EECS 351-1: Introduction to Computer Graphics

Instructor: Jack Tumblin

Project B: Hinged Views

Winter 2012

Your mission in Project B is to move from 2D to 3D drawing by making multiple side-by-side, colorful, interactive 3D perspective and orthographic drawings of objects on a 'ground plane' that move on-screen. As with ProjectA, draw anything you like for your project, something *you* find interesting, meaningful and/or compelling. Use any and all inspiration sources. How about some clockwork gears? An interactive Rubik's cube, or even an NxN Rubik's cube (not just 3x3)? A steerable butterfly that flies around in 3D by flapping its wings? A fanciful ornithopter? A forest scene made of fractal/graftal/L-system trees and bushes? Could you make motorcycles, bicycles, tricycles, perhaps with a chain-drive that really works?

Requirements: **Project Demo Day (and due date): Thurs Feb 9, 2012**

1) In-Class Demo Day: on the Project's due date you will demonstrate your program to the class and to two other students who will fill out the already-posted Project B 'Grading Sheets' with numerical scores. The instructor(s) will also view your in-class demo, your project report, and the two student-filled grading sheets to determine your grade. **WARNING: Don't miss this demo day! If you do, you lose 'class participation' points!**

2) You must submit your work to Blackboard/CMS in the 48 hours *after* the demo day.
(allows you to make improvements in response to Demo Day comments and suggestions).
Submit just one single compressed (ZIP) file named your_netID_ProjB.zip.
For example, my filename would be **jet861_ProjB.zip**. The file contains:

a) your Project Report as a PDF file (same filename: e.g. **jet861_ProjB.pdf**), and

b) just one directory that holds all your source code and your CodeBlocks files (.cbp etc).

Do not include executables, object files, etc; keep it compact!

Project B consists of:

1)—Report: A short written, illustrated report, submitted as a printable PDF file, longer than 1 page and typically <5 pages, but you should decide how much is sufficient. A minimal report consists of 4 sections:

a)—your name, netID, title for your project.

(e.g. Project B: Toyota Prius Planetary Gear Transmission, not just "Project B")

b)—a brief description of your goals; more than 4 descriptive sentences, but a half-page is plenty. Explain WHAT your interactive program depicts, why is it interesting to you, and what your goal(s) you had for it (e.g. I've been curious to see just how the wheels turn in the strange planetary gear transmission of a Toyota Prius Hybrid).

c)—a brief 'User's Guide' that explains how to make your on-screen image move and change (e.g. A,a,F,f keys rotate outer ring forwards/backwards; S,s,D,d keys rotate inner ring forwards/backwards; arrow UP/DN keys speed up/slow down the electric motor, left/right keys act as the car's accelerator. Text below displays car's velocity in kilometers/hour.) **CAUTION:** Your classmates should be able easily understand and run your code from ONLY this report.

d)—a brief 'Code Guide' that explains your classes/data structures/files, and how they're accessed from GLUT callbacks. You can refer readers to well-commented code rather than re-explaining it, but explain enough for your work to be sensible and extensible by your classmates.

e)—a brief, illustrated 'Results' section that shows **at least 4 still pictures** of your program in action (use screen captures; no need for video capture), with text explanation of what the pictures are showing us, and how the project met your goals.

2)—User Instructions: When your program starts or when users press the F1 (help) key, print a brief set of user instructions, either in the console window or the graphical display window. (e.g. 'arrow keys move skateboard, left mouse drag steers, right mouse drag aims flame-thrower'.

3)---'Ground Plane' Grid: Your program must clearly depict a 'ground plane' that extends to the horizon -- a pattern of lines (or dots or any other openGL drawing primitive you like) fixed to the **x,y plane** of your 'world-space' coordinate system. This grid should make any and all camera movements obvious on-screen.

4)—Adjustable, Jointed 3D Shape: Your code must show **at least one jointed 3D object** with at least 4 sequentially-jointed segments connected by 3 independent joints. Let users adjust those joint angles smoothly by interactions with the mouse or keyboard.

(NOTE: You must create your own 3D shapes from vertex arrays, or for extra credit, from vertex buffer objects **(no glVertex() calls allowed)**. Parts made from GLUT-defined objects (teapots, torus, cube,etc) are OK, but you must make at least one vertex-array-defined object for each and every GLUT-defined object you use.

5)—Additional shapes: 4 or more separate, multi-colored 3D shapes 'Separate' means individually positioned, spatially separate, distinct, different-shaped objects; the top part of a robot and the bottom part make up just one object, as you wouldn't move one leftwards while you moved the other rightwards. 'Multi-color' means an object uses at least 3 different vertex colors; openGL will then blend between these vertex colors within the openGL drawing primitive.

(NOTE: Again, you must create your own 3D shapes from vertex arrays, or for extra credit, from vertex buffer objects **(no glVertex() calls allowed)**. Parts made from GLUT-defined objects (teapots, torus, cube,etc) are OK, but you must make at least one vertex-array-defined object for each and every GLUT-defined object you use.

I prohibit use of lighting, materials and texture maps in Project B, as we're saving them for projects C and D.

6)—Show 3D World Axes, 3D Model Axes: Draw one set fixed at the origin of 'world space' coordinates, such as those found in the starter code, and at least two others to show other coordinate systems within your jointed object.

7)--Multiple Viewports: Your program must depict the 3-D scene in 4 separate viewports in the window, arranged in a 2x2 grid that together fill the entire window without gaps and without distorting the images, even after window re-sizing makes the window taller or wider (e.g. 'anamorphic'combination of viewport and camera settings). Of the 4 viewports, 3 show fixed, orthographic views of front, top, side; the 4th shows an image with mouse view control

8)—View Control: smoothly & independently vary both 3D Camera position and its aiming direction (as in Proj B). Your code enable users to explore the 3D scene via user interaction. Try mouse dragging, arrow keys, menu selection, a 'virtual trackball', airplane controls (speed forward/back; roll, pitch, yaw) or something you invent for yourself. Any method will do, but the eyepoint must move smoothly to any 3D point, and smoothly 'look around' in any direction from that point. **(!!WARNING!!* the'starter' code' camera positioning is insufficient**.** If you use gluLookAt(), modify BOTH the camera position (VRP) AND the target or 'look-at' point independently.

9) Switch 3D Cameras: Your program must let users **switch back and forth between an orthographic and a perspective camera**, without changing viewing position or direction. 'Different cameras' mean you must change the GL_PROJECTION matrix; do not change GL_MODELVIEW. For extra credit, make your movable camera adjustable—try an architectural 'view' camera offers adjustable asymmetric left/right, top/bottom edges.

10) On-Screen Image/Bitmaps: Your program must show at least 3 user-movable images loaded from a file by your program, and transferred to the on-screen framebuffer using openGL commands. Fixed-pipeline commands are OK, but using Pixel Buffer Objects for each will earn extra credit!

Outside Sources: You are welcome to use any, all, or none of the 'starter code' I supplied; I guarantee it works: just modify it gradually step-by-step. You may also use 'starter code' from our book or others, from websites, or any other useful resource you can find, but you must credit their work properly—no plagiarism! List the URLs on CMS/Blackboard discussion board, etc. You must SUBSTANTIALLY MODIFY what you find--make this YOUR code, and not just a minor adjustment to the work of others.

Have fun with this!