CS 3505 A8 Code Style Document

Contributors:

Andrew Wilhelm, Allison Walker, AJ Kennedy, Brett Baxter, David Cosby, Mason Sansom

1. Naming Conventions

1. **Classes, Functions, Methods & Structs:** Use camelCase to maintain consistency and readability.
2. **Variables & Member Variables**: Use camelCase for variables to create a uniform style.
3. **Enums:** Apply descriptive names and capitalize each word for clarity.
4. **Straightforward and Meaningful Variable Names:** Choose names that convey the purpose of the variable, allowing a streamlined code understanding.

2. Formatting

1. **Spacing Around Operators and Commas:** Introduce spacing to promote code readability.
2. **Curly Braces:** Place curly braces on a new line, providing a clean and consistent structure.
3. **Indentation:** Use the auto-indentation tool consistently, which displays 1 tab or 4 spaces for all files.
4. **Logical Organization:** Group code logically based on functionality, thus providing a coherent and maintainable codebase.
5. **Class Member Order:** Define class members/write code in the same order as they appear in header files.
6. **Include Guard Formatting:** For formatting the include guards, organize and display with system headers first, then project-specific headers, and then external library headers.

3. Comments

1. **Formal Documentation:** Use multi-line `/**/` and `@brief` in files to provide formal documentation.
2. **Informal Comments:** Utilize `///` for function and class comments in source files; use `//` for single-line comments to explain functionality.
3. **Avoiding Redundant Comments:** Avoid commenting on self-explanatory code to maintain code cleanliness and reduce redundancy.

4. General Specifications/Guidelines

1. **Meaningful Identifiers:** Ensure function, variable, and parameter names are meaningful and reflect their purpose/functionality.
2. **Separation of Model-View Functionality:** Clearly distinguish between model and view functionality to create modularity.
3. **QT Convention Adherence:** Follow current QT conventions for naming signals and slots to align with current version established practices.
4. **Redundancy Elimination:** Simplify code and remove any redundant elements to enhance efficiency.
5. **Consistency Enforcement:** Maintain consistency throughout all files to facilitate a consistent coding style.
6. **Error-Free Compilation:** Address and resolve any compiler errors or warnings.
7. **Optimized #Includes:** Only include relevant #includes, and group them logically to streamline code organization.
8. **Spell Checking:** Perform spell checks on variable names and documentation to check accuracy and coherency.