Queen Mary
University of London

# EBU5405: 3D Graphics Programming Tools
# Coursework Part1: Labs 4 & 5

## 1. Introduction to the EBU5405 coursework

### 1.1. Aim

In this coursework you will develop an **articulated and interactive object in the shape of an animal of your choice** (hereafter called the "object") using C and OpenGL.
The main objectives of this project are to practice and demonstrate: hierarchical modelling, model rendering, viewing transformations, lighting, interactivity and animation using OpenGL and the GLUT.

### 1.2. Coursework Part 1 (Labs 4 and 5) and Part 2 (Labs 6 to 8)

The coursework is divided into two parts. The present document is a general introduction to the coursework followed by a detailed description of the coursework part 1 (labs 4 & 5). Details of coursework part 2 (labs 6 to 8) will be provided in a separate document.

During the first part of the coursework, you will develop the 3D hierarchical model of your articulated object and implement some rendering and interactivity. In the second part of the coursework, you will add lighting to your object and create an animation.

The outcome of each lab should be saved in a folder with the name of the lab (i.e. Lab4, Lab5, etc.). In each folder, you should include the source code, the makefile, and the compiled application. We should be able to run and understand your code (please add comments!).

The outcome of the coursework (i.e. all the saved five lab folders) should be placed in a folder named **jp09xxxxEBU5405** (replace jp09xxxx by your student number). This folder will be submitted via Blackboard after lab8 (i.e. there is only one submission). Details of the deadline can be found on Blackboard.

### 1.3. Practical details

- Individual coursework: your animated articulated object must be unique!
- Supported by lab sessions
- Two parts but one submission only at the end of Lab 8 (see details of the submission and cutoff dates on Blackboard).
- The coursework is worth 20% of the final mark and will be marked out of 20.
- Late submission penalties: -1 mark per day late, up to -5 marks (5 days).
- No extension will be granted and no submission will be accepted after the cutoff date unless valid extenuating circumstances (ECs) have been submitted and accepted following the usual EC submission process.

# 2. Lab 4

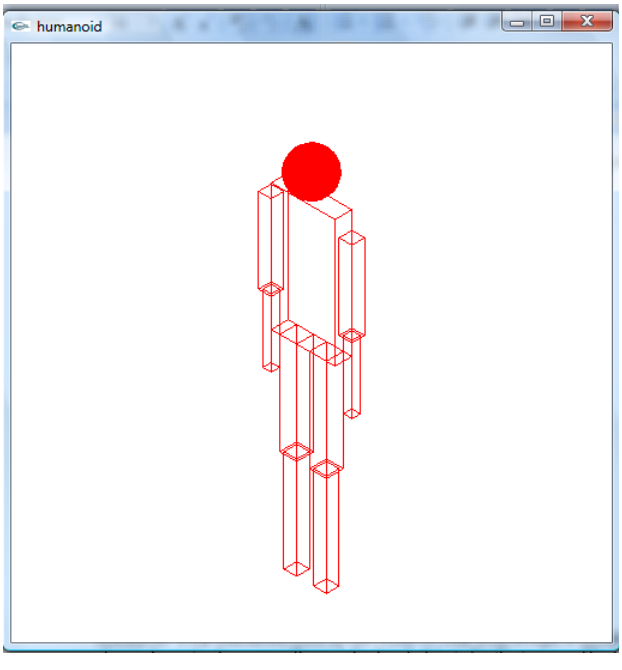
## 2.1. Aim

The aim of Lab 4 is to build the articulated object.

The basic requirements for the articulated object is that it should:

- be made of **at least 8 articulated** parts;
- present **some dependencies between the parts**;
- evoke the **shape of a known animal** (e.g. a cat, a dog, a giraffe, an elephant, etc.).

See in the figure below an example of a humanoid (this is only for illustration; your articulated object cannot be a humanoid). It is made of 10 parts: the head, the torso, the 2 upper arms, the 2 lower arms, the 2 upper leg parts, the 2 lower leg parts. The head and upper limbs are attached to the torso. In addition, the lower limbs are attached to their corresponding upper limb. This hierarchy creates 9 joint angles (as the head is fixed on the torso).




Attention! Your articulated object must be the result of your own design, it should not look like the example above.

## 2.2. Content of Lab4

### 2.2.1. Object parts

You should implement each part in a separate function.

If you decide that some of the parts should be cylindrical objects, you can use the GLU quadrics objects (e.g. gluCylinder) to implement them.

The GLU quadrics objects are based on Quadrics, which are a way of drawing complex objects. To use Quadrics, you must first declare a variable:

**GLUquadricObj *p;** // pointer to quadric object

Initialise it:

**p = gluNewQuadric();** // initialisation

And then use it to create the GLU quadrics object, for example:

**gluCylinder (p, 1.0, 1.0, 3.0, 32, 32);** /* cylinder of radius 1.0 at its base; radius 1.0 at its top; height of 3.0; 32 slices (subdivisions around the z axis); 32 stacks (subdivisions along the z axis) */

For a reference to the GLU quadrics objects, please see here:

http://pyopengl.sourceforge.net/documentation/manual/reference-GLU.html

#### 2.2.2. Modelling transformations

Write a function called "object". In this function, you will write the modelling transformation statements (e.g. glRotatef and glTranslatef) that will allow you to build the object using its individual parts.

#### 2.2.3. Display callback and isometric view

The "object" function above will be called in the display callback. But before you call this function, make a call to the gluLookAt function in order to create an isometric view of the object (as illustrated in the humanoid figure above).

#### 2.2.4. Reshape callback

Write a reshape callback in which you should call the glOrtho function. Choose carefully the arguments of the glOrtho function so that the whole object appears in the middle of the window.

### 2.3. Outcome of Lab4

The outcome of this lab should be a static 3D graphics, which shows an isometric view of the object.

Save your work in a folder named Lab4, include all your code, makefile and .exe application. Do not forget to comment your code.

# 3. Lab 5

## 3.1. Aim

The aim of Lab 5 is to interact with the object. For this, you will need to revise the outcome of lab4 to insure that the object is modelled as a hierarchical object.

## 3.2. Content of Lab5

### 3.2.1. Hierarchical modelling

Think carefully about the dependencies between the individual parts of the object. Refer to the slides of the HierarchicalModelling tutorial and revise the object function you developed in Lab4 accordingly. Remember that to model the object as a hierarchical object, you must use the matrix stack.

Hint: In the individual modelling functions, it may be a good idea to insure that the end points of any future rotation (see details of the interaction below) stand at the origin of the coordinate system. The limbs of a humanoid for example will rotate from an end point located at the top of the limb, so the limbs should be modelled with their top at the origin.

### 3.2.2. Menu based interaction

You will write a menu which will allow you to test the hierarchical modelling of your object. The menu will be used to modify the position of the object.

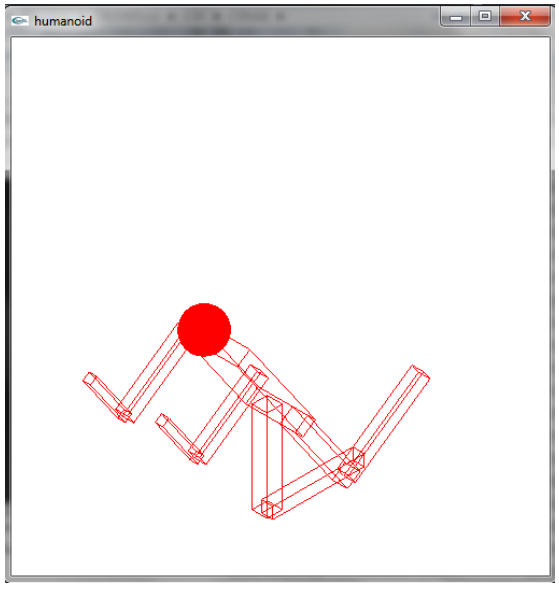
The menu should be attached to the right button of the mouse and show the following three options: **Translating, Bending and Rotating**.

### 3.2.3. Translating

When the Translating option of the menu is chosen, the object should be animated to change its position (not its shape) inside the window. The animation should only stop when the option is selected again in the menu.

### 3.2.4. Bending

When the Bending option of the menu is chosen, the articulated object should be animated so the various articulated parts are moving. The animation should only stop when the option is selected again in the menu. For illustration purposes, the figure below is showing a “bending” humanoid.



#### 3.2.5. Rotating

When the Rotating option of the menu is chosen, the object should rotate on itself along the Y axis. The animation should only stop when the option is selected again in the menu.

### 3.3. Outcome of Lab5

The outcome of this lab should be an interactive animated 3D graphics which allows us to change the position, shape and orientation of the object.

Save your work in a folder named Lab5, include all your code, makefile and .exe application. Do not forget to comment your code.