# Class Six

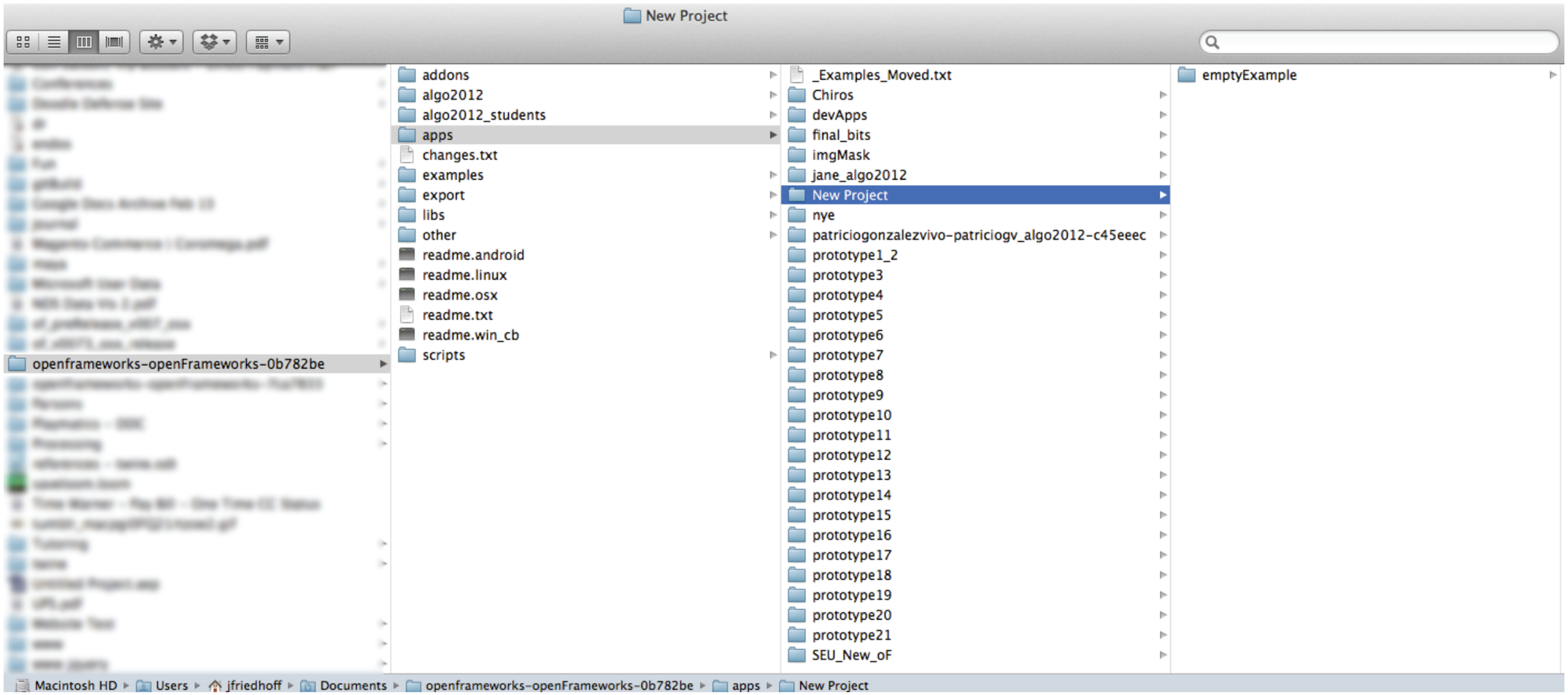
openFrameworks!

- A software frameworks, by which we mean: a software infrastructure that provides low-level functionality

- Will allow you to start playing with assets (images, sounds) as well as interactivity (mouse movement, keyboard interaction)

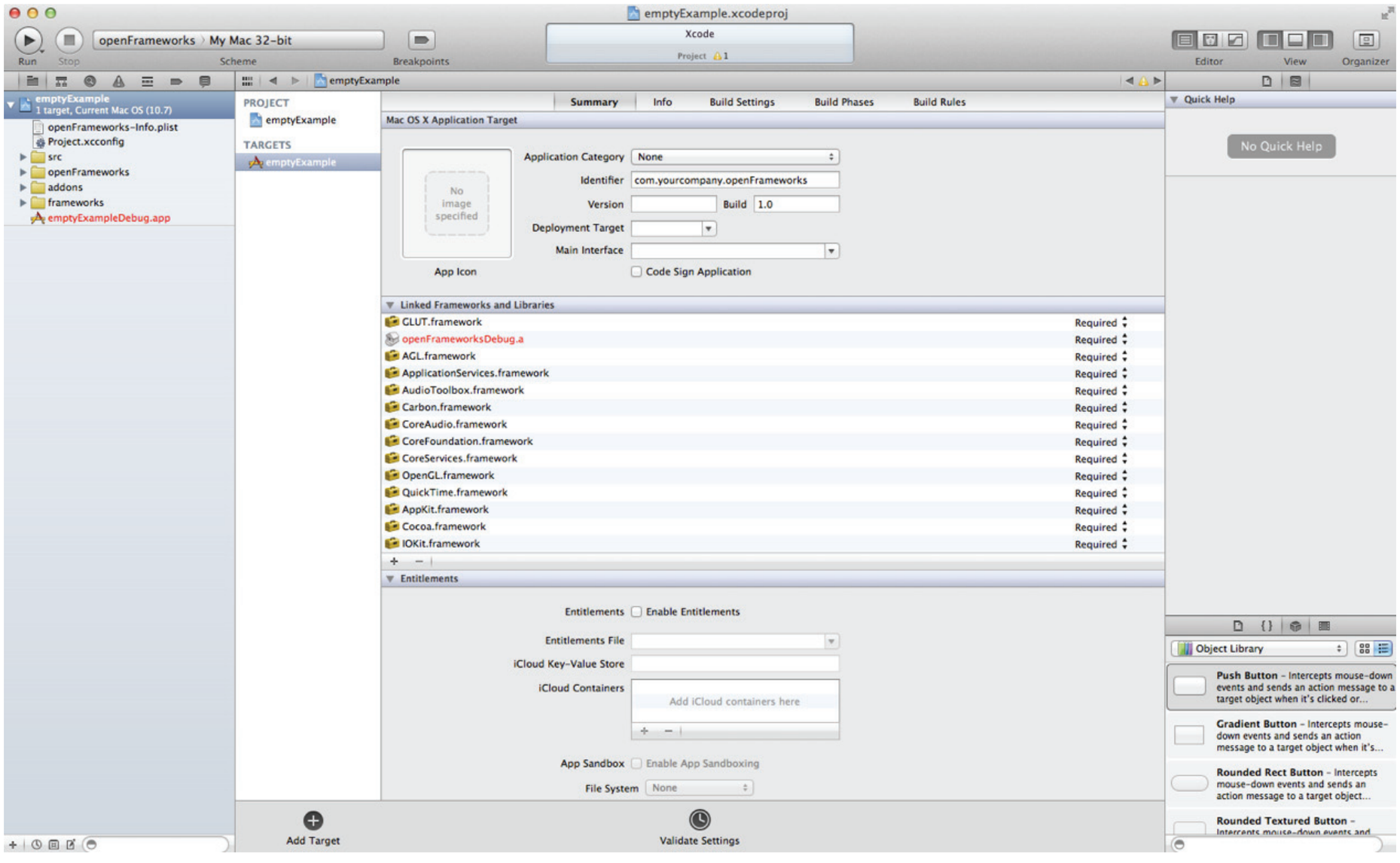
Download it from openframeworks.cc. And check out the reference!

- Making a new project: navigate to the emptyExample folder and copy it
- Make a folder within the app folder (not required, but helps keep things neat
- Paste emptyExample into to your projects folder
- Your project folder has to be exactly three levels down from the base oF folder!



A basic oF app starts with three files in src: main.cpp, testApp.h, and testApp.cpp.

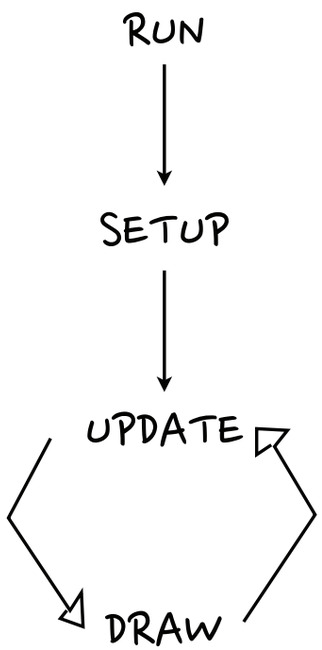
- main.cpp: largely handles launching the app; no longer our main place to work in
- testApp.h: where you declare global variables and function prototypes
- testApp.cpp: your new main!



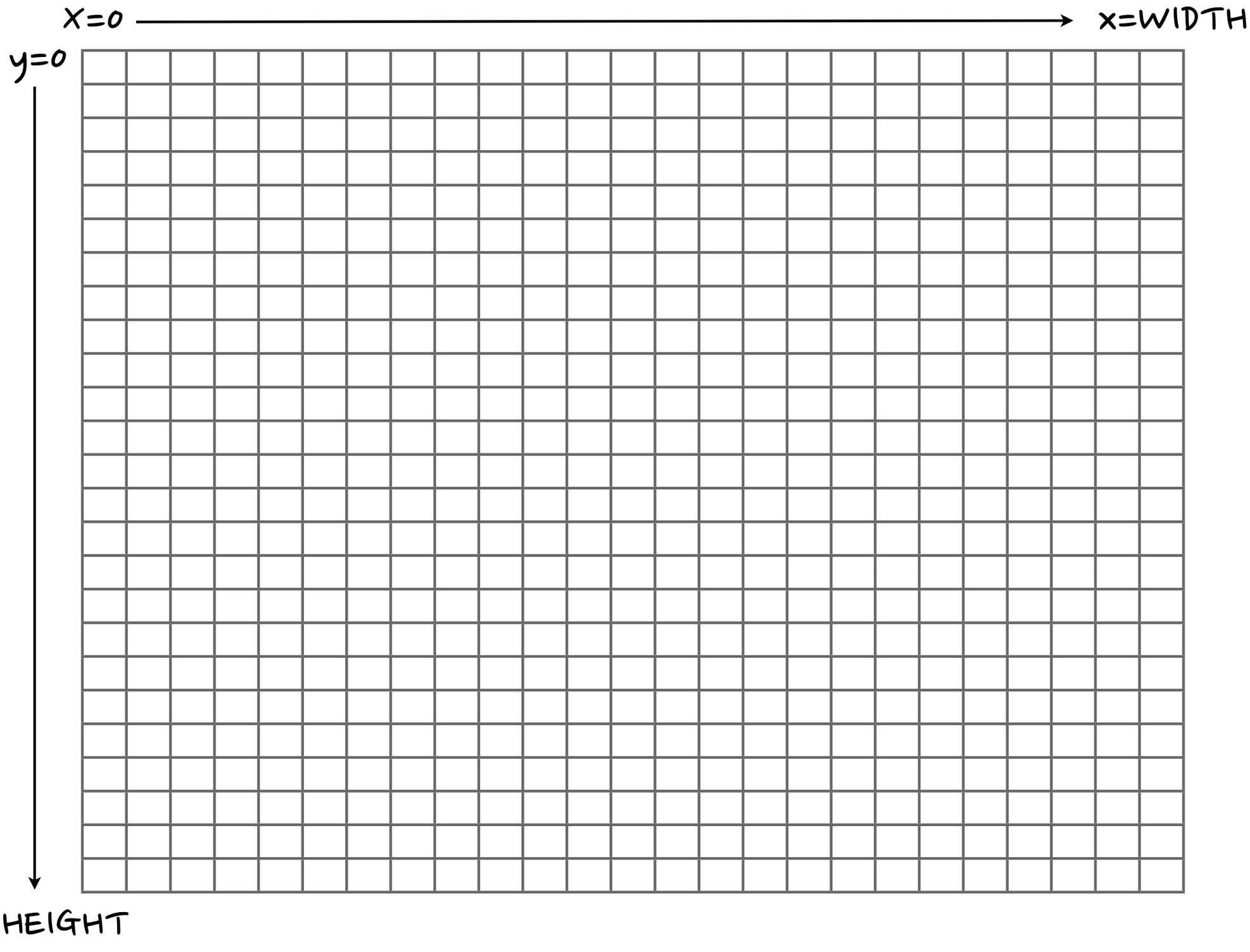
openFrameworks runs on a frame system. There are typically 60 frames per second.

There are three main functions/loops in openFrameworks.

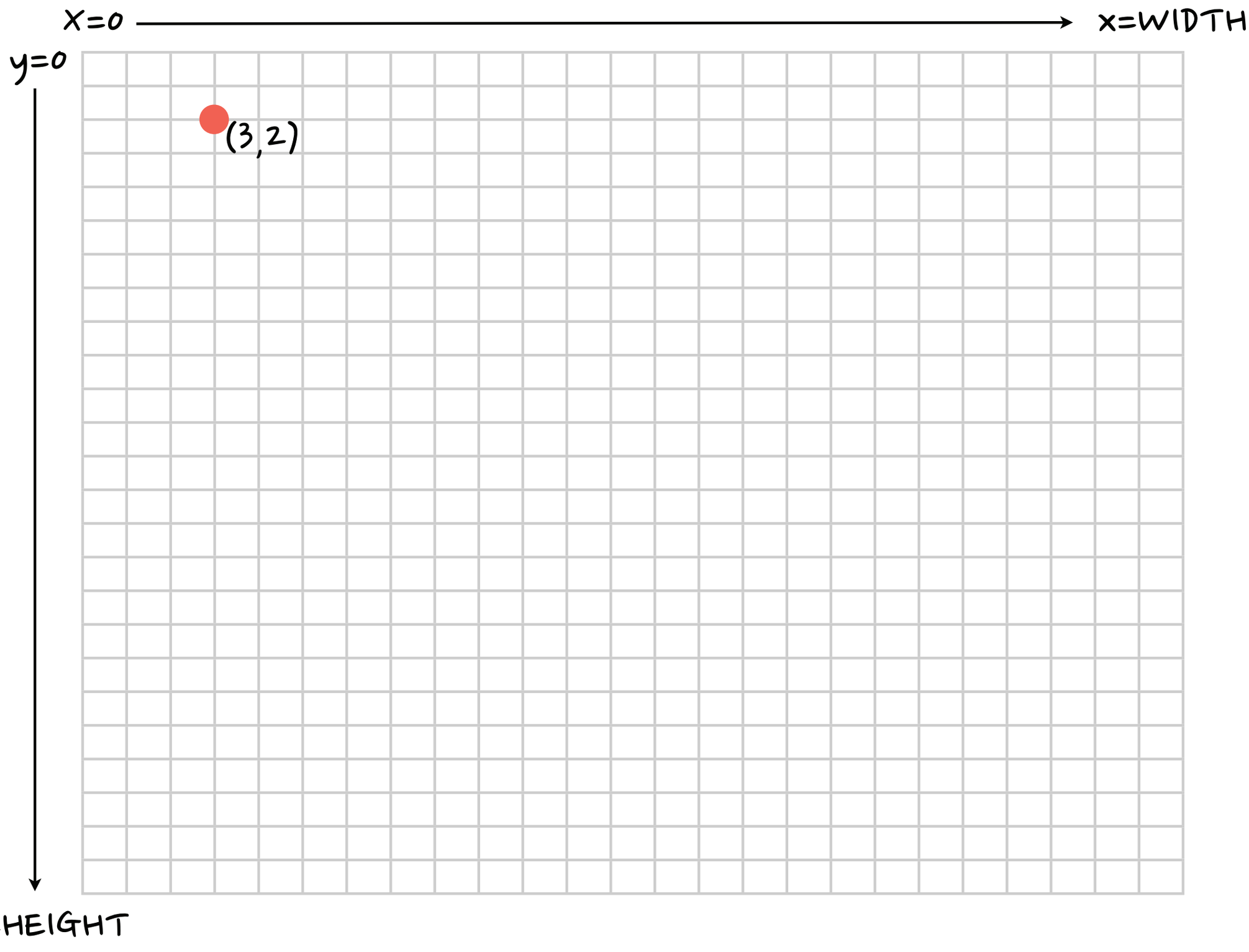
- void testApp::setup() runs exactly once, right when the program starts. Useful to give starting values to variables, e.g. start the player's health at 100.
- void testApp::update() runs once per frame. This is where you should put your number crunching, e.g. if the player is poisoned, their health should go down by 1 per frame.
- void testApp::draw() also runs once per frame, after update loop. This is where you should put visual stuff, e.g. drawing the health value.



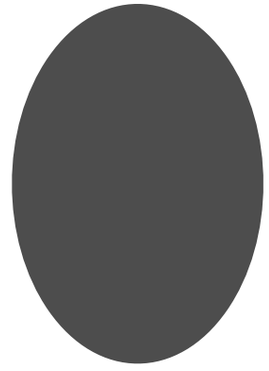
In openFrameworks, x-values get bigger to the right, and y-values get bigger going down.



In openFrameworks, x-values get bigger to the right, and y-values get bigger going down.

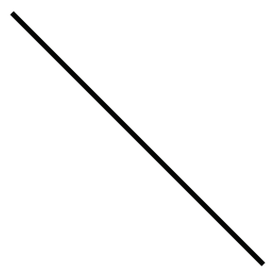


ofEllipse(x, y, width, height);



ofRect(x, y, width, height);

ofLine(x1, y1, x2, y2);



(and several more, plus diy shapes!
check the reference for more info.)

```
//--------------------------------------------------------------
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){

}

//--------------------------------------------------------------
void testApp::windowResized(int w, int h){

}

//--------------------------------------------------------------
void testApp::gotMessage(ofMessage msg){

}

//--------------------------------------------------------------
void testApp::dragEvent(ofDragInfo dragInfo){

}
```

(these functions fire automatically when their event happens)

```
//--------------------------------------------------------------
void testApp::keyPressed(int key){
    if (key == 'a') {
        // code goes here
    }

    if (key == 97) { // ascii for a
        // code goes here
    }

    if (key == OF_KEY_RIGHT) {
        // code goes here
    } else if (key == OF_KEY_LEFT) {
        // code goes here
    } else if (key == OF_KEY_UP) {
        // code goes here
    } else if (key == OF_KEY_DOWN) {
        // code goes here
    }

}
```

(you have several options for testing which key has been pressed)

ofGetWidth() ⟶ returns the app width as an int

ofGetHeight() ⟶ returns the app height as an int

mouseX ⟶ returns the mouse's x position as an int

mouseY ⟶ returns the mouse's y position as an int

ofSetColor(int r, int g, int b) → sets color

ofSetColor(int r, int g, int b, int a) → sets color with alpha

ofFill() → next shapes will have fill

ofNoFill() → next shapes won't have fill

ofEnableAlphaBlending() → enables transparency

ofDisableAlphaBlending() → disables transparency