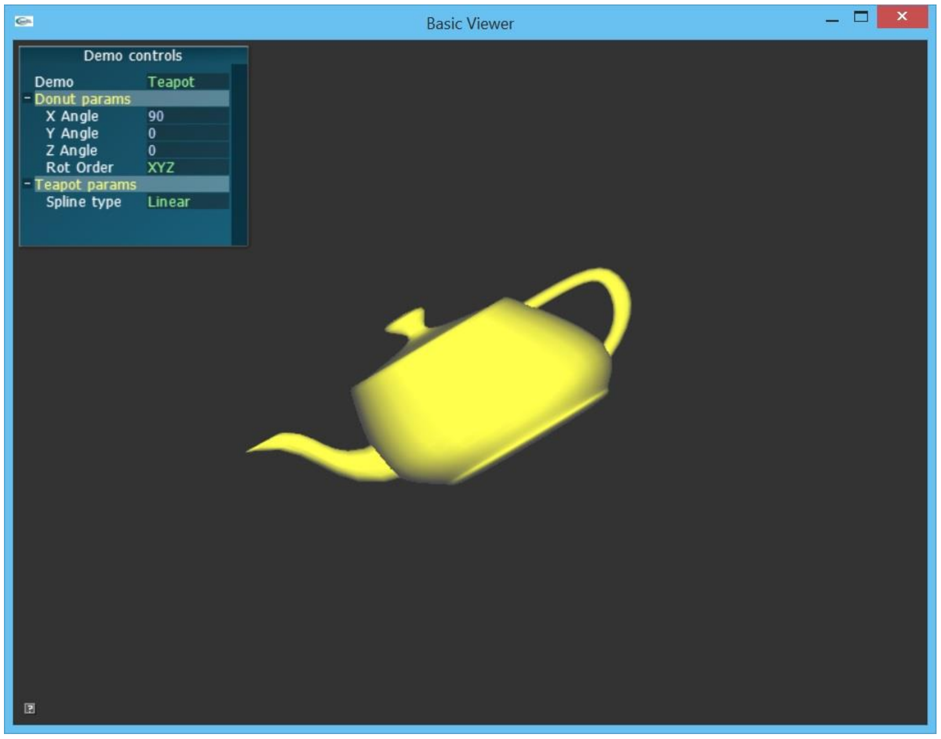
The assignment contains two modes to help you test your functions.

The **Convert Demo** mode computes the same orientation 9 different ways using Euler angles, rotation matrices, and quaternions. If all your orientation functions are implemented correctly, every teapot should match the red teapot in the top right-hand corner of the screen as shown in the screenshot below.



```
Demo controls
Demo            Convert
- Convert params
   X Angle      0
   Y Angle      0
   Z Angle      0
   Rot Order    XYZ
- Curve params
   Spline type  Linear
```

The **Curve Demo** mode uses a rotational spline to animate the orientation of the yellow teapot shown in the screenshot below. In this assignment you will animate the yellow teapot using either spherical linear interpolation (Slerp) or spherical cubic interpolation (Scubic).



```
Demo controls
Demo            Teapot
- Donut params
   X Angle      90
   Y Angle      0
   Z Angle      0
   Rot Order    XYZ
- Teapot params
   Spline type  Linear
```

In the RotationsViewer.exe application the Demo Controls panel lets you select the demo mode and set properties for either Curve and Convert demo modes.

About the AnimationToolkit Basecode

This assignment uses the same AnimationToolkit structure as the curve editor, but adds a new library and application for assignment 2. If you did not complete assignment 1, you may edit the basecode A2a-Rotations project file to use the libAssignment1-soln.lib solution library. To do this, right click on the A2a-Rotations project, select Properties, and then expand the Linker options. Select 'Input' and then edit the 'Additional Libraries' to point to libAssignment1-soln.lib instead of libAssignment1.lib.

Assignment Details

1. (50 points) **Orientation Representations.** In this part of the assignment you need to implement conversions between 6 different orientation representations by filling in the details of the functions listed below. The function `RotationViewer::drawDemo1()` has been created to help you debug your conversion routines.

   1. (10 points) Function to convert from Euler Angles to a 3x3 rotation matrix: `mat3::FromEulerAngles()`
   2. (10 points) Function to convert from a 3x3 rotation matrix to Euler Angles: `mat3::ToEulerAngles()`
   3. (10 points) Function to convert from a 3x3 rotation matrix to a quaternion: `quat::ToRotation()`
   4. (10 points) Function to convert from a quaternion to a 3x3 rotation matrix: `quat::FromRotation()`
   5. (5 points) Function to convert from a quaternion to an Axis/Angle representation: `quat::ToAxisAngle()`
   6. (5 points) Function to convert from Axis/Angle representation to a quaternion: `quat::FromAxisAngle()`

   Test your conversions by changing the X, Y, and Z values of the Euler angles. Also test each of the rotation orders.

2. (50 points) **Quaternion Splines.** The AnimationToolkit basecode includes the class ASplineQuat whose interface is similar to the vec3 splines you developed in the previous assignment. In this assignment, you need to complete the implementation of ASplineQuat by implementing the `Slerp`, `Scubic`, `computeControlPoints`, `getCubicValue`, `getLinearValue` functions. The function `RotationViewer::drawDemo2()` has been created to help you debug your conversion routines.

   - (20 points) **Linear quaternion interpolation.**
     - o `quat::Slerp()`
     - o `ASplineQuat::getLinearValue()`
   - (30 points) **Cubic quaternion interpolation.**
     - o `quat::Scubic()`
     - o `ASplineQuat::getCubicValue()`
     - o `ASplineQuat::computeControlPoints()`

Compare the animation of the teapot using linear versus cubic interpolation. What are the differences? (Note: The keys used for the demo are hard-coded in the RotationViewer constructor.)

3. (Extra credit, 15 points) **A little teapot, short and stout**

- (10 points) Animate the position of the teapot in the RotationViewer demo using a ASplineVec3
- (5 points) Animate the color of the teapot in the RotationViewer demo using a ASplineVec3
- (15 points) Create a keyframed animation sequence to make the teapot dance in the in the RotationViewer demo