

All functions in the OpenGL Utility Toolkit have the prefix “glut”.

Initialization functions are preceded by “glutInit” and include the following:

`glutInit(int *argc, **argv);`

- These are the same values given to the program via the main function

`glutInitPosition (int x, int y);`

- Both parameters are the suggested number of pixels from the left (top) of the screen. Using -1 will leave this position up to the window manager

`glutInitDisplayWindowSize (int width, int height);`

- Suggested height and width of the window in pixels.

`glutInitDisplayMode (unsigned int mode);`

- A boolean combination (OR bit wise) of predefined values. For example, GLUT_DOUBLE is required to have a smooth animation (prevents flickering) and GLUT_DEPTH specifies a depth buffer (there are others). We’ll use the examples for now.

Once these values are set, we can then create the window with by calling the `glutCreateWindow (char*title)` function. Note the name of the window as it appears to the user is defined here.

We are still not done. Two more things need to be set up. First, we need to develop a function to display the scene and tell GLUT which function it is. Giving the function name via the `glutDisplayFunc`, which is registering a callback, accomplishes this. There are also display functions in the example, but these can be looked up independently.

Finally, we run the event manager never ending function `glutMainLoop()`. This will have to be halted either by a user event (hitting escape for example) or quitting the application.

Suppose now we want to resize the window. Because GLUT is assuming width/height = 1, the triangle will only appear properly under this condition. To fix it, we can write a function to change the perspective appropriately and give this callback function to GLUT using the function `glutReshapeFunc`.

More importantly, this will run when the window is first created, so if the window manager decides to give you a size that is not consistent with the one you suggested, your scene will still render properly.

In our `changeSize` function, a few new GLUT and non-GLUT functions are introduced. These deal with setting the current matrix to be the one desired and initializing it with the identity matrix, which is always a good idea just to be safe.

The non-GLUT functions are the OpenGL Utility Library function for setting the perspective (`gluPerspective`) and the camera position and orientation (`gluLookAt`).

Note the last three parameters of the up vector can be changed to turn things upside down (pretty cool, give it a try)!

For animations (we will do a simple one), we need to tell GLUT how to render a scene when it is idle, and in that update function we need to change the settings so the object “moves.” This can be done using “`glutIdleFunc`,” which can be the same function as the display one.

If we would like to add keyboard interaction, we can do so with the functions `glutKeyboardFunc` and `glutSpecialFunc`.

`glutKeyboardFunc` is used for the “normal” keys, any key that has an ASCII value. In addition to providing the key, it will also provide the mouse position in relative to the corner in x/y coordinates.

The function `glutGetModifiers` can tell you if shift, control or alt are also pressed with our desired key. In the example in lecture, this key is in tandem with up.

One limitation of tracking keys, however, is performance may be affected. To improve this, we can turn off checking for multiple keys by giving 1 as a parameter to the function `glutIgnoreKeyRepeat`.