Assignment 1—Hello 3D World

Due: Monday January 25 at 7am

1. Implement a 3-D vector class that stores 3 floats and has the following functionality:
	1. Constructor from 3 floats
	2. Read/write access through square brackets operator “[]”
	3. Addition of two vectors
	4. Subtraction of two vectors
	5. Dot product
	6. Cross product
	7. Multiplication by scalar
	8. Division by scalar
	9. Normalization
	10. Length computation
	11. Rotation of “this” point about arbitrary axis
	12. Rotation of “this” vector about arbitrary axis (same as above, except that the origin of the axis is the origin, i.e. (0, 0, 0), and not a parameter Oa)
	13. Output to ostream, input from istream
2. Implement a 3x3 matrix class that stores 3 3-D vectors as rows and has the following functionality:
	1. Constructor from 3 3-D vectors
	2. Read/write access to rows through square brackets operator “[]”
	3. Function to get column
	4. Function to set column
	5. Multiplication with 3-D vector
	6. Multiplication with another matrix
	7. Inversion (see code in Appendix 1 at the end of this document)
	8. Transposition
	9. Function to set matrix as rotation about principal axis by theta degrees
	10. Output to ostream, input from istream
3. Implement a frame buffer class that stores unsigned int pixels and the resolution of the frame buffer and that has the following functionality:
	1. Constructor from input resolution
	2. Set all pixels to given color
	3. Set one pixel to given color
	4. Drawing of axis aligned rectangle
	5. Drawing of 2D (i.e. projected) triangle with constant color
	6. Load / save from tiff file
4. Implement a planar pinhole camera class that stores the center of projection, the vector from the eye to the top left corner of the image, the pixel width vector, the pixel height vector, and the image resolution, and that has the following functionality:
	1. Constructor from resolution and horizontal field of view
	2. Left-right, up-down, and forward-backward translations
	3. Pan, tilt, roll rotations about center of projection
	4. Projection of 3-D point
	5. Save / load from text file
5. Demonstrate your code.
	1. Create a scene with a single cube that rotates about an arbitrary axis. The axis should not pass through the center of the cube and it should not be one of the axes of the (world) coordinate system.
	2. Render a 12s 30Hz 720p video sequence illustrating your rotating point. The video will have one frame for each position, i.e. 12x30=360 frames in total. For the first 6s, the camera should be stationary showing the cube rotating. For the second 6s, the camera should translate and rotate with all six degrees of freedom (i.e. three translations and three rotations). The video file should be in a popular format. Use the video making software of your choice.
6. Turn in via blackboard one zip archive that contains
	1. Source code
	2. Executable
	3. Video file

© Popescu 2016, popescu@purdue.edu

Appendix 1:

Sample matrix inversion code

The function returns the inverse of the matrix. ^ is the cross product operator between 3D vectors (i.e. V3 objects). \* is the dot product operator.

M33 M33::Inverted() {

 M33 ret;

 V3 a = GetColumn(0), b = GetColumn(1), c = GetColumn(2);

 V3 \_a = b ^ c; \_a = \_a / (a \* \_a);

 V3 \_b = c ^ a; \_b = \_b / (b \* \_b);

 V3 \_c = a ^ b; \_c = \_c / (c \* \_c);

 ret[0] = \_a; // set row 0

 ret[1] = \_b; // set row 1

 ret[2] = \_c; // set row 2

 return ret;

}