oh haiiiii CCLab!!1!!!!!

using objects and addons in OpenFrameworks

liz_rutledge
dec 4, 2012

about me

liz_{DT} =

creative coding

web development

visual design

data visualization

liz$_{now}$ =

UI design

data visualization

web development [front-end]

and soon...Disney!

Processing

vs.

OpenFrameworks

Processing

quick and easy to get up and running

great documentation and helpful community

OpenFrameworks

more control over system functions (lower-level)

not as processor-intensive “pound for pound”
(faster and won’t turn your laptop into a frying pan)

tons of crazy addons

small differences that can create big headaches:

multiple file types

(.h and .cpp)

file structure is CRAZY critical

a place for everything and everything damn well BETTER be in its place

testApp::setup( ) things you need to specify

ofSetVerticalSync(TRUE);
ofSetFrameRate(30);
ofEnableAlphaBlending( );
ofEnableSmoothing();

it’s even more important to watch out for these little “gotchas” when you start using things like objects and addons!

objects

why are objects so important in OpenFrameworks?

efficiency AND consistency

condense hundreds of lines of code and create consistency across entire groups of objects

modularity [reduce, reuse, recycle!]

allows you to reuse objects across multiple unique projects, or adapt objects from other open source projects to your own project

entities with a mind of their own

can create rules that allow your objects to interact with each other without your active involvement
(think particle systems!)

yayyy examples!!!

making a class [and how to get all those files to talk to each other]:

each class gets TWO files
(one .cpp and one .h needed per class)

each .cpp files ONLY links to its matching header file
#include “className.h” or “#include “testApp.h”

ALL .h files need to link to the ofMain header file AND include either #pragma once or all that #infdef crap
#include “ofMain.h”

testApp.h needs to ALSO include the header file of any included classes (same goes for any type of parent class)
#include “className.h”

making a class: annoying details you CAN'T forget to do

header files need to have a conditional header to protect against loading the same classes more than once.

if you're in testApp.h, then a "#pragma once" will do.

```
#pragma once
```

if you're in className.h, then use the following:

```
#ifndef CLASSNAME_H
#define CLASSNAME_H
#include "ofMain.h"
[...rest of your class code...]
#endif
```

readyyyy...go!

addons

addons extend the core functionality of the larger OpenFrameworks library by “adding on” pre-bundled collections of libraries

these can contain just one class, a system of classes, or can even contain unrelated C++ libraries that are bundled together with a class that makes it easier to access that functionality through OpenFrameworks. Usually they’ll even have an example project to work from!

standing on the shoulders of [open-source] giants

using addons lets you get to work on your killer concept instead of spending time figuring out how to (for example) access your computer’s microphone. The whole point of the oF community is to make cool shit while helping other people make cool shit.

That’s why they’re there!

yessss more examples!!

using an addon: setting it up

download (and unzip) the addon from the oF site and rename the folder to fit the ofxAddonName convention

this will simply mean deleting the author's name before the first hyphen and deleting any other weird verion number type stuff on the other side of the second hyphen

from the Finder, move (or copy) any example files from inside the addon folder into the apps folder on the same level as your other project folders

this means the example folder you're dropping will end up in another directory that is inside the apps directory

if you're adding to an existing project, drop the folder into your addons folder from inside your project file

nothing fancy here. Just make sure you can see it!

using an addon: tying it into your project

open the example to see how they're making use of the addon's functionality

from the example file, you'll be able to see how the files are linked, and how different types of functionality are achieved. If you haven't started your project yet, you can even duplicate the example and start from there. Otherwise, you'll want to use it as a reference.

link to the addon files from the testApp.h header file so that it knows where to find any function calls to those libraries

if you started from the example, this will be done for you. If not, then include:

#include "ofxAddonName.h"

at the top of your testApp.h right along with #include "ofMain.h" and any other classes you may be including.

heeeere we go!

thanks!!!

liz_rutledge
esrutledge@gmail.com
http://lizrutledge.com