Spitfire Engine Framework

Copyright Robert “rmMinusR” Christensen 2020

The Spitfire Engine framework is not a true framework at all: Instead, it is a console application library exposing functions and classes to aid input logic and simplify display. Most input and utility functions are in sfio.h, as well as a couple basic display functions. textstyle.h contains more advanced rendering features such as colored text, dialogues, and ASCII pseudo-image objects.

Partially implemented:

- Switching between multiple games, where each game file must `#define` a `MAIN_FUNC`

Future version to-do list:

- Move everything to its own namespace
- Split files by purpose better
- Unity-like GameObject based true framework
- Logic, draw, and keyboard event hooks
- Game board management and definable win condition checks
- Game state serialization and IO

File contents:

sfio.h/.cpp

Intended for general utility functions and basic display.

- `char cquerych();`
    - Query the user for a character. Wait until one is given. Can be used as a safer, simpler alternative to system(“pause”)
- `char cquerycht(float timeout);`
    - Query the user for a character. If the user does not provide a character within `timeout` milliseconds, returns `‘\0’`.
- `void csetcurpos(unsigned int x, unsigned int y);`
    - Set the cursor position on the console.
- `int cgetw();`
    - Get width of the console window.
- `int cgeth();`
    - Get height of the console window.
- `void csetcurvis(bool visiblity);`
    - Set the cursor visibility. Doesn’t actually disable the cursor, only makes it small.

- `void cfill(char c, int x1, int y1, int x2, int y2);`
    - Fill an area on the screen with one character.
- `void cdrawbox(int x1, int y1, int x2, int y2);`
    - Draw a box on the screen.
- `void cclear();`
    - Clear the console.
- `void showDialog(std::string str, int x, int y);`
    - Display text, surrounded by a box. Centered on X and Y.

textstyle.h/.cpp

Advanced, image-like ASCII text and text blocks.

- `unsigned char ascol(bool r, bool g, bool b, bool light);`
    - Change rgb + light into a compatible color code
- `void csetcolb(unsigned char col);`
    - Set active text color, from color code
- `void csetcolc(bool r, bool g, bool b, bool light);`
    - Set active text color from components

textstyle.h/.cpp class TextStyle

Represents the style code applicable to text

- `TextStyle();`
    - White, unstyled
- `TextStyle(const bool& r, const bool& g, const bool& b, const bool& light);`
    - Constructor that generates a color code like `ascol()`
- `void setR(const bool& r);`
    - Set red channel
- `void setG(const bool& g);`
    - Set green channel
- `void setB(const bool& b);`
    - Set blue channel
- `void setLight(const bool& light);`
    - Set lightness channel
- `void applyStyle();`
    - Set the active style to be this one.
- `operator<<(std::ostream& out, const TextStyle& style);`
    - Allows TextStyle to be <<'d onto `cout` to apply current style.

textstyle.h/.cpp class StyledChar

Represents a char + TextStyle.

- `StyledChar();`

    - Alias for white `'\0'`
- `StyledChar(const char& c);`
    - White version of c
- `StyledChar(const char& c, const TextStyle& s);`
    - Stylized version of c
- `operator<<(std::ostream& out, const StyledChar& c);`
        - Print this styled character. Color only works when used with `cout`.

textstyle.h/.cpp class TextStyle

Represents a block of StyledChars

- `StyledTextBlock(const int& w, const int& h);`
    - Makes a blank block.
- `StyledTextBlock(const std::string& src);`
    - Makes a block exactly big enough to contain `src`
- `StyledTextBlock(const StyledTextBlock& src);`
    - Copy constructor.
- `void setStyledChar(const StyledChar& sc, const int& x, const int& y);`
    - Write a StyledChar to (x, y) on the internal text block
- `void setStyle (const TextStyle& style, const int& x, const int& y);`
    - Set the style of (x, y) on the internal text block
- `void setChar (const char& chars, const int& x, const int& y);`
    - Set the char of (x, y) on the internal text block. DOES NOT reset style code.
- `void fillStyledChar(const StyledChar& styledChar, const int& x1, const int& y1, const int& x2, const int& y2);`
    - Fill a rectangle with a StyledChar
- `void fillStyle (const TextStyle& style, const int& x1, const int& y1, const int& x2, const int& y2);`
    - Set a rectangle to use a TextStyle
- `void fillChar (const char& chars, const int& x1, const int& y1, const int& x2, const int& y2);`
    - Fill a rectangle with a char. Does not reset style.
- `void putStr (const std::string& str, const int& x, const int& y);`
    - Write a string at x, y. Clips if it hits an edge; does not wrap. Memory-safe.
- `void drawBox(const TextStyle& style, int x1, int y1, int x2, int y2);`
    - Draw a box (rectangle) using a TextStyle. Similar to sfio's `cdrawbox()`
- `void renderAt(const int& x, const int& y);`
    - Render this StyledTextBlock on the console at (x, y)