Dominik Jedruszczak
Coding Standard

1. Comments.
    a. General.
        i. Both C-style and C++-style comments are allowed.
    b. C-style.
        i. Comments (/* This is a C-style comment. */) must have a space between the first star and the first letter of the comment, and the period and the last star.
        ii. Comments must begin with a capital letter and end with a period.
    c. C++-style.
        i. Comments (// This is a C++-style comment.) must have a space between the second backslash and the first letter of the comment.
        ii. Comments must begin with a capital letter and end with either a period or suspension point.
            1. unsigned int numApples; // The number of apples.
            2. for (unsigned int i = 0; i < numApples; i++); // For each apple…
    d. Files.
        i. All files must begin with a C-style comment in the following format:
        /*(newline)
        (tab)<author's name>(newline)
        (newline)
        (tab)<file-name>(newline)
        (tab)<description>(newline)
        */
        ex.
        /*
                Dominik Jedruszczak

                camera.hpp
                Declaration of the camera class and associated definitions.
        */
    e. Variables.
        i. All variables with the exception of iterators declared in the initialization of for loops are required to have a C++-style comment describing the variable.
    f. If-statements and loops.
        i. If-statements and loops are required to have a c++-style comment ending in a suspension point describing the condition of the statement or the meaning of the iteration.
            1. if (numApples > WOVEN_BASKET_CAPACITY) // If the number of apples is greater than the capacity of the woven basket…

2. for (unsigned int i = 0; i < numApples; i++); // For each apple…

g. Classes.
   i. Must have a comment before the declaration with a short description of the class.

2. Naming.
   a. Files.
      i. General.
         1. Must be in lower camel case.
      ii. Source files must end in ".cpp".
      iii. Header files must end in ".hpp".
   b. Variables.
      i. Must be written in lower camel case.
      ii. Must avoid unnecessary abbreviations. Variable names with abbreviations must include the un-abbreviated name in the descriptive comment.
   c. Classes.
      i. Must be written in upper camel case.
      ii. Must avoid unnecessary abbreviations. Variable names with abbreviations must include the un-abbreviated name in the descriptive comment.
   d. Definitions.
      i. Must be written in upper snake case.
3. Indentation.
   a. Classes
      i. Private and public keywords must be indented 1 tab.
      ii. Data members and function must be indented 2 tabs.
   b. Content within brackets must be indented 1 tab further than the last bracket.
4. Brackets.
   a. Must begin at the end of the statement, loop, etc.
      ex.
      if (condition) {
            ;
      }
   b. All conditional blocks must use curly braces, even if they contain only one line of code.
5. Spacing.
   a. All arguments but the last must include a space after the comma.
   b. All operators must be padded with a pair of spaces.
   c. There must be spacing between statements and parentheses, and before brackets.
      ex.
      if<space>(condition)<space>{
      for<space>(int i = 0; i < 5; i++)<space>{

    d. There must be spaces after each semicolon in for statements.
6. Includes must be ordered alphabetically.
7. Headers.
    a. Must include only declarations, only implementations.
    b. Must contain all definitions.
    c. Must be structured in the following manner.
        i. File comment.
        ii. Include guard (open).
        iii. Includes.
        iv. Definitions.
        v. Declarations.
        vi. Include guard (close).
8. Classes.
    a. The private section must precede the public section.
    b. Both private and public keywords must exist, even if they are empty.
    c. No use of the protected section.
    d. Templates must be on the line preceding the class declarations.
    e. Constructors, destructors, operators, data members, and functions must be separated and commented in the following order and style.
    ex.

```
/* An example class. */
template<typename T>
class Example {
        private:
                /* Data member(s). */
                T exampleNumber; /* The example's number. */
        public:
                /* Constructor(s). */
                Example(T _exampleNumber);
                /* Destructor(s). */
                ~Example();
                /* Operator(s). */
                Example& operator = (const Example& example);
                /* Data member(s). */
                /* Function(s). */
                T getExampleNumber(); /* Return the example's number. */
}
```

    f. Inheritance must be formatted in the following fashion.
    ex. class Square : private Shape, public Polygon {...
    g. Private inheritances precede public inheritances, and must be ordered alphabetically.
    h. Initializer lists should begin on the line under the function name.
9. Other.

a. Integer constants must be replaced with appropriate definitions.
b. Empty conditional blocks must contain a semi-colon.
   ex.
   ```
   for (Iterator* i; i != NULL; i = i->next) {
           ;
   }
   ```
c. Argument names which are simple passes must be preceded by an underscore.
   ex.
   ```
   void setExampleNumber(int _exampleNumber) {
           exampleNumber = _exampleNumber;
   }
   ```