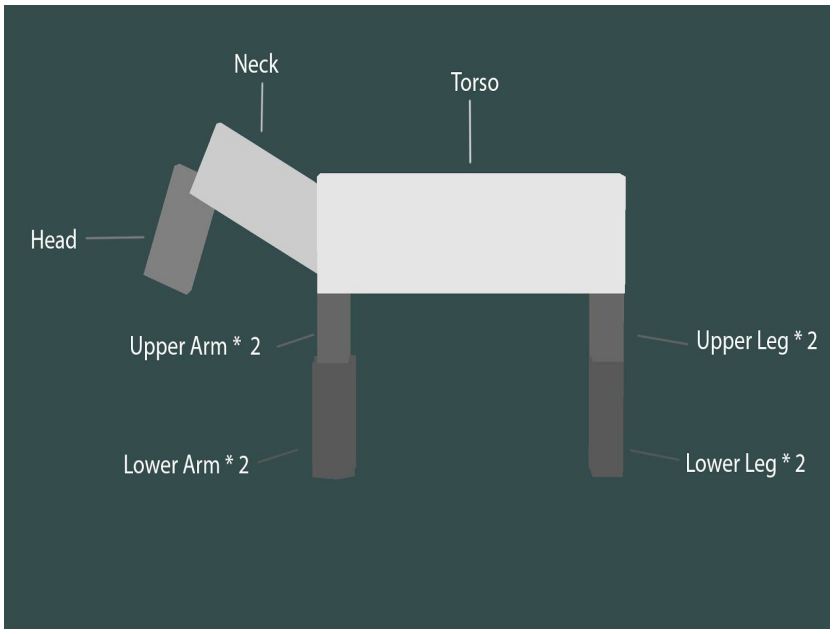
Programming Assignment 1

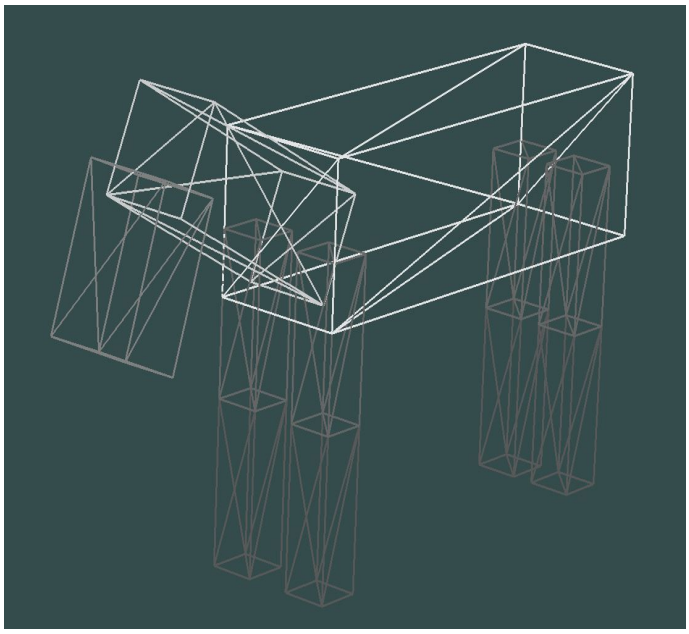
Announcement:	**Session # 2**
Submission Deadline:	**Session # 5**

Description

This OpenGL programming assignment will provide an introduction to OpenGL programming. More specifically, you will learn how to create simple virtual scenes consisting of objects, display them as wireframe meshes by drawing triangles, and also how to set up and manipulate the virtual camera to view the scene from different angles and distances.

A **mesh** is a set of polygons that share vertices and edges which describe the shape of a geometric object. In this assignment you have to first create a unit cube at the origin, then create a 3D version of a robot horse with head, neck, torso and legs (front_upper_right_leg, front_lower_right_leg, hind_upper_right_leg, hind_lower_right_leg, front_upper_left_leg, front_lower_left_leg, hind_upper_left_leg, hind_lower_left_leg). (see figure: side-view [left] perspective-view[right]).





Implementation Specifications

Develop an OpenGL application with the following functionality and features:

- Creates a 100x100 square grid (ground surface) in the XZ plane centred at the origin.
- Creates a set of three lines 5 grid units in length, in 3 different colors, representing each coordinate axis in virtual world space, centered at the origin.
- Creates a model of the horse like the one shown in the figure by suitably transforming **One** unit cube to create the different body parts. The horse is initially positioned at the centre of the grid facing along X axis with its feet on the ground. Different body parts in slightly different color.
- Places a virtual camera with the world space origin as the point of focus.
- For display and animation:
  - create a GLFW window of size 800x800 with double buffering support.
  - render the coordinate axis, ground and horse in the window.
  - The application should use a perspective view to display the objects and use the depth buffer for hidden surface removal.
- The application should handle the following input:
  - Pressing the spacebar should re-position the horse at a random location on the grid.
  - The user can incrementally size up the horse by pressing ‘U’ for scale-up and ‘J’ for scale-down. Each key press should result in a small size change.
  - The user can control the horse position and orientation using keyboard input i.e. A → move left 1 grid unit, D → move right 1 grid unit, W → move up 1 grid unit, S → move down 1 grid unit, a → rotate left 5 degrees about Y axis, d → rotate right 5 degrees about Y axis, w → rotate upwards 5 degrees raising the front legs, s → rotate downwards 5 degrees raising the hind legs.
  - The world orientation is changed by using keyboard input i.e. left arrow → R_x, right arrow → R_{-x}, up arrow → R_y, down arrow → R_{-y}. Pressing the “Home” button should reset to the initial world position and orientation.
  - The user can change the rendering mode i.e. points, lines, triangles based on keyboard input i.e. key ‘P’ for points, key ‘L’ for lines, key ‘T’ for triangles. The user can pan and tilt the camera as follows:
    - while right button is pressed → use mouse movement in x direction to pan; and
    - while middle button is pressed → use mouse movement in y direction to tilt.
  - The user can zoom in and out of the scene - while left button is pressed → use mouse movement to move into/out of the scene.
  - Window resize handling: The application should handle window resize events and correctly adjust the aspect ratio accordingly. This means that the meshes should not be distorted in any way.
- The application should use OpenGL 3.0 and onwards, and include brief comments explaining each step.

Submission (electronic submission through Moodle only)

Please create a zip file containing your C/C++ code, vertex shader, fragment shader, a readme text file (.txt). In the readme file document the features and functionality of the application, and anything else you want the grader to know i.e. control keys, keyboard/mouse shortcuts, etc.

Additional Information

- You can use the skeleton code provided during the lab sessions to get started.
- A video demonstrating the functionality is posted on YouTube: **https://www.youtube.com/watch?v=5pcc_O-5110&t=8s**

Extra Credit (10% points)

(1) A more detailed model of the horse which could later enable more realistic movement if the horse were to be made to trot or gallop.

Evaluation Procedure

You MUST demonstrate your solution program to the lab instructor during lab hours. You must run your submitted code, demonstrate its full functionality and answer questions about the OpenGL programming aspects of your solution. Major marking is done on the spot during demonstration. Your code will be further checked for structure, non-plagiarism, etc. However, ONLY demonstrated submissions will receive marks. Other submissions will not be marked.