Code Style Document

Coleman Cook, Jack Koster, Tayin Wallace, Jonathan Mercado

Identifiers (variables, class names, methods):

Variable names and methods should use camelCase formatting, with the first word being lowercase and subsequent words capitalized. For example, "frameIndex" or "spriteWidth" (these may not be actual variable names in our code). Like these examples, variable names and method names should be descriptive and indicate clearly what the variable represents, or method does. Avoid using single-letter variable names unless they are common mathematical variables (e.g. i, j, k for loop iterators). Use const whenever possible to indicate that a variable's value should not be changed after initialization. When initializing variables and where possible, use initializer lists instead of assignment in the constructor. Finally, try to limit the scope of variables to the smallest necessary scope to minimize potential naming conflicts and to improve code readability. Class names should be descriptive and use CamelCase with the first word capitalized.

Indenting:

To ensure consistent and readable code, we will follow the C++ standard indenting practices in our sprite editor app. This means using 4 spaces for each level of indentation. We will use whitespace to improve code readability, such as adding a space after a comma to separate function arguments or around binary operators to visually separate them from their operands. Additionally, we will use blank lines to separate logical sections of code, such as between functions or before and after loops, or conditional statements. We will use curly braces for all code sections, even if they are just one line. Following these standards will make our code more readable and easier to understand between teammates.

Comments:

Good code is usually self-commenting, and simple methods usually are readable enough without comments. However, if you think your code needs a comment, it probably does. In our C++ sprite editor app, we will use comments to explain the purpose and functionality of classes, functions, as well as to document any non-obvious or complex code. Our comments will supplement, not replace, the readability of our code.