EBU5405 Learning Checklist

This page details the main learning objectives for this module on a topic by topic basis. This is what you should be able to do by the end of the module. Tick them off as you are confident you can do them.

0: Getting Started (what you should be able to do at the start of the module)

□ use the C language to do basic things like declaring arrays; reading from files; using pointers; declaring structures; and writing decision, loop and case control statements.
□ compile and run C programs (Eclipse IDE is recommended)
□ do basic matrix calculations (addition and multiplication)
□ understand and use simple geometry and geometric calculations such as: points and vectors; affine and Euclidian spaces; convex polygons; similar triangles; dot products and cross products.

1: 3D Graphics programming

□ give my own definition of 3D Graphics programming
□ explain the terms: modelling, rendering and imaging
□ explain the benefits of a pipeline architecture
□ describe what each step of the 3D Graphics pipeline does
□ describe the natures of the input and output of the 3D Graphics pipeline
□ explain what a 3D Graphics transformation is
□ explain the role of the “viewer” (virtual camera) in 3D Graphics programming
□ give examples of 3D Graphics programming applications

2: OpenGL basic programming (2D)

□ link the three openGL librairies to my C projects (Eclipse IDE is recommended)
□ explain why OpenGL uses three libraries
□ compile and run a simple OpenGL program that opens a window on my display
□ write a display callback function to change the window background colour
□ use OpenGL primitives to make graphics appear inside the window
□ explain how the above programs work and the concepts involved
□ recognise, from the name of an OpenGL function: the library it belongs to, its purpose, and the number and type of parameters it requires
□ explain how OpenGL operates as a state machine
□ use OpenGL functions to change the variables’ default values for: colours, window parameters and viewing
□ recognise if an OpenGL function is a “state-changing”, “primitive-generating” or “query” function
□ use the glOrtho function to change the size and position of the graphics inside the window
□ use viewports
□ explain how OpenGL is event-driven
□ write callback functions to animate and interact with graphics using the mouse and the keyboard
□ use double buffering
□ write a reshape callback function to preserve the aspect ratio of my graphics
□ use menus

3: Modelling
□ explain the differences between point-based, surface-based and constructive modelling
□ give my own definition of a polygon mesh
□ explain the differences between boundary representations and polygon meshes
□ describe how polygon meshes are represented
□ explain why polygons are used for modelling
□ describe the properties polygons should have when used for modelling
□ write a polygon mesh to model a simple geometric figure
□ write programs that can draw graphics using polygon mesh models as inputs
□ compare the properties of some of the widely used 3D file formats: STL, OBJ, FBX, COLLADA, VRML and X3D

4: OpenGL programming: moving to 3D
□ describe the default parameters of the OpenGL camera
□ use OpenGL functions to change the default parameters of the camera
□ explain what hidden surface removal is
□ describe how hidden surface removal is implemented in OpenGL
□ use OpenGL functions to request hidden surface removal
□ explain how hidden surface removal contributes to the visual perception of 3D shapes

5: Modelling transformations
□ explain the difference between a model (or prototype or symbol) and an instance
□ explain how modelling transformations result in a change of coordinate system
□ explain why matrices are used to represent transformations
□ write the general form of a scaling transformation matrix
□ write the general form of a 2D rotation around the origin transformation matrix
□ explain why homogenous coordinates are used
□ compare the properties of linear and affine transformations
□ write the general form of a translation transformation matrix
□ write the general forms of 3D rotation transformation matrices along the 3 canonical axes
□ explain how matrix composition by post multiplication makes transformations efficient
□ use OpenGL functions to request modelling transformations
□ write in OpenGL a general rotation transformation using a combination of canonical rotations
□ write in OpenGL a rotation transformation about a fixed point other than the origin
□ use the OpenGL matrices
□ explain the role of the OpenGL "Current Transformation Matrix" (CTM)

6: Hierarchical modelling
□ write the hierarchical model of a simple articulated 3D object
□ explain stack-based traversal
□ use the OpenGL matrix stack
□ write a program to create an animated articulated graphical 3D object

7: Viewing
□ explain how viewing transformations result in a change of coordinate system
□ use the OpenGL MODEL-VIEW matrix
□ explain why modelling and viewing transformations share the same OpenGL matrix
□ use the gluLookAt function
□ achieve the isometric view of an object using the gluLookAt function

8: Colours and Lighting

□ understand the terms hue, saturation and brightness
□ describe a colour (hue, saturation and brightness) expressed in the RGB model
□ describe a colour (hue, saturation and brightness) expressed in the HSV model
□ use OpenGL functions to change the drawing colour
□ use OpenGL functions to reset the colour buffer
□ use OpenGL functions to change the shading model
□ use OpenGL functions to change colours' transparency and blending properties
□ understand the terms illumination, lighting and shading
□ explain the difference between empirical and physical-based lighting models
□ explain how lighting contributes to the visual perception of 3D shapes
□ explain the difference between direct and indirect illumination
□ describe the properties of an ambient light source
□ describe the properties of a distant light source
□ describe the properties of a point light source
□ describe the properties of a spot light source
□ describe the properties of diffuse reflection
□ compute diffuse reflection
□ describe the properties of specular reflection
□ describe the Phong model's components
□ use OpenGL functions to enable lighting calculations
□ use OpenGL functions to set up various types of light sources
□ use OpenGL functions to set up various material properties
□ calculate the colour of a 3D object from reading the OpenGL code used to set up the light sources and the object's material properties

9: Projection

□ explain how projection transformations result in a change of coordinate system
□ describe the properties of parallel projection
□ describe the different types of parallel projection
□ write the matrix of a simple orthographic projection transformation
□ explain the screen-space transformation matrix
□ describe the properties of perspective projection
□ explain what is foreshortening
□ explain what a vanishing point is
□ write the matrix of a perspective projection
□ use OpenGL functions to set the parameters of orthographic projection
□ use OpenGL functions to set the parameters of perspective projection

10: Rasterisation

□ explain what rasterisation is
□ explain how triangulation can be done
□ recognise a good triangulation result from a bad one
□ explain the edge-walking rasterisation optimisation technique
□ explain why convex polygons are easier to rasterise than concave polygons
□ discuss triangle rasterisation issues
□ explain the edge-equations rasterisation optimisation technique
□ discuss the use of a parity test for the rasterisation of random polygons
□ discuss the use of an active edge table for the rasterisation of random polygons