**Good Coding Habits**

**Part 1: Meaningful Names**

**Foreword**

Hello readers.

I've never actually written a magazine article before and I usually have a bit of an odd writing style, so this will be interesting to say the least.

(I was also going to say I'll try not to write a text-wall, but in a way that's what magazine articles are supposed to be, right?)

While this article is aimed mainly at beginners, I'd recommend reading it even if you aren't a beginner. Even the smartest of people can write code that's hard to follow, and I doubt there is someone out there who wouldn't benefit from being reminded about some good coding habits.

This article is going to be all about 'good coding habits', which means it's going to be full of tips that should make your code easier to read, easier to work with and easier to maintain.

Writing good code does involve some effort, but others will thank you for it, and hopefully you'll also see the benefits.

As an added bonus, most of these techniques are transferable skills that will help you when programming things other than the Arduboy and in other programming languages.

Anyway, here goes my first ever article.

**Introduction**

This first part is about using meaningful names. I've noticed that beginners (and sometimes even experienced programmers) have a habit of not giving their variables very good names.

Sometimes people get lazy and can't be bothered to type out long words so they just use shortened versions. Sometimes people just don't know what to call their variable. In fairness, it's widely accepted by programmers that naming things can be really hard, but it is important for writing readable code so it deserves care and attention.

So I'm going to try my best to offer some advice on what makes a good name and how you can become a better namer.

**Tip 1: Use Nouns And Verbs**

To start with, there's one very easy way you can improve the readability of your code.

M*ake sure all your variables are nouns (or 'thing words') and all your functions are verbs (or 'doing words')*.

In programming, a variable is by its very nature a 'thing'.

It's some sort of object, a block of data, maybe a number or a representation of a sprite.

Variables don't 'do' anything, they just sit there. For a variable to 'do' anything is has to be altered by an operator (e.g. =, +, - etc.) or by a function.

For example, say I want to add 5 to an integer:

variable = variable + 5;

The variable doesn't do the adding, the variable is just the place where the number is stored. Instead, the addition operator + is used to take the variable's value and add 5 to it to create a new value and then the assignment operator = takes that value and puts that value back into the variable.

So naturally, since variables are in fact 'things' it makes sense that variables should be named with 'thing' words, e.g. player, enemy, entity, gameState.

Functions on the other hand, should be named with 'doing' words, e.g. update, draw, addHealth, checkForWin

**Tip 2: Use Descriptive Names**

For beginners, it's especially common to think about what a variable is rather than thinking about what it's used for. I've seen quite a few beginners working on their first attempt at doing some maths in code and decide to call their variable number or number1. That's fine for a first attempt, but you shouldn't be doing that in production code.

*When picking a name for something, you should pick a name that reflects what the purpose of that thing is.*

For example, call your Arduboy2 variable arduboy, because that's what it represents.

If you've got a variable to store the player's name, call it playerName.

If you have an object representing the player, don't call it myHero, call it player.

If you're calculating the distance between a player and an enemy then distance is probably a good name, although distanceToEnemy would be much more descriptive even if it's a bit more typing.

If you've got a boolean variable, you should try to name it with words like 'is' or 'can', for example isOnGround or canFly. Likewise you shouldn't use 'is' or 'can' for things that aren't boolean variables, for example if you have a counter that you use to do something special when it reaches zero, that shouldn't be a canDoSomething variable because that isn't an accurate description of what it does, a better name would be something like framesUntilSomething – it's a counter, not a condition.

Another example, although this one is a bit more of a grey area: if you're writing an asteroids game and you're representing the ship's direction as an angle, don't call it shipDirection, call it shipAngle so it's clearer that it's just an angle and not some other way of representing direction (e.g. a vector).

Using descriptive names gets more and more important as you start to write more advanced code, and sometimes it gets more difficult too.

There are some incredibly complicated topics in programming so it really pays to be as clear as possible.

Remember, at some point you will stop working on a project and start doing other things. If you decide to come back to that project later and find you've forgotten everything and you haven't used very good names then you're going to be struggling to understand it. There's nothing worse than reading some code, wanting to scream “what moron wrote this? I can't understand any of it” and then suddenly realising “oh wait, it was me... oops”, and I'll freely admit I've been there, as I suspect many others have.

**Tip 3: Avoid Short Names**

Historically programming used short names because back 'ye olden days' computers didn't have much memory for storing source code and there weren't any fancy newfangled auto-complete systems. Also most programmers were mathematicians who were doing serious business 'computer science', so they had the mathematicians' habit of single-letter variables.

One of the biggest reasons I never got on very well with algebra (or mathematics in general) is that mathematicians love to give their variables single-letter names and it's not very helpful for understanding an equation. Take a random mathematical equation using single letters for variables and show it to a non-mathematician and they won't have the slightest clue what it's used for. The same is true for programming.

*If most of your variables have one-letter names then your code will generally look confusing*.

For example, Einstein's famous equation could be written in code as

float e = pow(m \* c, 2);

but it could be written much more descriptively as

float energy = (mass \* lightSpeed) \* (mass \* lightSpeed);

Before the rewrite, most people would recognise the equation but not have a clue what it means, but after the rewrite it's more obvious that it's calculating the energy of something by multiplying the mass of something and the speed of light, then multiplying that value by itself. Programming need not be rocket science! (Or in this case theoretical physics.)

There are a few exceptions to this rule. For example, using x and y as coordinates and using i and j as loop variables are well established. Also in some cases where mathematical constants are involved (like Euler's number e or the speed of light c) the recognised mathematical letter is acceptable.

Using contractions of larger words is a bit of a grey area.

Some people are fine with contracting sprite to spr, width to w and height to h.

Personally I am against doing this because I think using the full names makes code easier to read because fully fledged words are easier to recognise.

(If you shorten arduboy to ab, know that I am frowning at you right now, and also ducking under my desk to dodge the internet projectiles you're probably throwing at me.)

Importantly, remember that modern text editors and IDEs have auto-completion of words, so being too lazy to type long names is a poor excuse. Also, using smaller variable or function names doesn't actually save space – it might make your source code smaller but it won't affect the compiled code at all.

**Tip 4: Keep Naming Consistent**

Lastly, when naming things it's important to keep your naming choices consistent. For example, if you were writing some functions involving USB communication, it's important to be consistent with the capitalisation of USB. If you wrote a function called openUSB, it would be confusing to then have another function called closeUsb because you'd be using two different approaches to capitalisation.

Using acronyms in code is another one of those awkward grey areas. Sometimes people prefer putting the letters all in capitals, such as openUSB and closeUSB, and sometimes people prefer treating an acronym as its own word, like in openUsb and closeUsb. Neither is right or wrong, but it's important that when you pick a naming pattern, you stick to it.

Similarly, it is important to keep the order of naming consistent. By which I mean don't mix openUSB and usbClose, stick to using either openUSB and closeUSB or usbOpen and usbClose. When your names follow a distinct pattern it makes it much harder to make mistakes and much easier to remember names.

**Summary**

Naming might be hard at times, but it's worth the effort to get it right. Whether you code alone or in a team, giving your variables, functions and classes good meaningful names can be very helpful.

Even if you do code alone, you might take a break from your project and forget all about it. Then six weeks down the line you rediscover your project and want to work on it again – that's going to be a lot easier if everything's clearly named. The less time you have to spend understanding your code, the more time you can spend writing games.

Then there's the matter of asking for help. Everyone needs help with their code sometimes, and when that time comes you wouldn't want the person helping you to have a hard time trying to understand your code. If your code is easy to read and easy to understand, that person helping you will be able to sort your problem a lot quicker than they would if your code was full of confusing names.

I think I'd like to end this with a relevant and somewhat profound quote from a well known programming book (that I've heard of but never read):

*“Names are deeply meaningful to your brain, and misleading names add chaos to your code.” -* Andrew Hunt, 'The Pragmatic Programmer'.