CPS511 Assignment 1

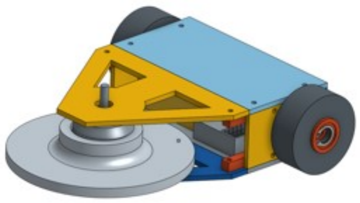
Using Geometric Transformations to Construct and Manipulate Multi-Part Models

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

Due Date: Thu., Oct. 15 11:59pm

Spinner 1

Spinner 2


Spinner 3

Hammer

Saw

SUBZERO
2020
Arm Flipper

Wedge Flipper

You will construct and interactively manipulate a complex multi-part model – a **BattleBot**. 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. Your program will be checked for plagiarism.

Program Functionality Requirements

Your program should use OpenGL transformations (**glTranslate**, **glScale**, **glRotate**, **glPushMatrix**, **glPopMatrix** etc) to construct one of the battle-bot models in the figure above. **You will each be randomly assigned a battle bot design and you must build that design (you cannot change).** Your battle bot model does not have to look exactly like the image in the figure. It does not have to look sophisticated – you are not marked on how good your model looks. However, it should be a simplified and recognizable version of your assigned bot.

You may use the hierarchical 3D robot example program as a guide for your assignment (i.e. you may use it as “skeleton” code). This code sets up lighting and shading for you and creates and draws a flat quad mesh for you to act as a ground plane. The example code also shows you how to set up the material for your model. It creates the robot by calling glutSolidCube() and other shape primitives and uses simple geometric transformations to construct it in a hierarchical manner. Use this hierarchical design for your bot. The skeleton code also provides code for a simple cube mesh for you to use in this assignment if you wish instead of glutSolidCube() or in future assignments. In this robot example the cube mesh is currently used as a target object.

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.

NOTE: you may also use your own meshes for battle-bot parts **as long as you are positioning them using the OpenGL transformation functions**!!!

Requirement 1

1. Your battle-bot must consist of at least 5 parts. You should designate one part as the base part. Each bot must have 2 or 3 or 4 visible wheels (or at least partly visible). The wheels must rotate as the bot moves forward and back. You may spin your bot to turn it. When turning, the wheels on one side rotate and the wheels on the other side do not. Each bot design has a weapon that can move. The spinner bots have rotating weapons. Use a key to control turning spinning on and off. The hammer bot rotates the hammer quickly to strike a blow when a key is activated. The saw bot is like a spinner. There are two flipper bots. Both should have their flipper on the ground as the bot moves. A key should activate the flipper and rotate it up quickly in a flipping action. NOTE: You must use glTranslate(), glRotate(), glScale(), glLoadIdentity() and glPushMatrix() and glPopMatrix() to implement all transformations!! Your code must be also hierarchical with nested push/pop function calls. See the example 3D robot program and the slides.

2. You must use a key to control your weapon. I suggest the space-bar key.

3. You must use keys (or the mouse) to control the movement of the bot. Use 2 keys to control incremental forward/backward motion of the bot (I suggest the ‘f’ and ‘b’ keys). Use glTranslate() to implement forward/backward movement. When moving forward/backward the wheels should turn.

4. Use 2 keys (I suggest the left and right arrow keys) to control incrementally spinning your bot so that you can turn in in a new direction. Use glRotate() to implement turning. After the bot turns the new forward direction should be in the direction it is facing.

Requirement 2

You must add another object to your scene using the cube mesh code. You may also use your own mesh. You may use immediate mode rendering to draw your battle-bot and ground mesh and cube mesh. **However, to obtain full marks in this assignment you must use indexed VAOs or VBOs to draw your meshes (ground mesh, cube mesh).**

Requirement 3

Implement the “help” key F1 that when pressed tells the user how to control the bot (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: this requirement is worth 0 marks but you will be assigned 0 if you do not implement it. The TAs need to be able to interact with your program easily.

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)

1) Construct and manage your own 4x4 transformation matrices and use glLoadMatrix() , glMultMatrix() etc to set the CTM in OpenGL (1 mark)

2) For advanced students: use a vertex shader and a fragment shader for rendering (1 mark) instead of the fixed pipeline. The vertex shader should animate at least one of your meshes (cube mesh or ground mesh) (hint: you could use a sine wave).

NOTE: The maximum bonus for the assignment is 1 mark.

Grading (out of 12 marks)

Multipart battle-bot with working weapon. Must use glTranslate, glRotate, glScale, glPushMatrix, glPopMatrix and glLoadIdentity to perform transformations of bot parts in hierarchical fashion.	6 marks
Simple animation via glTranslate and glRotate to move battle-bot and weapon. Use keys (or mouse) to control forward/backward motion, left and right turns. Wheels turn properly	5 marks
Immediate Mode Rendering for all meshes **OR** Indexed VBO and/or vertex+fragment shader for one mesh	0.5 marks **OR** 1 mark
Help key	0 marks (REQUIRED!)
Bonus	1 mark max
Total	13 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 they can, with little effort, compile and run your program**. I am being flexible in terms of your operating systems choice so you must try to meet me halfway. If the TA has trouble compiling your program, they will have the discretion to deduct marks and/or will ask you to compile and run your program “in person”.