# Welcome!

Instructor: Nina Freeman

Topics to cover: Introduction to games programming using C++ and openFrameworks

# What is C++?

C++ is a programming language.

A programming language is a set of formal instructions used to communicate with a computer.

# What is openFrameworks?

"openFrameworks is an open source C++ toolkit designed to assist the creative process by providing a simple and intuitive framework for experimentation."

What does all that mean?

*quote from openframeworks.cc

# What is openFrameworks?

A software framework is like a toolbox. It has built in functionality that does a lot of the dirty work for you!

For example, oF provides simple functions to draw shapes on the screen, which would otherwise be quite complicated in plain old C++.

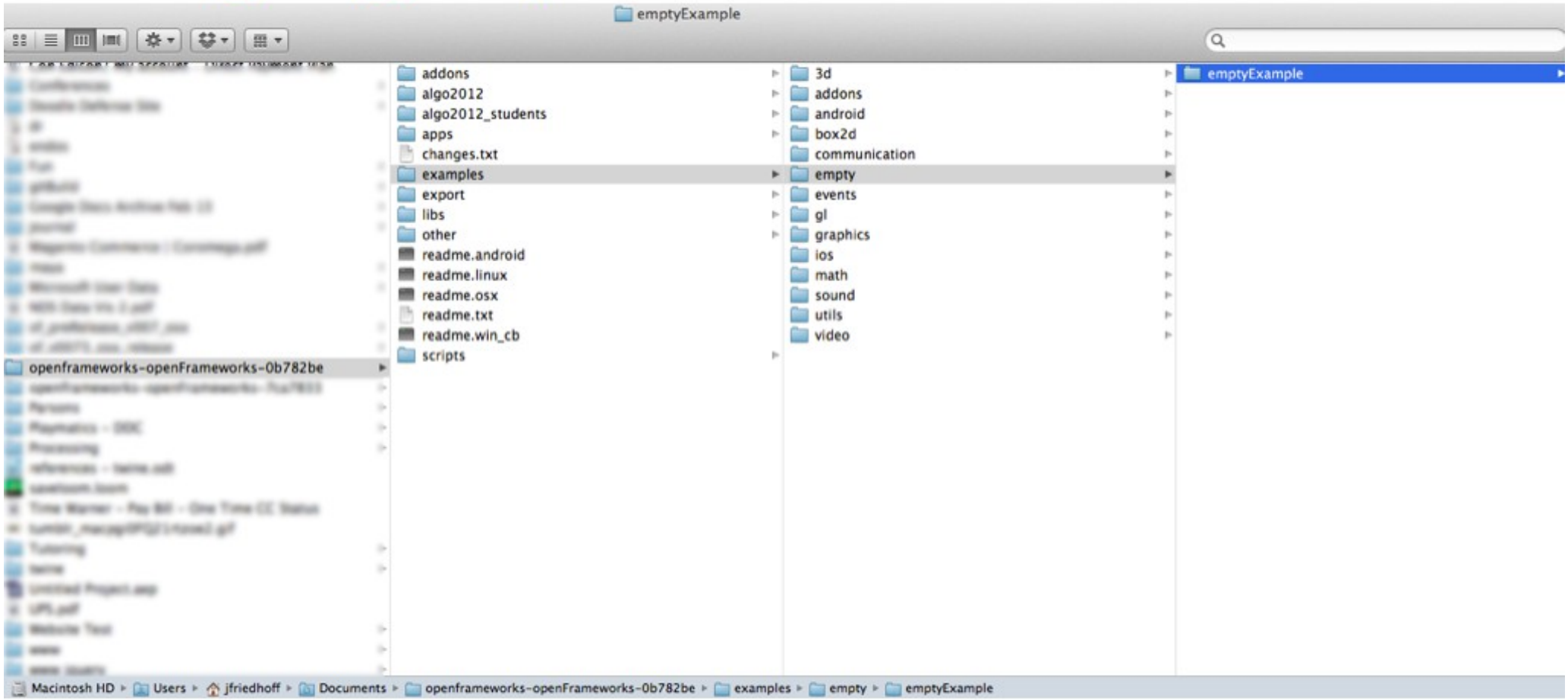
# What kinds of games can I make with these tools?

Ridiculous Fishing

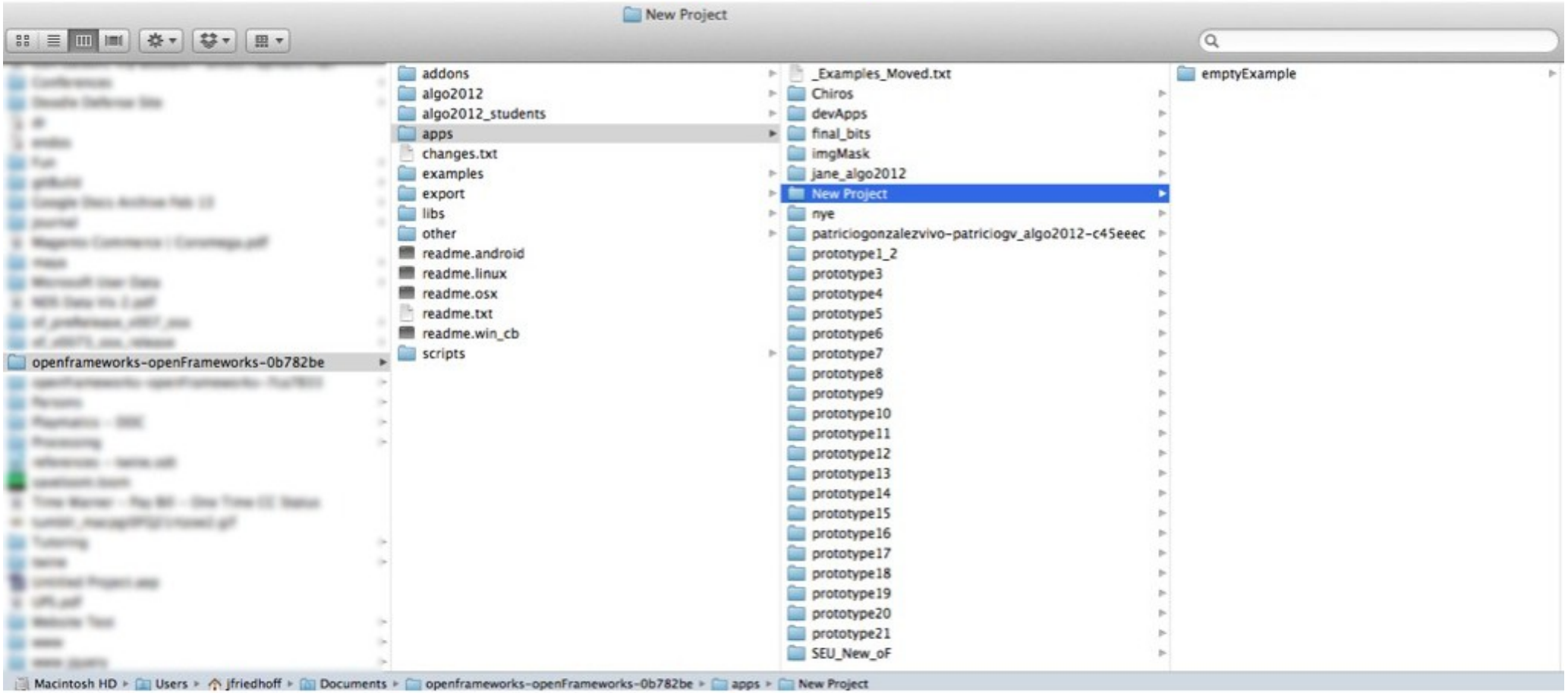
SpellTower

# Let's set up a new openFrameworks project!

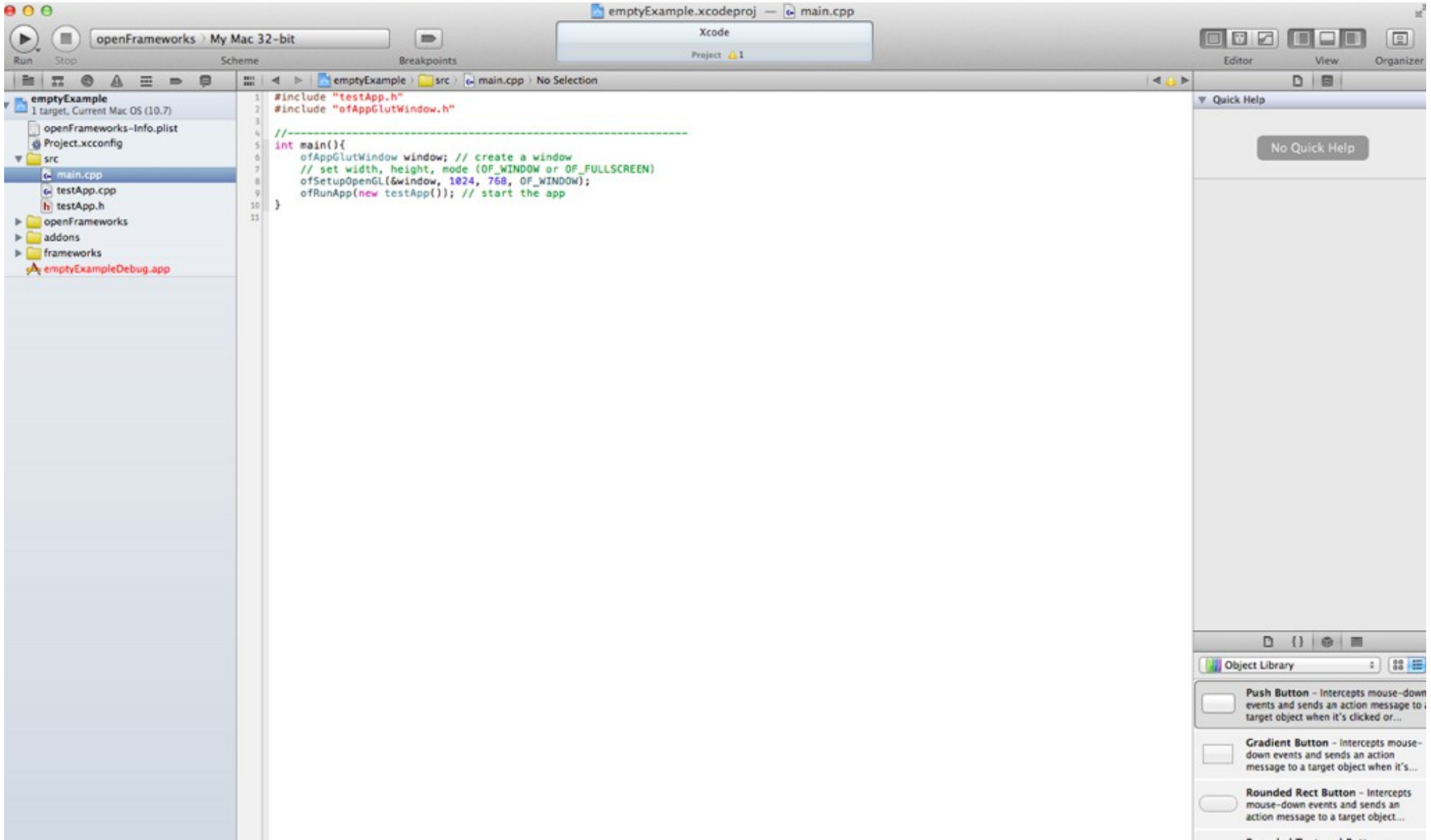
# Part 1: Copy emptyExample



# Part 2: Make a new folder in “apps” and paste it there



# Part 3: Open the project



# Smart tips!

You can and should take notes in your code.

```
// This is a one-line comment.

/*
    This comment can span multiple lines.
    Check me out, taking up all the space.
    Echo...
        echo...
            echo...
                echo...
                    echo...
                        echo...
 */
```

# Let's write our first program!

# Statement

Type the following under setup(){

```
cout >> “Hello world!”;
```

A statement is like a sentence—it's one line of code ending in a semicolon.

Fun fact: The smallest statement you can make in C++ is a single semicolon.

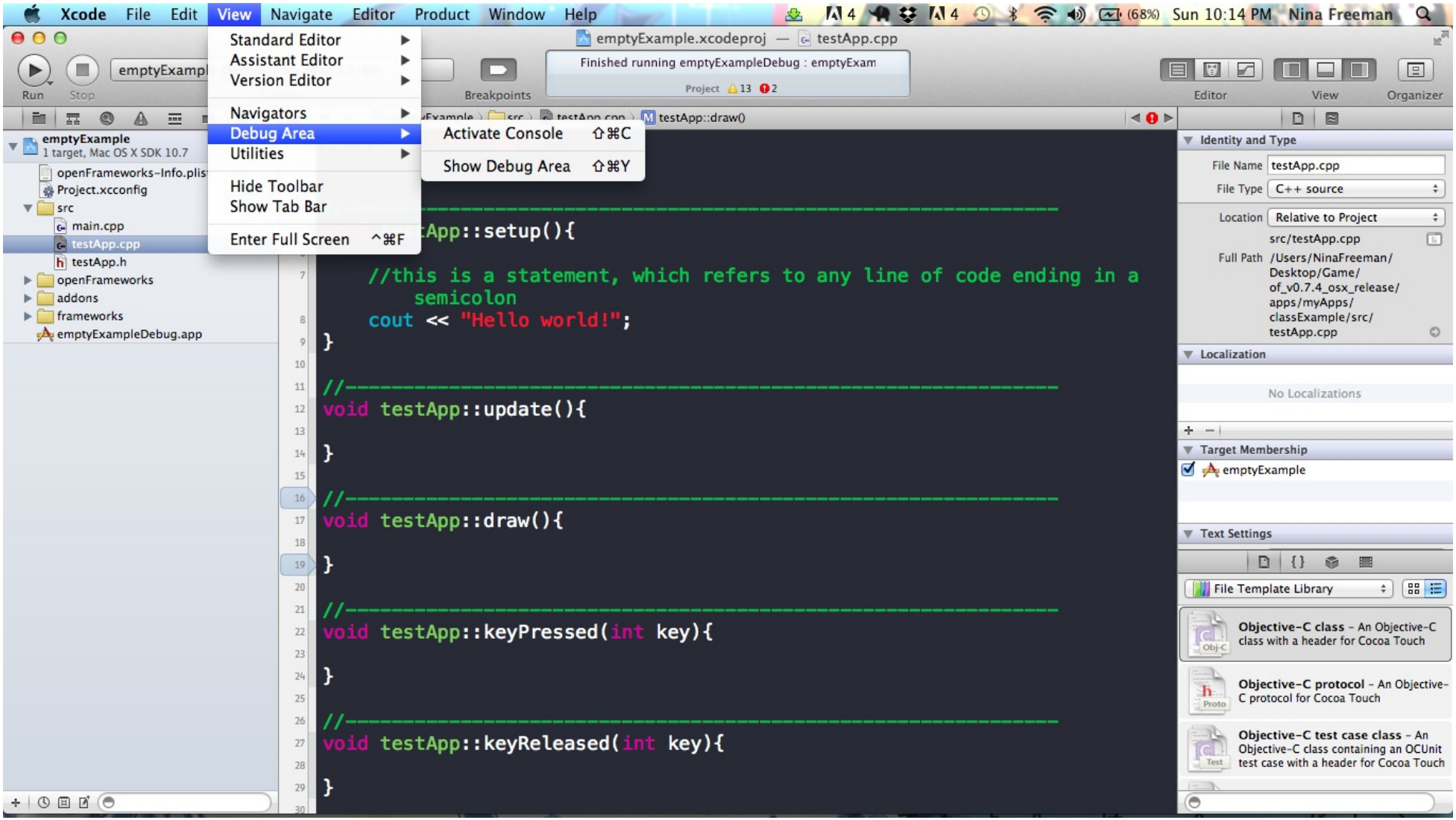
```
cout >> "Hello world!";
```

"cout >>" means "standard output stream"

This statement inserts a sequence of characters into the standard output stream, which is our console.

This statement is defined in the C++ standard library.

# Click Activate Console. This is where “Hello World!” will print out.



# To see what this code does, hit run in the upper right corner of Xcode.

Your program compiled (hopefully)!

When you hit run, Xcode compiles all that code you just wrote into an executable.

The executable is the program your computer actually runs.

Coding is like writing a book. You start with some drafts that get sent to a publisher...

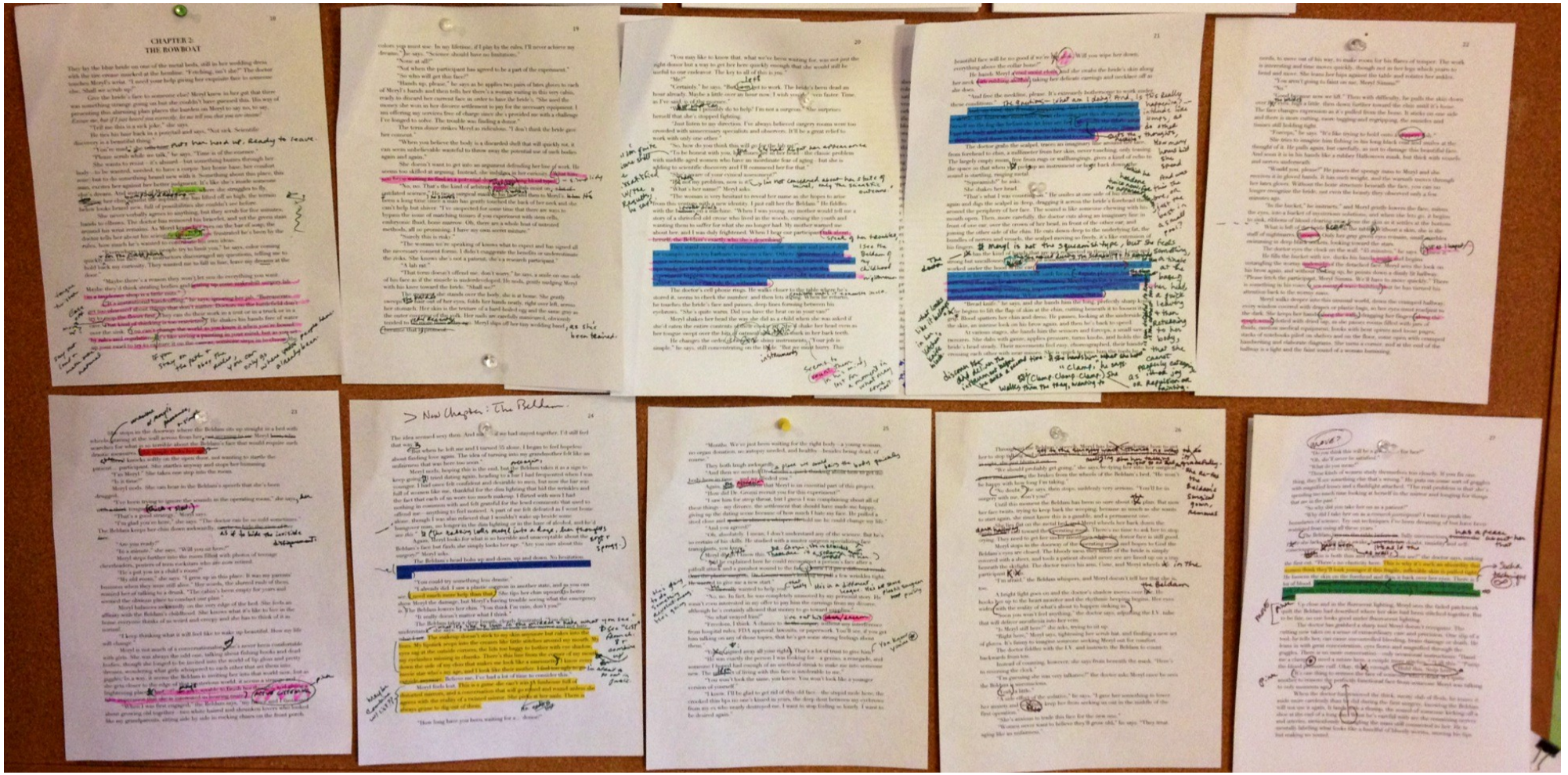


Photo found here

# Then the publisher (Xcode for us!) compiles it and out comes a bound book!

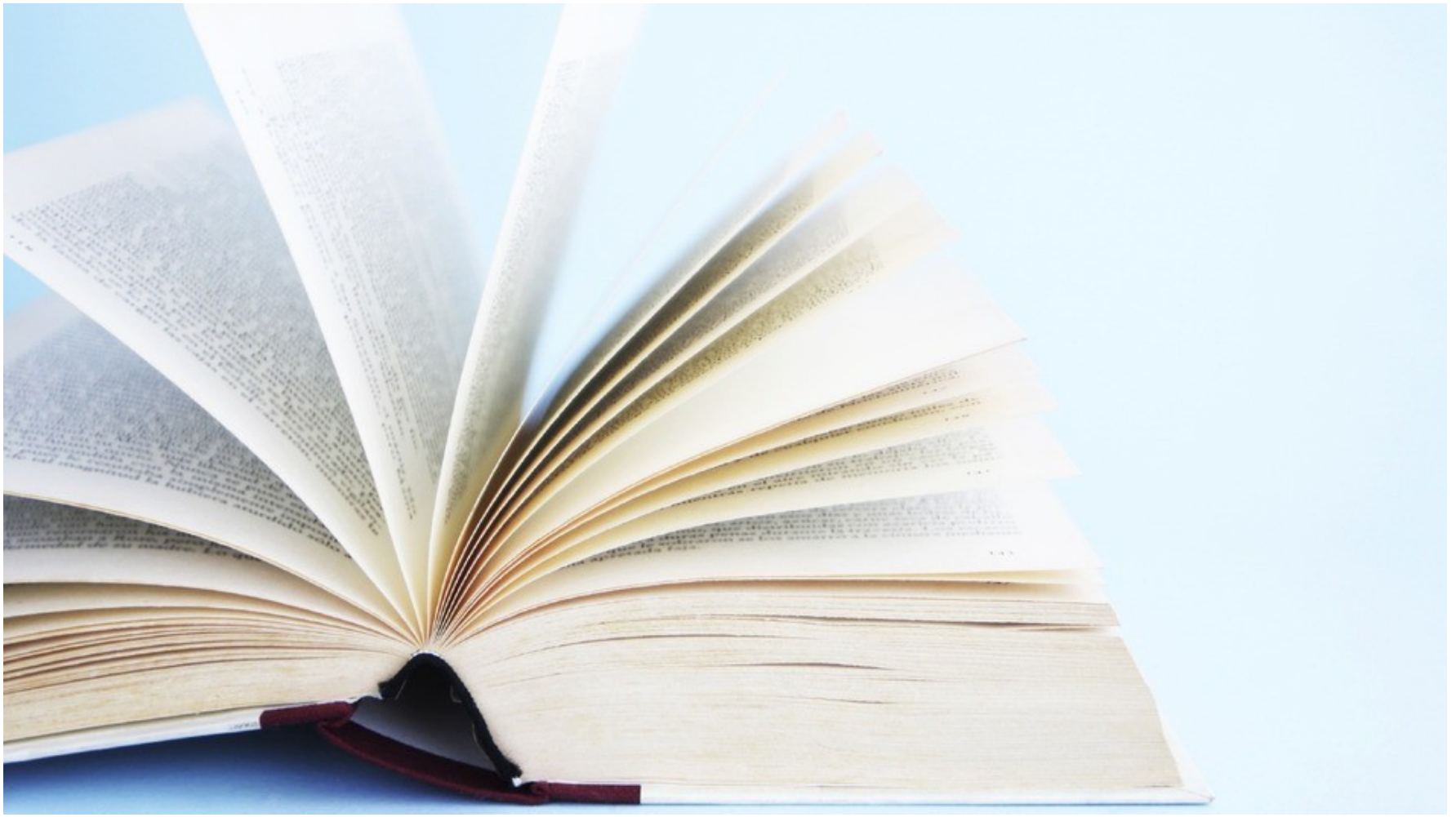


Photo found here

# What's what in openFrameworks? What does my code do where? What are all these setup, update and draw things?

```
void testApp::setup(){

}
```

```
void testApp::setup(){

}
```

Code written between these brackets executes ONCE at the very start of your program.

Good for setting variables like names or starting location.

```
void testApp::update(){

}
```

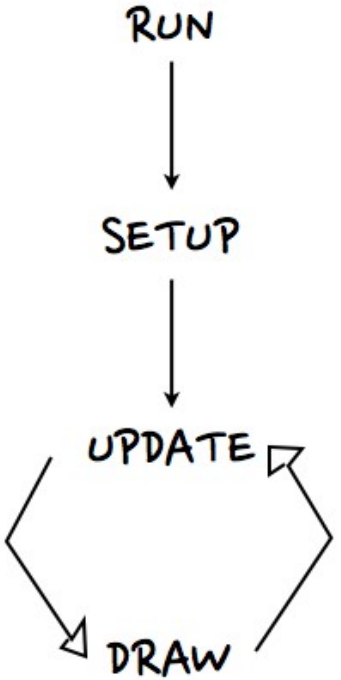
Code written between these brackets executes every other frame, switching off with Draw.

Do math and number crunching here!

```
void testApp::draw(){

}
```

Code written between these brackets executes every other frame, switching off with Update.

Draw things to the screen here!


RUN
SETUP
UPDATE
DRAW

oF also has listeners, which execute code when certain events happen: e.g. mouse clicks, drags or key presses.

```
//--------------------------------------------------------------
void testApp::keyPressed(int key){

}

//--------------------------------------------------------------
void testApp::keyReleased(int key){

}

//--------------------------------------------------------------
void testApp::mouseMoved(int x, int y){

}

//--------------------------------------------------------------
void testApp::mouseDragged(int x, int y, int button){

}

//--------------------------------------------------------------
void testApp::mousePressed(int x, int y, int button){

}

//--------------------------------------------------------------
void testApp::mouseReleased(int x, int y, int button){

}
```

Ok, so...

What about the code that goes into all those useful functions?

Next, we'll talk about variables and operators!

# Variables and Data Types

What if you want to say something like "Hello World!" in multiple different parts of your program?

You would want to tell the program to remember that sentence so you could use it again later, right?

Create a variable to hold it!

# Variables and Data Types

Variables are declared at the very top of your program, outside of Setup{}

You're telling the program to set aside some memory for a string called greeting, but it's not holding any data yet...

```
string greeting;
```

Data Type  Identifier

# Variables and Data Types

Variables should be initialized in Setup{}

Now, you're saving your data to the variable, in this case, called greeting.

```
greeting = "I'm a variable!";
```

Assignment Operator

The data you want to save.
Must match data type!

# Variables and Data Types

Variable names need to start with an _ (underscore) or a letter.

Variable names can *only* consist of letters, numbers or underscores.

You cannot have any spaces in a variable name.

Variable Name

```
greeting = “I'm a variable!”;
```

# Variable Data Types

Type: Integer

```
int integerVariable = 10;
```

An int variable can hold any negative or positive whole number value.

20, 2000, -3, 4.....

# Variable Data Types

Type: Character

```
char charVariable = 'a';
```

A char variable can hold any character. Uses 'single quotes', not “double”.

A, b, c, d, E, F.....

# Variable Data Types

Type: String

```
string stringVariable = "hi";
```

A string variable can hold characters within quotation marks. Sentences, words, etc.

"My Name", "Dog", "yay!".....

# Variable Data Types

## Type: Boolean

```
bool boolVariable = true;
```

A bool variable can TRUE or FALSE.

# Variable Data Types

## Type: Floating Point

```
float floatVariable = 1.2;
```

A float variable can hold a number with up to 7 digits after the decimal.

4.5, -3.444, 777.7, 8.9.....

# Variable Data Types

Type: Double

```
double doubleVariable = 3.45;
```

A double variable can hold a number with up to 16 numbers after the decimal point. More precise than floats.

4.5, -3.444, 777.7, 8.9.....

# Const Variables

You can also make “constant” variables. Any variable labelled as a constant cannot be altered after it is initialized.

```
const int iNeverChange = 1;
```



# Variables and Data Types

Where do you declare your variables?

Where do you initialize your variables?

What are some examples of data types?

# Declare and initialize 3 variables:

A variable of type int called circleX.
Initialize to x in mouseDragged.

A variable of type int called circleY.
Initialize to y in mouseDragged.

A variable of type int called circleRadius.
Initialize to anything between 20-100 in setup.

When you're done we will draw a circle!

# OpenFrameworks function For Drawing Circles

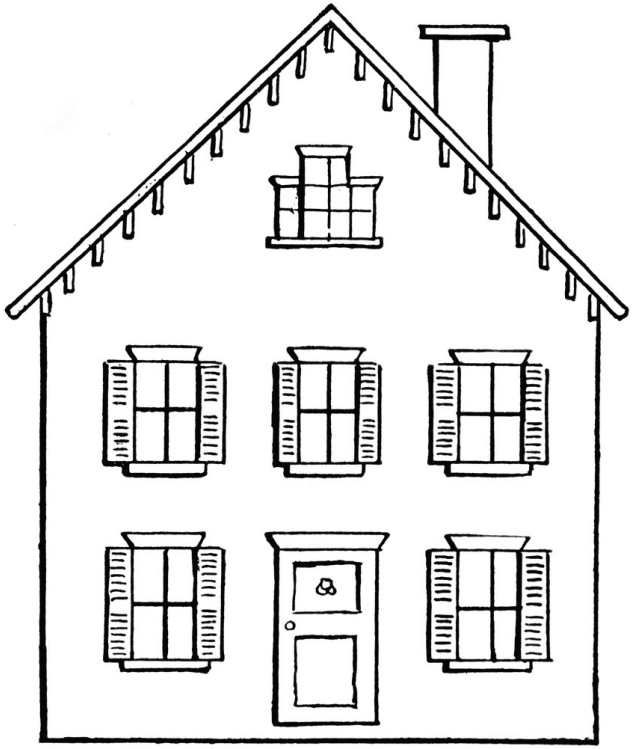
```
void testApp::draw(){

    ofCircle(circleX, circleY, circleRadius);

}
```

# Back to variables!

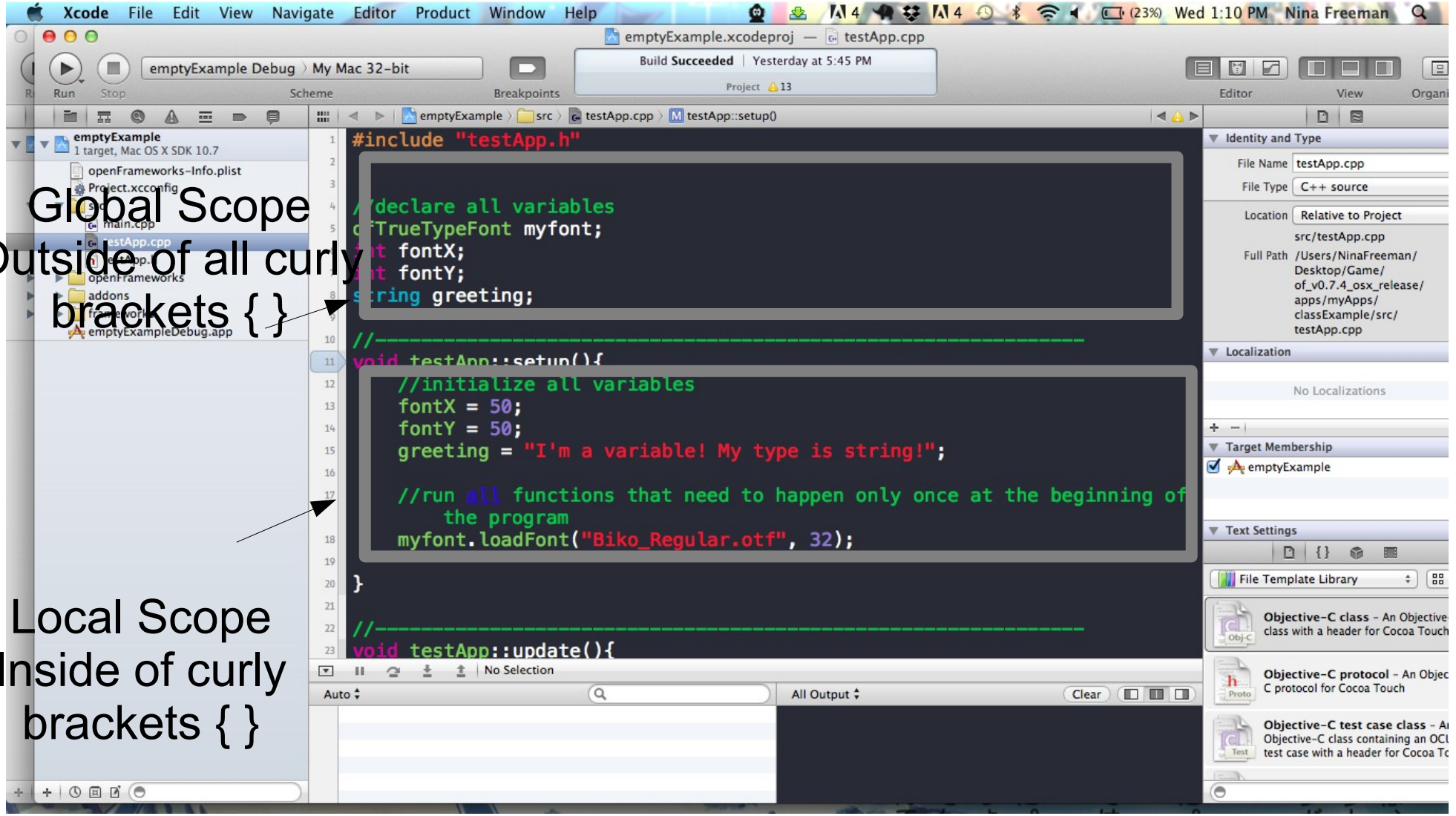
# Wait, what's a variable again?

# A variable holds onto data for you to use later!

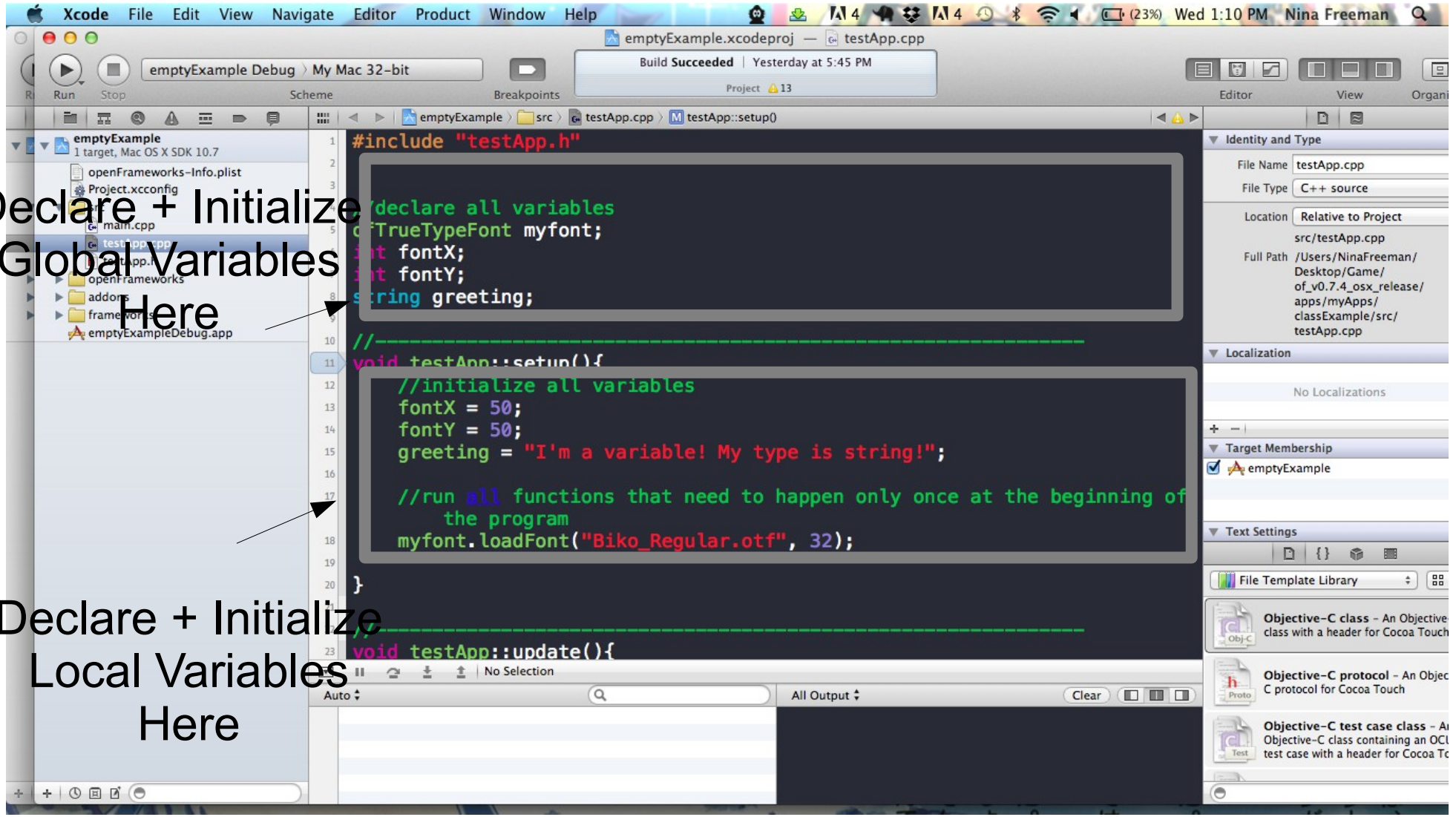
Variables can be either local or global. This determines which part of your program can see and use the variable.



# Why? Scopes!


Global Scope
Outside of all curly brackets { }
Local Scope
Inside of curly brackets { }
#include "testApp.h"
//declare all variables
ofTrueTypeFont myfont;
int fontX;
int fontY;
string greeting;
//--------------------------------------------------------------
void testApp::setup(){
//initialize all variables
fontX = 50;
fontY = 50;
greeting = "I'm a variable! My type is string!";
//run all functions that need to happen only once at the beginning of the program
myfont.loadFont("Biko_Regular.otf", 32);
}
//--------------------------------------------------------------
void testApp::update(){

# Global vs. Local Variables



What do you think would happen if:

I declared AND initialized variable X = 1 in draw.

Then, tried to assign 2 to that same variable (e.g. X = 2) in setup?

What do you think would happen if:

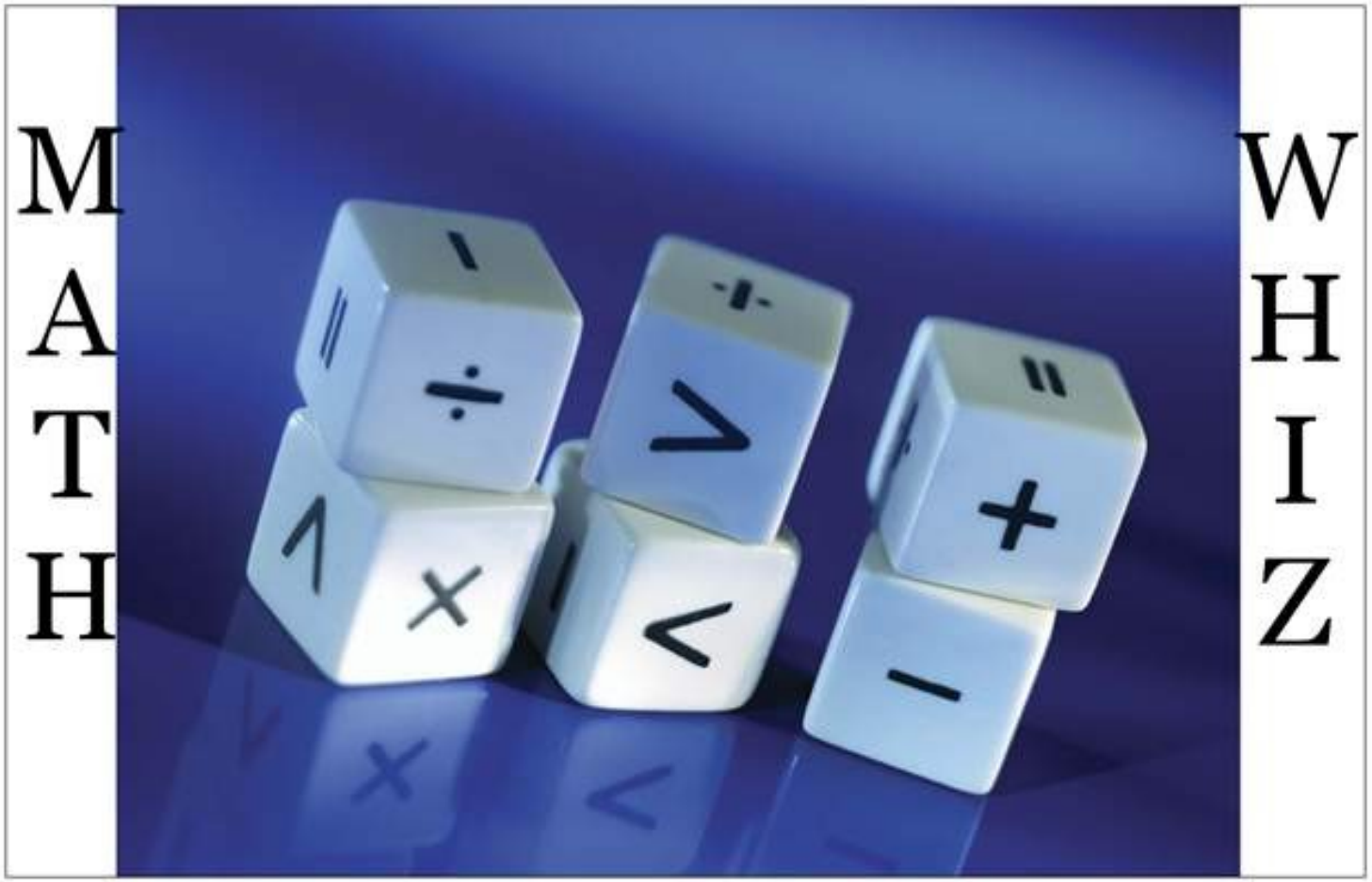
I declared AND initialized variable X = 1 in draw.

Then, tried to assign 2 to that same variable (e.g. X = 2) in setup?

Error! X is out of scope! X was only declared in draw, not setup!

Where do you declare local variables and what parts of your program can see them?

What about global variables?


M
A
T
H
W
H
I
Z

# Arithmetic Operators

+ addition
- subtraction
* multiplication
/ division
% modulo

# Modulo

Modulo returns the remainder of a division of two values. For example:

```
a = 11 % 3;
```

The variable a will contain the value 2, since 2 is the remainder of 11 divided by 3.

# Coming up next week!

# Relational and Equality Operators

== Equal to
!= Not equal to
> Greater than
< Less than
>= Greater than or equal to
<= Less than or equal to

# Logical Operators

! Not
&& And
|| Or