EECS 351-1: Introduction to Computer Graphics
Instructor: Jack Tumblin

ProjectA: Shape

Winter 2014

Your mission in Project A is to:
a) learn how to draw moving, turning, jointed colored shapes with openGL's basic drawing primitives (lines, triangles, etc) using arrays of 3D vertex coordinates,
b) to use a GL_MODELVIEW-like matrix stack to transform them interactively,
c) ensure that parts of your on-screen image move continuously without user input (animation) and
d) make these parts and possibly other parts move in response to keyboard and mouse inputs.

You may choose to draw ANYTHING for your project that meets the listed requirements.
Our goal is to learn enough WebGL to make an *interactive* drawing that shows something *you* find interesting and compelling. An octopus? Fractal trees that grow, wave in the wind? An N-legged walking creature whose legs consist of M segments (M>2)? A stick-figure model of a human hand that opens and closes? A bicycle? A car? A pterodactyl? A helicopter?

Requirements: Project Demo Day (and due date): Mon Feb 3, 2014

In-Class Demo: on the Project's due date (Mon Feb 3) you will demonstrate your completed program to the class. Two other students will each evaluate your work on a 'Grading Sheet', as will Professor Tumblin or TA Paul Olczak. Based on Demo Day advice, you then have 72 hours to revise and improve your project before submitting the final version for grading. Your grade combines grading sheet scores and your improvements.

Submit your finalized project to Blackboard/CMS no more than 72 hours later (11:59PM Thurs Feb 6) to avoid late penalties. Submit just one single compressed folder (ZIP file) that contains:
a) your written project report as a PDF file, and
b) just one sub-folder that holds your source code: HTML, JavaScript, any libraries needed (e.g. 'lib' directory)
---IMPORTANT: Name your ZIP file and the directory inside as: **Familyname**Personalname_ProjA
For example, my project A file would be: TumblinJack_ProjA.zip, and it would contain my report file namedTumblinJack_ProjA.pdf, TumblinJack_ProjA.html, TumblinJack_ProjA.js ,etc,
---To submit your work, use Blackboard→Assignments. DO NOT e-mail your project!
---BEWARE! SEVERE LATE PENALTIES! (see Blackboard→Syllabus→'Course Details')

Project A consists of:
1)—Report: A short written, illustrated report, submitted as a printable PDF file.
Length: >1 page, and typically <5 pages, but you should decide how much is sufficient.
A complete report consists of these 3 sections:

a)--your name, netID, and a descriptive title for your project
(e.g. Project A: Planetary Gear Transmission, not just Project A)

b)—a brief 'User's Guide'. Begin with a paragraph that explains your goals, then give user instructions on how to run and control the project. (*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, HUD text shows velocity in kilometers/hour.") Your classmates should be able to read ONLY this report and easily run and understand your project without your help.

c)—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 figure captions and text explanations.

2)—User Instructions: When your program starts or when the user presses the F1 (help) key, print a brief set of user instructions onscreen somewhere. Could you put it in the webpage, outside of the 'canvas' element where WebGL draws pictures? Or within the 'canvas' element using the 'HUD' method in the book? or in the JavaScript 'console' window (in Google 'Chrome' browser)? You decide! Make your program self-

explanatory; for example, you might tell the user: 'arrow keys steer the skateboard, drag left mouse key to make robot crouch/stand; right mouse drag to aim the flame-thrower'.

3)—WebGL + Vertex & Fragment Shaders + VBOs: your program must use WebGL calls to draw something interesting and colorful on-screen in 2D (OK to define 3D shapes and move them in 3D, but don't use a 3D camera or show 3D perspective, textures, or lighting; that's Project B).
Your program ***MUST*** use non-trivial vertex and fragment shaders, and ***MUST*** draw shapes using WebGL drawing primitives (e.g. points, lines, triangles, triangle strips, triangle fans). Do not substitute 'canvas' drawing primitives (e.g. context.filledRect()), because your program MUST use vertex buffer objects (VBOs) to store the vertices that define your program's shapes, demonstrated in WebGL Programming Guide, Chapter 3,4). Create your own unique and colorful shapes, and make at least one with more than 12 vertices.

3)—Event Handlers: your program and its shaders should make proper use of registered event handlers for **keyboard, mouse and display**, as demonstrated in Chapter 3 and in the 'starter code' posted. Event handlers let your programs respond to the mouse, to changes in the display window size, to keyboard inputs, and more. You may wish to try other user controls in HTML, such as buttons, knobs, sliders, edit windows, etc.

4)—Matrix-Only Transformations: Your program and its shaders must create a GL_MODELVIEW-like combination of 4x4 matrices to transform the points of your object. Apply matrix transforms to sets of vertices to position, scale, and orient/rotate them in pleasing ways. Use matrix multiplies to combine separate 4x4 matrices that translate, scale, rotate vertices, and apply the result in your Vertex Shader.
Warning! I will not accept hard-coded expressions for any transforms, such as "x = x + xtrans; y = y + ytrans;"

5)--Motion! Like all projects in this course, your program must show a picture that moves and changes, both by itself (animation) and also moves/changes in response to user inputs (interaction) from mouse or keyboard. Users must be able to move objects, and animation must also move/pause them in interesting ways onscreen.

6)—At least two (2) or more Different *Kinds* of jointed, moving objects. Your project must demonstrate how trees of transformations can make jointed objects by nested 'drawing axes' or 'reference frames' in WebGL. You must construct and draw at least two different *kinds* jointed objects, each with at least 3 sequentially-connected hinged segments. For example, a one-legged hopping robot with 3 segments might have body, thigh, and shin segments. You'll need to 'push' and 'pop' matrices from your GL_MODELVIEW-like matrix stack to draw each kind of jointed object. Each and every joint in your jointed objects must rotate, yet the joints must stay pinned together as if connected by hinges and/or sliding joints; joints should not separate as they move and flex.

7)—Note opportunities for extra credit by adding more features to your project; see Grading Sheet.

Sources: BEGIN with any, all, or none of the 'starter code' supplied, or ideas from any book or website you like (e.g. openGL.org), and build up your own project from it. You MUST give proper credit to other people, books, or websites that help you: list names, URLs. If you find something good, share it with the class – post it on Blackboard→Tools→DiscussionBoard, etc. Of course, what you submit must YOUR OWN work, your own code, and not just a slightly tweaked version of the work of others.
(Submitting the work of others as your own prompts the Dean of Students to investigate you for cheating).