CPS511 Assignment 1

Using Geometric Transformations to Construct and Manipulate  
Multi-Part Models

(*Worth* ***12*** *percent of your mark*)

Due Date: Sun., Oct. 14 11:59pm



You will construct and interactively manipulate a complex multi-part model. This programming assignment will increase your knowledge of geometric transformations, shape modeling, and simple animation. **You must do this assignment individually – no groups. Do not attempt to find source code on the web for this assignment. It will not help you and you risk extremely serious academic consequences.** Begin designing and programming early! Start by reading this description carefully and studying the provided skeleton code. **If there is some part of the assignment you do not understand, please see me in class or email me as soon as possible and I will clarify the issue.**

# Program Functionality Requirements

Your program should use OpenGL transformations (**glTranslate**, **glScale**, **glRotate**, **glPushMatrix**, **glPopMatrix** etc) to construct a multi-propeller drone model as in the figures above. Your model does not have to look sophisticated – you are not marked on how good your model looks, however it should look something like a multi-propeller drone. In this first assignment, you do not need a camera as part of the drone – it will be added for assignment 3.

You may use the skeleton code provided with the assignment. This code sets up lighting and shading for you, and creates and draws a quad mesh for you to act as the ground plane. The skeleton code also sets up the material for your drone model. It creates a cube by calling glutSolidCube() and uses simple transformations to position and display it – you should replace this cube with your drone. The skeleton also provides code for a simple cube mesh for you to use in this assignment for the bonus if you wish or in future assignments. In the skeleton code the cube mesh is currently unused.

You may use the glut shape primitives to draw the parts:

glutSolidCube(size);

glutSolidSphere(r, nLongitudes, nLatitudes);

glutSolidCone(rBase, height, nLong, nLat);

glutSolidTorus(rCrossSection, rAxial, nConcentric, nRadial);

and/or the glu quadric primitives:

gluSphere()

gluCylinder()

gluDisk()

gluPartialDisk()

Keep in mind that the quadric surfaces have to be created using glu functions. The following lines of code are an example of how to use them:

GLUquadricObj \*mySphere;

mySphere = gluNewQuadric();

gluQuadricStyle(mySphere, GLU\_LINE);

gluSphere(mySphere, r, nLong, nLat);

This will be explained in more detail in an upcoming lecture. You can google these functions to get details on the parameters. You may also use your own meshes for the drone **parts as long as you are positioning them using the OpenGL transformation functions**.

### NOTE: You must use glTranslate, glRotate, glScale, and glPushMatrix, glPopMatrix, glLoadIdentity to implement all transformations in a hierarchical way!! See example program GeoTransII.

# Requirement 1

Your drone must consist of at least 3 propellers. Your propellers must spin. Use glTranslate, glRotate, and glScale to position the parts and construct the drone. You must also have at least one object in your scene that is static (doesn’t move with the drone). For example, place objects such as primitive buildings on the ground mesh provided in the skeleton code.

# Requirement 2

1. You must use a key to start the drone’s propellers spinning. I suggest the ‘s’ key.
2. You must use the keys (or the mouse) to control the forward movement of the drone. Use 2 keys to control incremental forward/backward motion of the drone (I suggest the ‘f’ and ‘b’ keys). Use glTranslate() to implement forward/backward movement. When moving forward/backward the propeller should turn.
3. Use 2 keys (I suggest the left and right arrow keys) to control incrementally turning the drone right or left. Use glRotate() to implement turning. After the drone turns the new forward direction should be in the direction it’s facing.
4. Use 2 keys to control vertically raising or lowering the drone (I suggest the up and down arrow keys). There is no requirement for pitching the drone on an angle when raising or lowering – the drone may remain horizontal. Feel free to make your drone look more realistic when flying.

# Requirement 3

Take a look at the posted example programs GeoTransII and 3DGeoTransI. Also review the OpenGL Geometric Transformation slides. Similar to these programs, you must add detailed comments to your code above (or alongside) all the function calls to glTranslate, glRotate, glScale, glPushMatrix, glPopMatrix, glLoadIdentity showing how the ModelView Transformation Matrix is being constructed. Use notation like:

**p' = I \* T(3,4,2) \* R(25, 0, 1, 0) p**

**You need to demonstrate that you understand how the CTM is being constructed.**

# Requirement 4

Implement the “help” key F1 that when pressed tells the user how to navigate the drone (which keys are used and their purpose). You can simply use printf to print text to the other window (console). Printing could also be useful for debugging!

# Note:

* You do not need to provide any **view** navigation (camera position/orientation change) capabilities – you may use a fixed camera view.
* You do not need to provide a window reshape capability.

# Optional Bonus (1 mark)

### Use an indexed VBO to draw your ground mesh (0.5 marks).

### For advanced students: use a vertex and a fragment shader for rendering (0.5 marks) instead of the fixed pipeline.

### The maximum bonus for the assignment is 1 mark.

# Grading (out of 15 marks)

|  |  |
| --- | --- |
| Multipart drone having at least 3 working propellers. Must use glTranslate, glRotate, glScale, glPushMatrix, glPopMatrix and glLoadIdentity to perform transformations of drone parts. | 8 marks |
| Simple animation via glTranslate and glRotate to move drone. Use keys (or mouse) to control forward/backward motion, left and right turns, and up and down. | 6 marks |
| Comments explaining CTM construction | 1 mark |
| Bonus – indexed VBO and/or vertex+fragment shader | 1 marks (max) |
| Total | 16 marks (max) |

# Program Submission

Use D2L to submit your assignment. Submit all your source files. You should use C for your program (.c files). The skeleton code is written in C. You may use C++ (.cpp files) but be careful of advanced object-oriented features – OpenGL is a non-object-oriented API. **Zip everything up into one file**. **Do not include executable files! Do not use RAR.** If your program runs under Windows, include a README file describing how to compile your program. If you want to inform the TA about your program (special features, bonus work etc.) include this information in program comments and the README file. If only part of your program works, list the parts that work and that do not work in the README file. Include all makefiles (for Linux/Mac) or solution/project files (for example, if you used Visual Studio). **It is your responsibility to ensure the TA has enough information so that he can, with little effort, compile and run your program**. I am being flexible in terms of your operating systems choice so you must make an effort to meet me halfway. If the TA has trouble compiling your program, he will have the discretion to deduct marks and/or will ask you to compile and run your program in person.