User Interactions in OpenGL and FLTK

From the screenshots above, you can imagine how a tool with widgets can help an artist get the scene just right.

Description:
Previously we’ve rendered 3D shapes with an interactive interface. Today we will be exploring more of the functionality of the FLTK library. Our goals are to understand how event-based systems work. This lab will serve as an introduction to building interfaces and building tools useful for interactive graphics applications.

Your Task:

- You will start with the interface displayed in the left image above. Add mouse and keyboard and interactions:
  - One functions to modify: handle()
    - Left/Right Mouse should scale or zoom into the model
    - Or, in my case, I use the mouse wheel
    - Transform the camera with ‘w’, ‘a’, ‘s’, ‘d’ keys. (Move the camera forward back, and side to side)
  - To add or modify widgets, you will need to edit the constructor of the MyAppWindow class.
- Use your imagination to add at least one additional functionality (and/or widgets). Here are some ideas (but you should try to think of your own!)
  - Color control (Right image above is created by adding color control to the user interface)
  - Non-uniform scaling
  - Automated animation (e.g. rotation, translation)
- Understand how callbacks work in FLTK
  - Work with the event handler
- Spend some time looking at the source code to see how things are done.
- Be creative, this lab is more open ended!

Files Given:

- main.cpp – You will be working in the main.cpp. Within the main() function you will find lots of examples of the interface.
- ply.h, ply.cpp, geometry.h – These are all helper files you do not need to modify.
C++ Refresh:

Passing by pointer

1. Given an public variable, float boo, if you want to pass the value of the variable to another function, it’ll look something like: func (boo);
2. But if you want to pass a pointer of boo to the function (so that the function can modify the value of boo), you pass the pointer of boo: func (&boo); Inside the function, you will then modify the value of boo with something like: *boo = 100;

Callback (How these work)

3. A callback is a function that is passed as an argument in another function. In FLTK we have functions that catch keyboard and mouse events for example. When they are triggered, we can call into a function to perform an appropriate action.
4. In FLTK the callback functions to GUI events have a free variable (named userdata in the example code). You can use that free variable to do whatever you want.

Finished early?

- For this lab, consider how you might load multiple PLY objects into a listbox, and then be able to select which ones properties you are editing.
- Add some other manipulations so you can move the camera around (Have a camera look at function)
- Look for some FLTK examples and find interesting widgets to add.
- Try to make the keyboard movements smooth.
  - (Try adding acceleration or deceleration)
- Add some textboxes so we can input the global position, rotation, or scale of our model.
- Thing about your favorite videogame and what each mouse or keyboard command could do.
  - Not a PC gamer? How would you simulate a joystick or other device using the mouse, keyboard, or other widgets?

Going Further:
There are many similar windowing libraries to FLTK for OpenGL such as wxWidgets, QT that offer many other widgets for your interface. Beyond the user interface, you may also consider other input devices (joysticks, controllers, wiimotes, haptic devices, gesture based movements, and more).