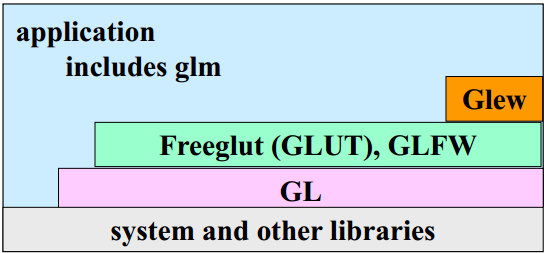
COMP 465 – Warbird Simulator Phase 1 Documentation

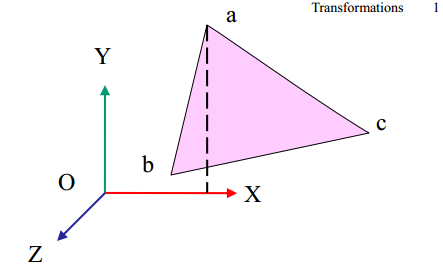
1. OpenGL is a cross-language, cross-platform application programming interface (API) for rendering 2D and 3D vector graphics. It’s a state machine in the sense that it runs in states that the developer turns states on and off. Uses C++ and C programming languages. C++ is used for its object oriented nature to create classes for visual objects. C is used for glut function calls and OpenGL function calls.
   1. Glew
      1. Included before gl or glut. Manages function pointers for OpenGL so we want to initialize GLEW before we call any OpenGL functions.
   2. Freeglut (also known as GLUT)
      1. Creates and manages the window that we draw in and handles input processing.
      2. Simple open-source library.
   3. GL
      1. These are the Core OpenGL function calls.
   4. Glm
      1. Is math library that handles all the mathematical functions in manipulating our 3D-world.
2. Main OpenGL functions
   1. Main()
      1. Has a number of tasks:
         1. Initializes and opens a window.
         2. Initialize the buffers and parameters by calling init()
         3. Specify the callback functions for events.
         4. Enter an infinite event loop.
      2. GLUT must first be initialized and configured.
      3. This class is set to use OpenGL version 3.3.
      4. Create the title for the window. The professor recommended putting the list of keyboard commands in the title so that we know how to manipulate the window.
      5. Initialize GLEW.
      6. Initialize scene with init().
      7. Set glut callback functions:
         1. glutDisplayFunc(display)
         2. glutReshapeFunc(reshape)
         3. glutKeyboardFunc(keyboard)
      8. Set the glutIdleFunc to be called when there are no other events pending. Used for animation timings.
      9. glutTimerFunc() is called with a specified timerDelay and passes an argument to a function to handle the logic of updating the screen every so often.
      10. glutMainLoop(), passes control to GLUT.
   2. Init()
      1. Called after OpenGL is initialized. Is only called once to maximize efficiency.
      2. First load the shader programs. In phase 1 we use the basic fragment and vertex shader files that the professor has given us.
      3. Need to generate VBO’s (Vertex Buffer Objects) to hold the vertices and color values for our models that are then held in a data structure called VAO’s (Vertex Array Objects).
      4. Load the buffers from the model files. Scale them based on their bounding radius.
      5. Load the ModelViewProjection matrix from the shader program.
      6. Initialize the shader program variables.
      7. Initialize the parameters for all the cameras.
      8. Set render state values
         1. glClearColor() clears the screen and establishes what color the window will be cleared to.
      9. Get the elapsed system time.
   3. Display()
      1. Required by FreeGLUT, will call this function when it detects that the screen needs to be rendered to. Is a callback function meaning that it’s event driven calling.
      2. glClear() clears the window to color specified in glClearColor()
      3. For loop to go through processing every model.
      4. Need to use glBindVertex array to set the model for its instance. Its rebinded everytime the model is changed in any way.
      5. Handle the logic for how each model is drawn based on which model it is.
      6. ModelViewProjectionMatrix is what handles drawing the entirety of our world through the projectionMatrix, mainCamera, and our models.
      7. glDrawArrays() draws all the vertices for our models. Our models are made up of triangles which is why we use GL\_TRIANGLES as part of the argument.
      8. Since we are using double buffer we use glutSwapBuffers(). Double Buffering has the current frame drawn on one buffer on the GPU, and the next frame drawn on the other buffer. To get the next frame the buffers need to be swapped. Using double buffering allows us to achieve smoother animations.
      9. Finally, we update the frame count based on the elapsed system time and update it in the window title.
   4. Keyboard()
      1. Called by FreeGLUT whenever the user presses a key. Callback Function. We use the v or x keys to cycle through our various camera positions.
   5. Reshape()
      1. Called by FreeGLUT whenever the window is resized. Callback Function. Indicates what action should be taken when the window is resized.
3. Our custom functions
   1. Update(int i)
      1. Is called to animate the scene objects by updating the rotation matrices .
      2. Needs to call itself again with glutTimerFunc().
      3. Calls glutPostRediplay() in order to explicitly tell OpenGL to redraw the scene after updating the rotation matrices.
   2. switchCamera(int camera)
      1. Handles the logic for switching and managing the cameras.

# Figures and diagrams:

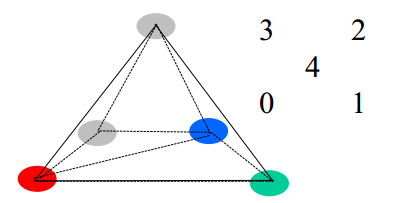
**OpenGL Application’s architecture**



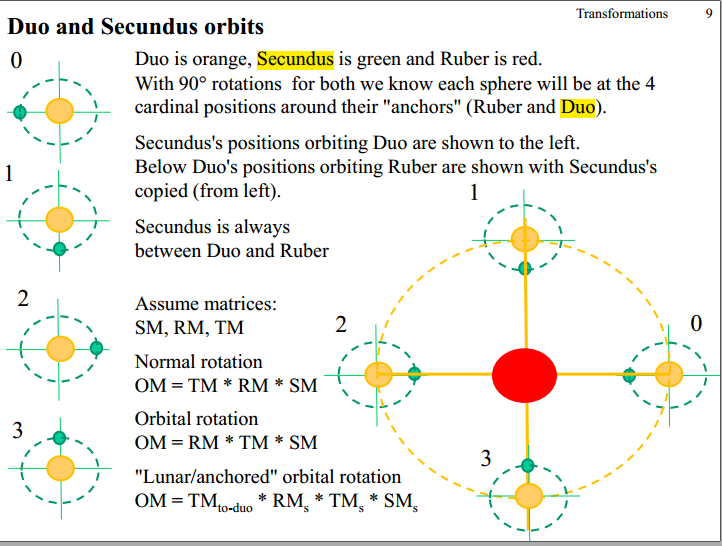
# Coordinates for a 3D model:



**How the models are drawn on the screen, in a wireframe view:**



**How the formula for the moon’s orbits work:**

****

The **TMto-duo** is the **transformation matrix** **for Duo**’s model which contains all the changes that were made to that model before the moon’s model was called. With that information the moon’s position is augmented to rotate around Duo’s model. **RMs** = is **Secundus’** (moon to Duo) **rotational matrix** to rotate Secundus’ model around Duo. **TMs** = is **Secundus’** (moon to Duo) **translation matrix** to translate the model position away from Duo’s model. **SMS**= the **scaling matrix for Secundus’ model** so that the model is the appropriate size.

**Building/Running the Project (beyond the obvious):**

Developed on the Windows Platform.