

# Advanced Higher Computing Science Project

*James McCorkindale*

*SCN: 1010 25152*



# Contents

Software Specification .....	6
Purpose .....	6
Scope.....	7
Software .....	7
Design.....	7
Implementation .....	7
Testing.....	7
Documentation .....	7
Evaluation .....	7
Website .....	8
Design.....	8
Implementation .....	8
Boundaries .....	9
Application Boundaries.....	9
Restrictions .....	9
End-User Requirements.....	9
Functional Requirements.....	10
Inputs .....	10
Processes.....	11
Output.....	12
Constraints .....	13
Technical .....	13
Software Design and Development .....	13
Website Design and Development .....	13
Economical .....	14
Software .....	14
Training .....	14
Timeline.....	14
Further Notes.....	15
UML Use Case Diagram.....	16
Gantt Chart Timeline.....	18
Design.....	21
Software Design .....	21
User Interface/Application Layout.....	21
Class Diagrams .....	22

Pseudo Code .....	34
Web Design .....	35
Hierarchal Diagram .....	35
User Interface .....	36
Implementation .....	39
Code for Software .....	39
DesignArk.h .....	39
DesignArk.cpp .....	41
MainFrm.h .....	48
MainFrm.cpp .....	50
ChildFrm.h .....	58
ChildFrm.cpp .....	59
Resource.h .....	61
pch.h .....	64
pch.cpp .....	65
framework.h .....	66
targetver.h .....	68
CDocView.h .....	69
CDocView.cpp .....	70
CTextDocView.h .....	71
CTextDocView.cpp .....	74
CTextDocument.h .....	111
CTextDocument.cpp .....	112
CTextHandler.h .....	120
CTextHandler.cpp .....	121
CApObject.h .....	130
CApObject.cpp .....	131
CTextEditorObject.h .....	134
CTextEditorObject.cpp .....	137
CTextLineObject.h .....	168
CTextLineObject.cpp .....	170
CSmartColour.h .....	185
CSmartColour.cpp .....	186
CSmartColourStatements.h .....	187
CSmartColourStatements.cpp .....	188
CSmartColourTypes.h .....	191

CSmartColourTypes.cpp.....	192
CSmartColourBuiltin.h .....	194
CSmartColourBuiltin.cpp.....	195
CSmartColourOther.h .....	198
CSmartColourOther.cpp.....	199
CAboutDlg.h .....	201
CAboutDlg.cpp .....	202
Code for Website .....	203
index.php .....	203
downloads.php .....	205
userdocs.php.....	208
banner.html .....	210
nav.html .....	211
footer.html.....	213
Final Product .....	215
Application .....	215
Website .....	216
New Skills/Knowledge Evaluation.....	218
Log of On-Going Tests .....	219
Highlighting Bug .....	219
Error When Pressing the Sidebar Button After Highlighting .....	221
About Dialog Assertion Failure .....	222
Copying Nothing Bug.....	224
Testing.....	225
Test Plan.....	225
Persona Testing.....	229
Testing Requirements (Actual Results of Texting) .....	230
Test Case ID 1.....	230
Test Case ID 2.....	230
Test Case ID 3.....	231
Test Case ID 4.....	232
Test Case ID 5.....	233
Test Case ID 6.....	234
Test Case ID 7.....	235
Test Case ID 8.....	235
Test Case ID 9.....	236

Test Case ID 10.....	236
Test Case ID 11.....	237
Test Case ID 12.....	237
Test Case ID 13.....	237
Test Case ID 14.....	238
Test Case ID 15.....	239
Test Case ID 16.....	239
Test Case ID 17.....	240
Test Case ID 18.....	240
Test Case ID 19.....	241
Test Results .....	242
Test Case ID 1.....	242
Test Case ID 2.....	242
Test Case ID 3.....	242
Test Case ID 4.....	242
Test Case ID 5.....	242
Test Case ID 6.....	242
Test Case ID 7.....	242
Test Case ID 8.....	242
Test Case ID 9.....	242
Test Case ID 10.....	242
Test Case ID 11.....	242
Test Case ID 12.....	242
Test Case ID 13.....	242
Test Case ID 14.....	243
Test Case ID 15.....	243
Test Case ID 16.....	243
Test Case ID 17.....	243
Test Case ID 18.....	243
Test Case ID 19.....	243
Evaluation .....	244
Evaluation Report .....	244
Fitness For Purpose.....	244
Maintainability.....	245
Robustness.....	245

# Software Specification

## Purpose

The purpose of this program is to allow the end-user to be able to design their software project's pseudocode, in a Windows Desktop Application, with specialised features. This application will have unique features that will be implemented such as SmartColour which will enable the user to define colours for specific key words depending on their writing style, as well as subline editing capabilities to write pseudo code with sublines.

A website will also be developed with this project. It will be a basic website that will have multiple web pages, allowing the user to view a homepage (which describes the application and displays some of its features), a download page (which allows the user to download the application and the user documentation) and a page which the user may view the user documentation in.

To meet the SQA Advanced Higher Computer Science project requirements, the software will be written with object-orientated programming and will include at-least one array of 'CString' objects in each property page of the CSmartColour property sheet (CSmartColourTypes, CSsmartColourStatements, CSsmartColourBuiltin, CSsmartColourOther). Each array is a linked list using the 'std::vector' type. These arrays will all be sorted using an insertion sort whenever the user closes its associated property sheet.

## Scope

### Software

#### *Design*

- A completed design of the software structure includes:
  - A design that precisely meets the end-user requirements
  - Application class structure
  - Pseudocode for the implemented standard algorithm, insertion sort
- A completed design of the user interface that includes:
  - A wireframe diagram of the application, including:
    - Main view
    - Dialog boxes
  - Descriptions that show inputs, processes, and outputs, labelled on the wireframe diagram

#### *Implementation*

- A working application which allows the user to design pseudo code for software projects
- The application software must precisely match the design
- The application user interface must precisely match the design
- Description of new skills and/or knowledge researched and developed
- A log of on-going testing, including:
  - A description of issues resolved
  - References used to resolve these issues

#### *Testing*

- A completed test plan for final testing of the fully implemented solution, including:
  - All requirements
  - Description of tests
  - Persona and test cases
- Evidence of requirements testing
- A report on the results of the final testing, in terms of fitness for purpose, maintainability and robustness

#### *Documentation*

- User documentation that is to be available via the website

#### *Evaluation*

- A detailed evaluation report in terms of:
  - Fitness for purpose, by discussing:
    - How closely the solution matches the requirements specification
    - The results of the testing
  - Maintainability
  - Robustness

## Website

### *Design*

- A design that precisely meets the end-user requirements
- A completed design of the website interface that includes:
  - A hierarchical diagram
  - Wireframe Diagrams
  - Descriptions that show inputs, processes, and outputs, labelled on the wireframe diagram

### *Implementation*

- A robust website that allows the user to:
  - Download the software package and install it onto their Windows machine
  - Download user documentation as a PDF
  - View user documentation on the website
- The website must precisely match the design

## Boundaries

### Application Boundaries

#### *Restrictions*

- The user will only be able to use monospaced fonts

## End-User Requirements

The user will expect to be able to complete the following tasks:

- Create a new file that will allow the end-user to write pseudo code
- In a file, be able to create and develop pseudo code:
  - Write in a text editor with standard functions like highlighting, copy and paste, etc
  - Add sublines for each line
    - Edit them as if they were their own text editor
    - Increment/decrement when the parent line is incremented/decremented
    - Restriction on editing them
      - Cannot remove the line via highlight
      - Cannot create a new line in the middle of the text
  - Zoom in and zoom out of the file
  - For all text be able to do the following:
    - Change fonts (SmartColour)
    - Change colours for set words and add new colours and words (SmartColour)
    - Change the colour of the highlighter
  - Be able to print the files in a slick layout (with the SmartColour settings)
  - Export project as PDF (same as what will be printed)
  - Be able to save and load files

## Functional Requirements

### Inputs

Take user inputs for the following functions. They will either be in the form of buttons, shortcuts, or interaction with the view. Not all functions will be always available.

- Menu Functions
  - File
    - New (text file)
    - Open
    - Close
    - Save
    - Save as
    - Print
    - Print Setup
    - Export as PDF
    - Open a recent file
    - Exit
  - Edit
    - Cut
    - Copy
    - Paste
  - View
    - Toolbars and docking windows
      - Enable/Disable Toolbar
      - Customise Menu bar
    - Enable/Disable status bar
    - Open SmartColour dialog box
      - Edit keywords
      - Edit colours
      - Edit indentation sizes
      - Edit fonts
      - Edit comments type
  - Window
    - New window
    - Manage tabs
  - Help
    - About
    - Help
      - Open Website
      - View user documentation on website

- Tool bar buttons
  - New (text file)
  - Open
  - Save
  - Cut
  - Copy
  - Paste
  - Print
  - About
- Interact with CDocView
  - Text editor/pseudocode editor
    - Create new line in place
    - Create a new subline
    - Backspace function
      - Deletes previous character
      - Deletes Line
      - Deletes highlighted segment
      - Delete subline
    - Interact with caret
      - Arrow keys to move it
      - Click mouse to move it
    - Edit highlighted region
      - Copy
    - Paste clipboard

## Processes

- Menu Functions
  - File
    - Create a file as a tab for editing
    - Open a file as a tab for editing
    - Save files
    - Open OS printing and communicate with OS to print file
    - Edit OS print settings
    - Save a file as pdf (Export as PDF)
    - Exit the application by completing any processes, handling the memory etc, and close the window
  - Edit
    - Copy text to the clipboard
    - Receive text from the clipboard and insert them into the application (paste)
  - View
    - Open a window that enables the customisation of the main window
    - Make the status bar visible and invisible
    - Edit the theme of the main window (colours, styles etc)
    - Open and close any window panels (tabs)
    - Open the SmartColour dialog box
      - Add/remove keywords
      - Change colour for keywords, comment, highlight colour and number colour

- Change font, indentation size
- Window
  - Open a new window (run a new instance of Design Ark app)
  - Open a new window to manage open tabs and be able to close them from the window
- Help
  - Open website, in the default browser
  - Open a window displaying info on the application
- Toolbar Functions
  - Create a file as a tab for editing
  - Open a file as a tab for editing
  - Save files
  - Copy objects to the clipboard
  - Receive objects from the clipboard and insert them into the application
  - Open OS printing and communicate with OS to print file
  - Open a window displaying info on the application
- Interact with CDocView
  - Add/Remove text to lines
  - Add/remove any lines and subline blocks
  - Run any necessary GUI objects such as highlighting and SmartColour

## Output

- Functions
  - Prints to a printer
  - A PDF formatted document with the pseudo code
  - Save as a .txt file, which can be imported back into the app, or edited in another text editor
- CDocViews
  - Text on each line as appropriate in a readable format
  - Text may be coloured depending on the SmartColour formats
  - View will render at an appropriate time, ensuring the windows aren't rendering when they don't have to, to make the program more efficient

## Constraints

### Technical

Following is a list showing what packages and software this project will be using during what stages.

### General

Operating systems used in the project:

- Windows 10
- Windows 11

### Analysis, Testing and Evaluation

Software that was used to construct the analysis, testing and evaluation:

- app.diagrams.net
- Microsoft Word
- Microsoft Excel
- ilovepdf.com (for merging PDFs together)
- Microsoft Edge
- Google Chrome

### Software Design and Development

#### Design

Software that was used to construct the software design:

- app.diagrams.net

#### Implementation

IDEs used for software development:

- Microsoft Visual Studio 2019
- Microsoft Visual Studio 2022

Languages used for software development:

- C++
- C

Notable Internal Dependencies (Header Files) used for software development:

#### MFC Libraries

- afxwin.h
- afxext.h
- afxdisp.h
- afxcmn.h
- afxcontrolbars.h

### Website Design and Development

#### Design

Software that was used to construct the user interface:

- app.diagrams.net

## Implementation

IDE used for software development:

- Microsoft Visual Studio Code

Languages used for software development:

- HTML
- CSS
- Javascript
- PHP

Other Packages Used:

- Apache Server

## Economical

### *Software*

- This project meets the requirements for the MS Visual Studio Community license and so there will be no cost associated with purchasing an IDE
- There are no additional costs associated with acquiring the MS 365 software packages used in the analysis and design stage as they have already been purchased by me and my place of education
- To create the development environment for the application, an add-on feature/tool for MS Visual Studio 2019 will need to be downloaded. This feature will take up around 74mb of data so is a negligible cost of production
- The final package will require to be run on a Windows 10 operating system

### *Training*

- I will have to increase background knowledge on Win32 API and Microsoft Foundation Classes (MFC). This could take time out of development but has been factored into the timescale.

### *Timeline*

Project Start: 23/09/2021

Deadline: 13/01/2021

Analysis Start: 23/09/2021

Developer Training and Upskilling: 23/09/2021

Design Start: 15/10/2021

Implementation Start: 15/11/2021

Testing Start: 21/02/2022

Documentation Start: 18/03/2022

Evaluation Start: 23/03/2022

\*For precise timeline view Gantt chart attached

\*All development sections are completed concurrently, with a small overlap of training and upskilling, and design

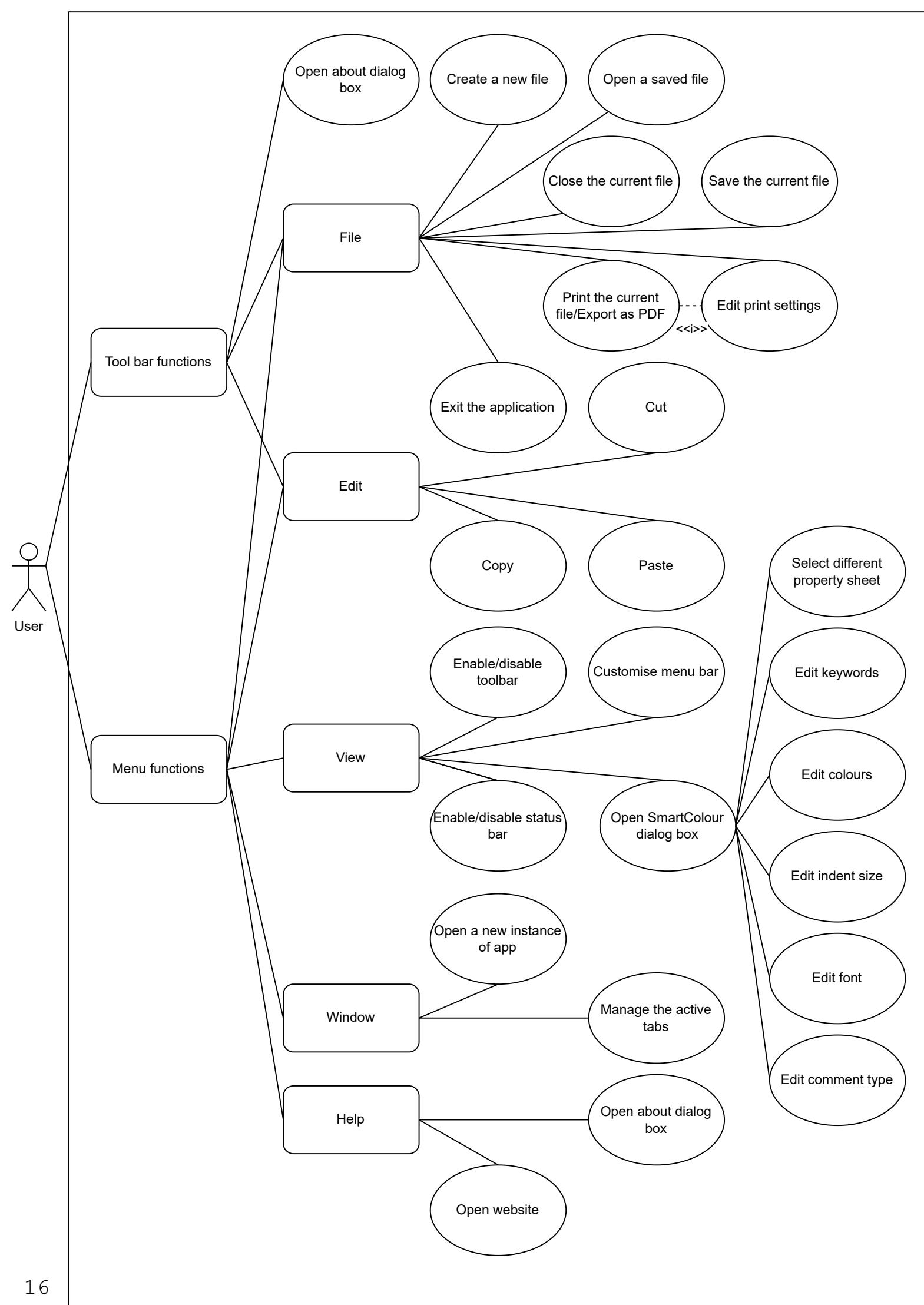
## Further Notes

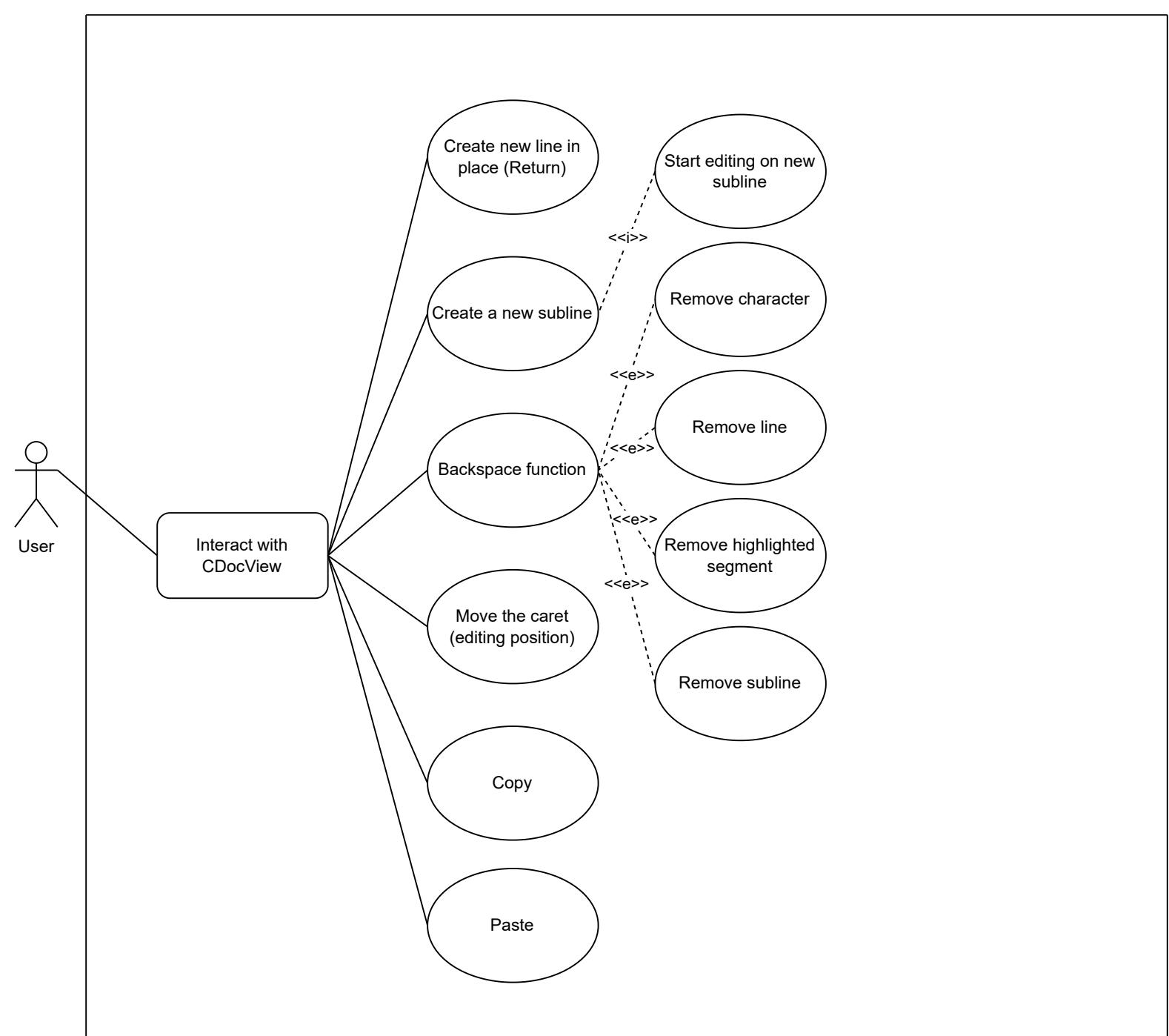
### General

- As of 28/03/2022, the device being used to make the project can upgrade to Windows 11. Everything will be moved over onto the new operating system.

### Implementation

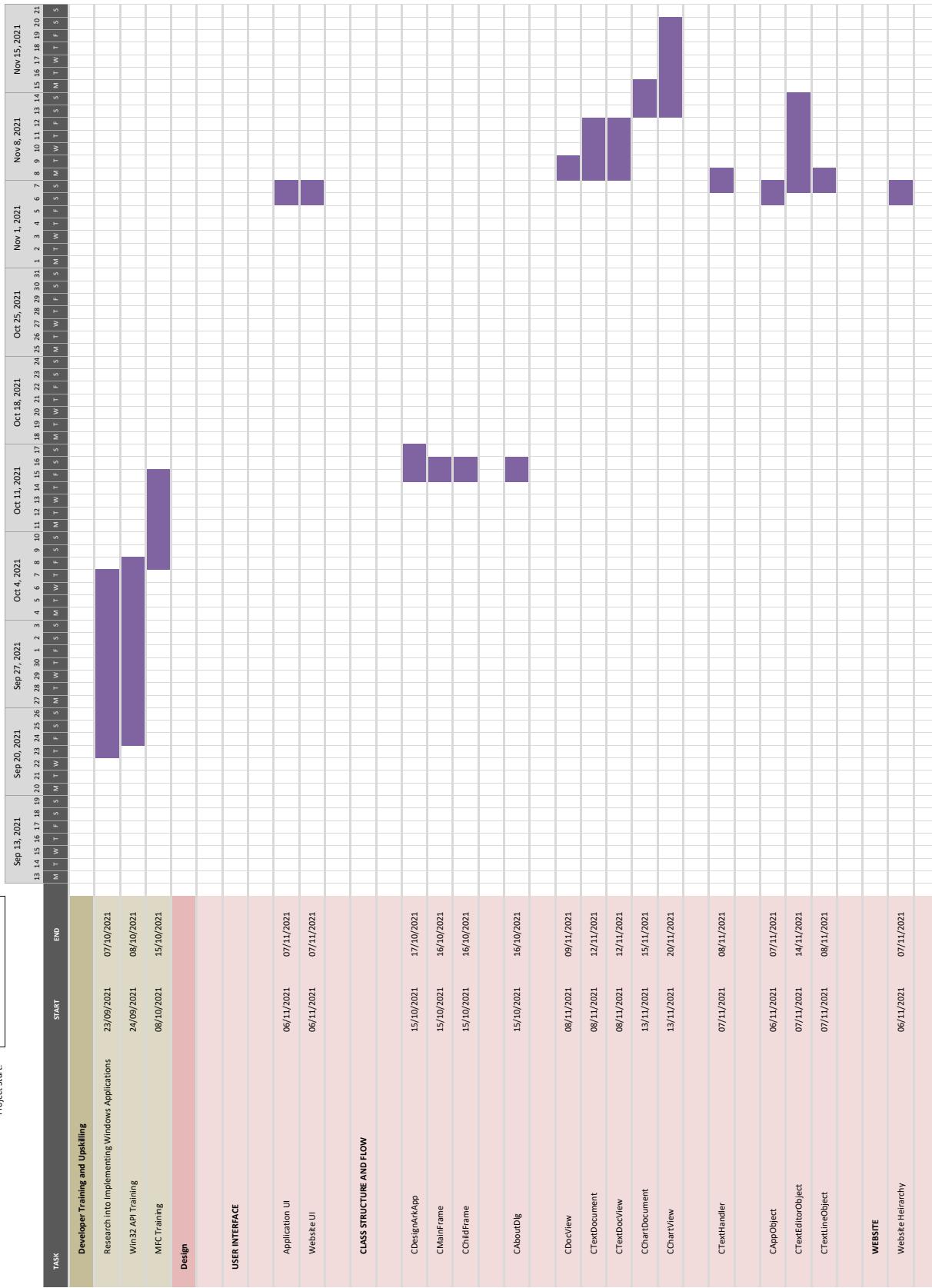
- As of 08/11/2021, MS VS 2022 is released so from that point forth, the rest of the implementation will be using that IDE and any previously written code will be moved for use with that IDE





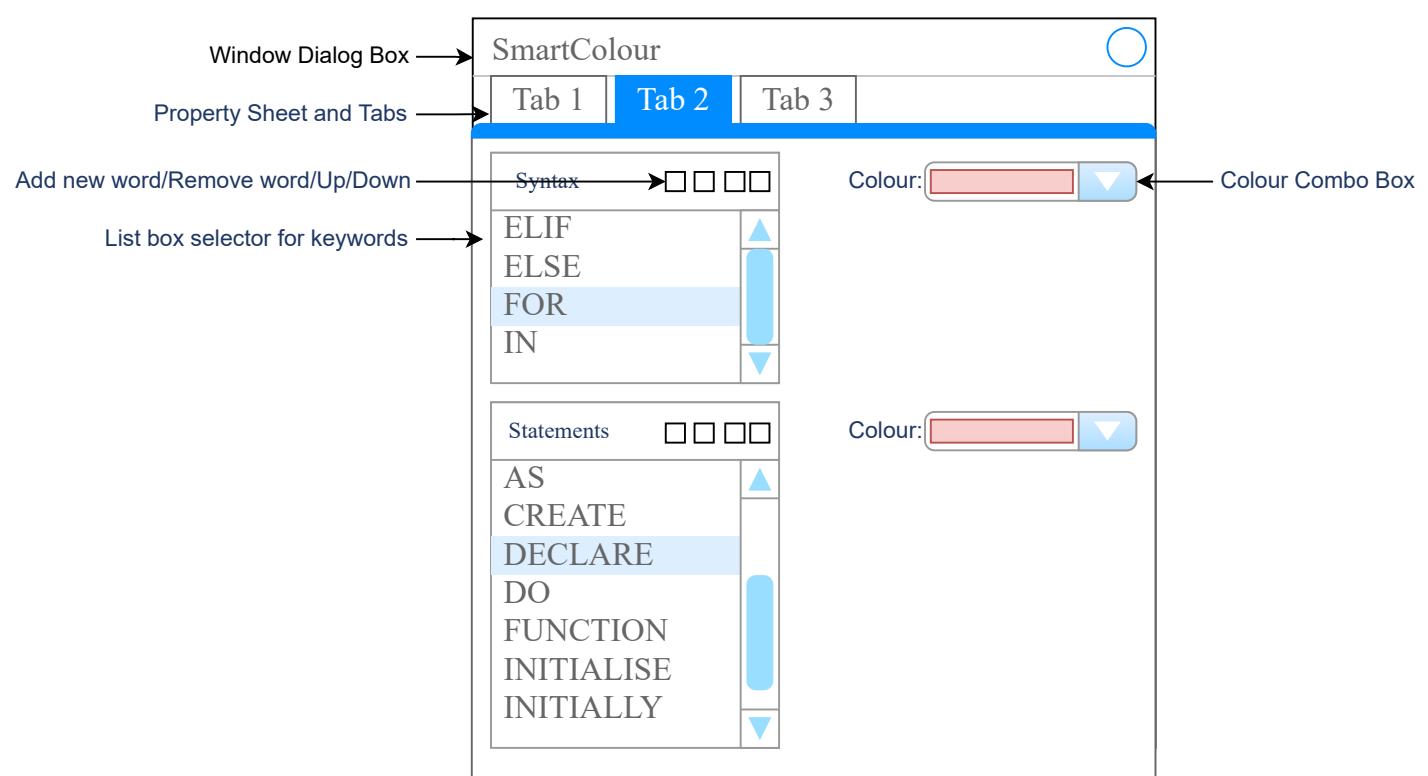
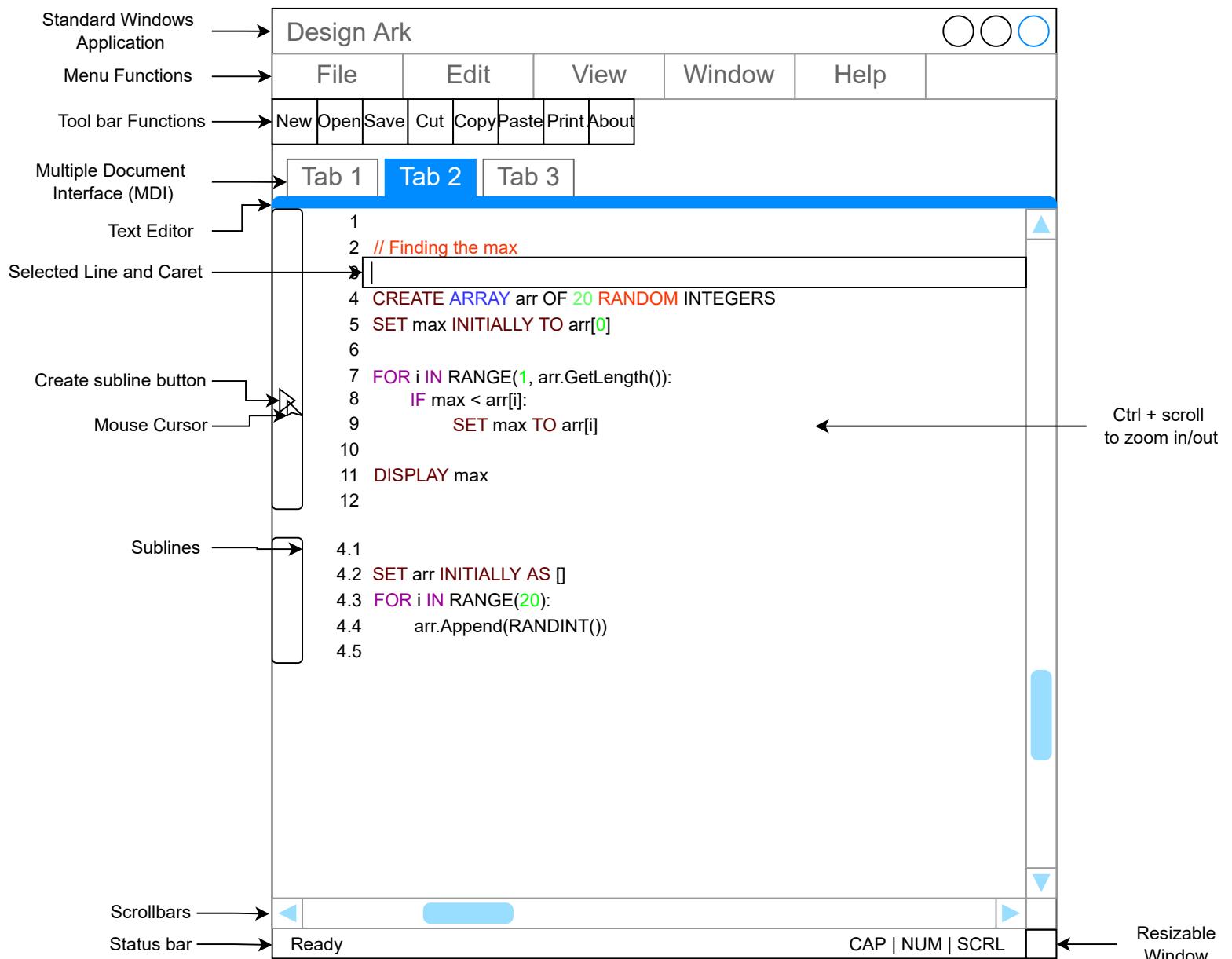
Advanced Higher Computing Project

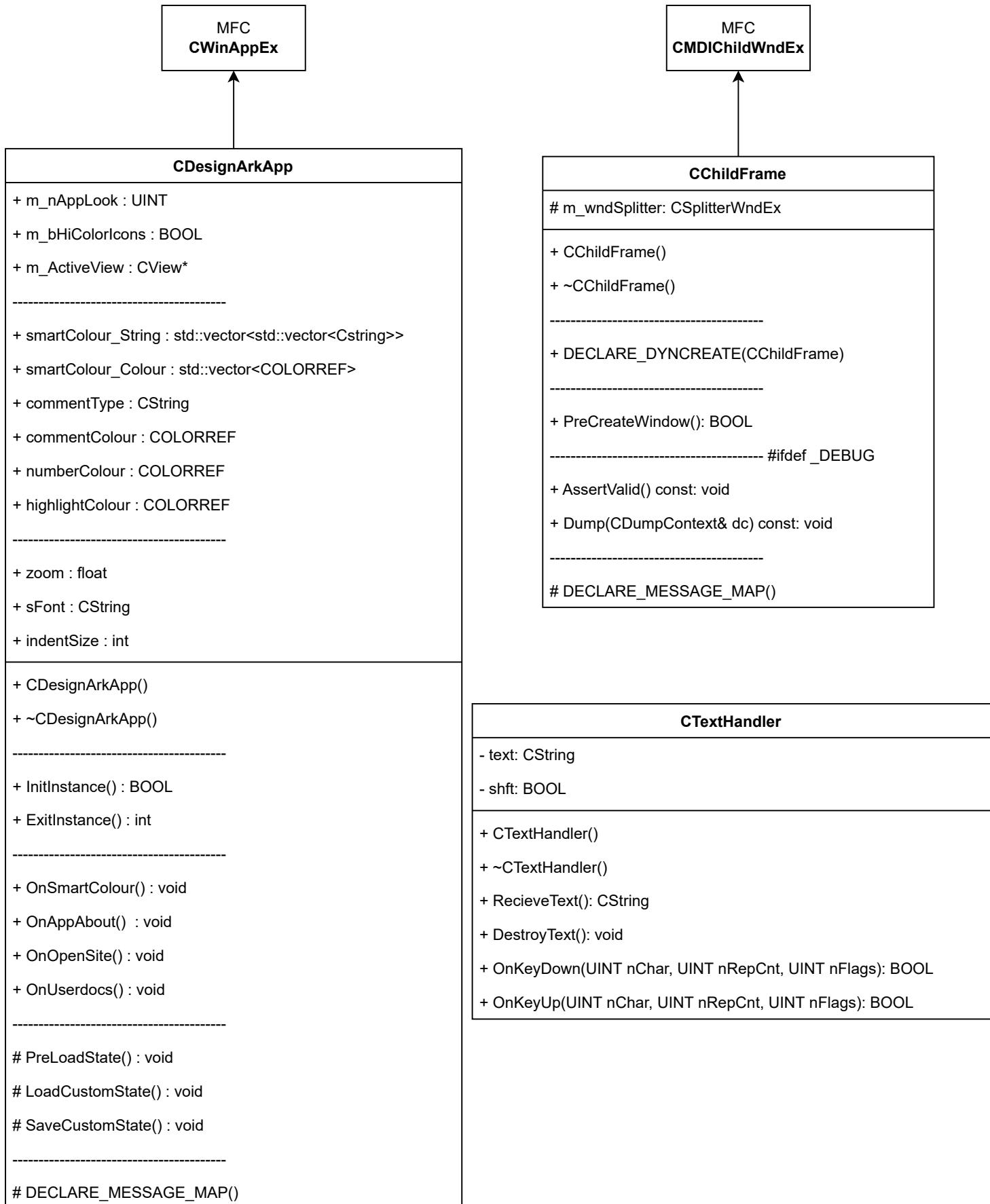
Project Start: 23/09/2021

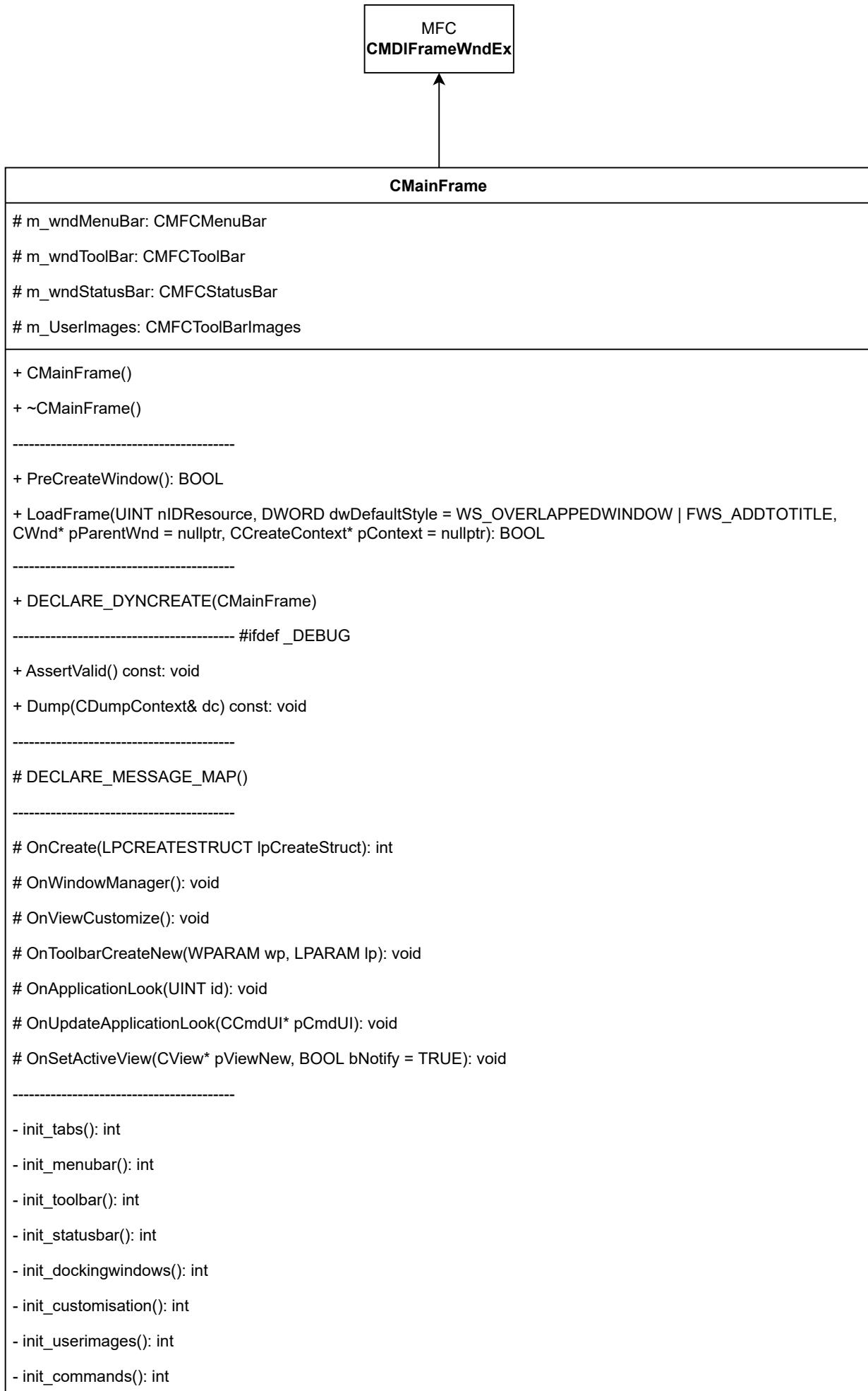


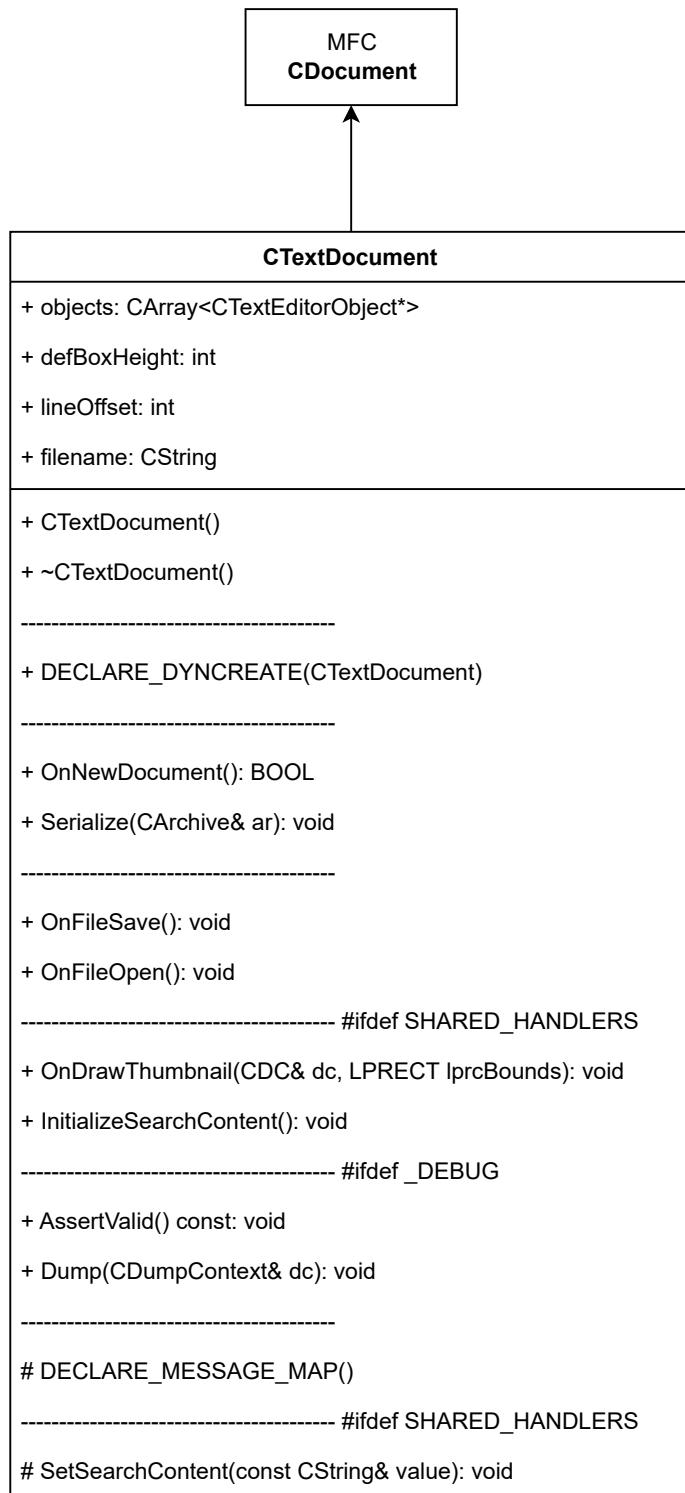


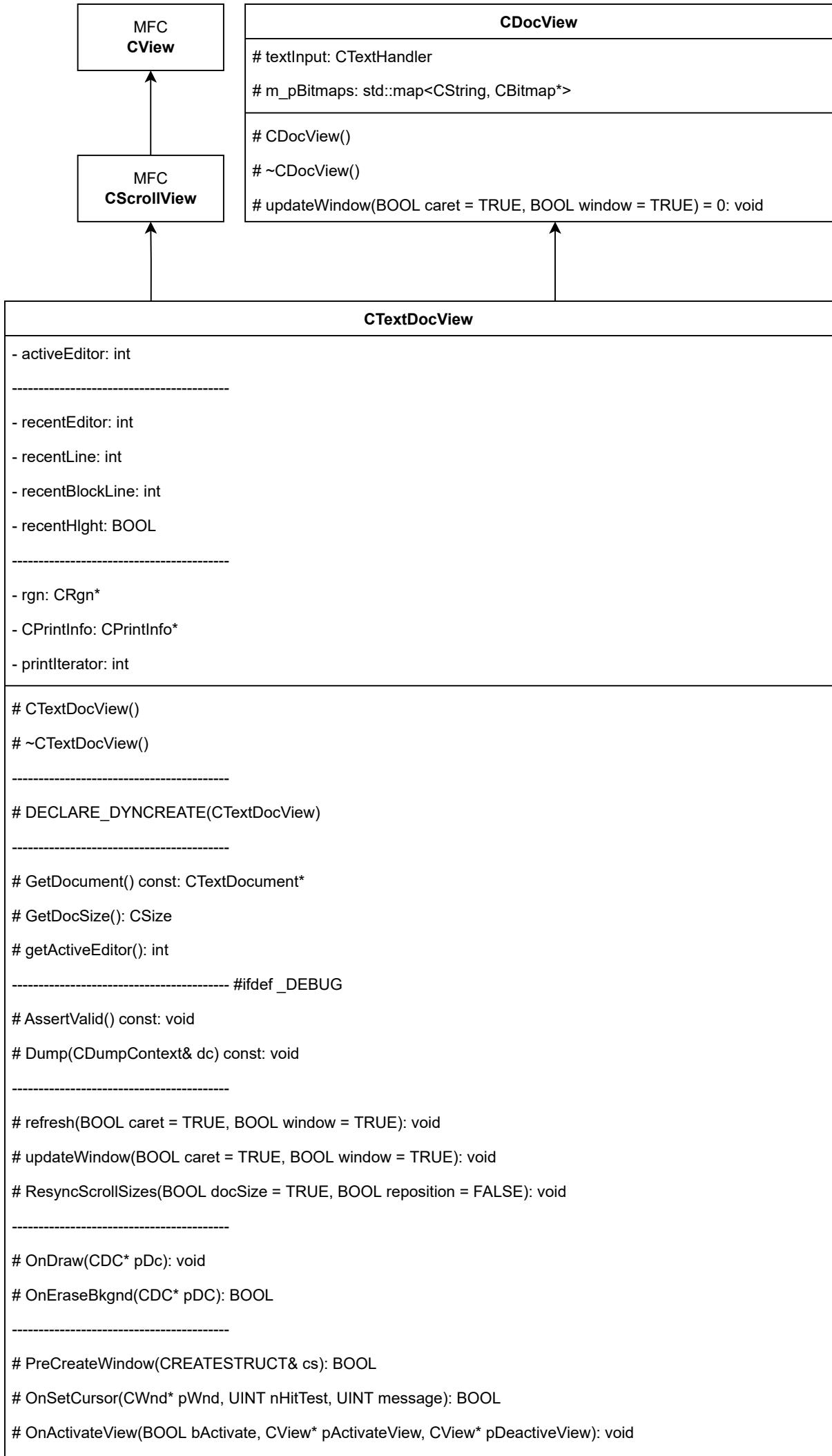
TASK	Feb 14, 2022	Feb 21, 2022	Feb 28, 2022	Mar 7, 2022	Mar 14, 2022	Mar 21, 2022	Mar 28, 2022
	14 15 16 17 18 19 20 21 22 23 24 25 26 27 28	1 2 3 4 5 6 7 8 9 10 11 12 13	14 15 16 17 18 19 20 21 22 23 24 25 26 27 28	21 22 23 24 25 26 27 28 29 30 31 1 2 3			
START							
END							
Testing							
Final Comprehensive Testing Plan	21/02/2022	26/02/2022					
Requirements Testing	01/03/2022	11/03/2022					
Test Evaluation	14/03/2022	17/03/2022					
Documentation							
End-user Documentation	18/03/2022	22/03/2022					
Evaluation							
Evaluation Report	23/03/2022	28/03/2022					











```
# OnPreparePrinting(CPrintInfo* pInfo): BOOL
# OnBeginPrinting(CDC* pDc, CPrintInfo* pInfo)): void
# OnEndPrinting(CDC* pDc, CPrintInfo* pInfo): void
# OnPrepareDC(CDC* pDC, CPrintInfo* pInfo = (CPrintInfo*)0): void
# OnPrint(CDC* pDC, CPrintInfo* pInfo): void

-----
# OnCreate(LPCREATESTRUCT lpCs): int
# OnInitialUpdate(): void
# OnSize(UINT nType, int cx, int cy): void

-----
# OnHScroll(UINT nSBCode, UINT nPos, CScrollBar* pScrollBar): void
# OnLButtonUp(UINT nFlags, CPoint point): void
# OnLButtonDown(UINT nFlags, CPoint point): void
# OnLButtonDblClk(UINT nFlags, CPoint point): void
# OnMouseMove(UINT nFlags, CPoint point): void
# OnMouseWheel(UINT nFlags, short zDelta, CPoint pt): BOOL
# OnRButtonUp(UINT nFlags, CPoint point): void
# OnRButtonDown(UINT nFlags, CPoint point): void
# OnKeyDown(UINT nChar, UINT nRepCnt, UINT nFlags): void
# OnKeyUp(UINT nChar, UINT nRepCnt, UINT nFlags): void

-----
# OnContextMenu(CWnd* pWnd, CPoint point): void
# OnCopy(): void
# OnPaste(): void

-----
# DECLARE_MESSAGE_MAP()

-----
- expandLineBounds(): void
- selectClippingRgn(int nAction): void
- drawHeader(CDC* pDc): void
```

MFC  
**CPropertySheet**

**CSmartColour**

```
# types: CSmartColourTypes
# statements: CSmartColourStatements
# builtin: CSmartColourBuiltin
# other: CSmartColourOther

+ CSmartColour(std::vector<std::vector<CString>> page_strings, std::vector<COLORREF> page_colour,
CString commentType, COLORREF commentColour, COLORREF numberColour, COLORREF highlightColour, CString fonts, CString indents,
LPCTSTR pszCaption = L"", CWnd* pParentWnd = nullptr, UINT iSelectPage = 0)
+ ~CSmartColour()

-----
# DECLARE_DYNAMIC(CSmartColour)
# DECLARE_MESSAGE_MAP()
```

MFC  
**CMFCPropertyPage**

**CSmartColourTypes**

```
----- #ifdef AFX DESIGN TIME
+ enum { IDD = IDD_SMARCOLOUR_TYPES}

-----
+ m_Listbox_Types_control: CVSListBox
+ m_Combo_Types_colour_control: CMFCCColorButton
+ m_Combo_Types_colour_var: COLORREF
+ m_listbox_types_strings: std::vector<CString>

+ CSmartColourTypes(std::vector<CString> types_string,
COLORREF colour, CWnd* pParent = nullptr)
+ ~CSmartColourTypes()

-----
# DoDataExchange(CDataExchange* pDX): void
# OnInitDialog(): BOOL
# OnKillActive(): BOOL

-----
# DECLARE_DYNAMIC(CSmartColourTypes)
# DECLARE_MESSAGE_MAP()
```

**CSmartColourStatements**

```
----- #ifdef AFX DESIGN TIME
+ enum { IDD = IDD_SMARCOLOUR_OPS}

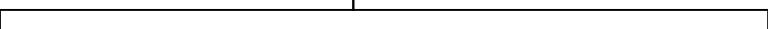
-----
+ m_Listbox_Statements_syntax_control: CVSListBox
+ m_Combo_Statements_syntax_control: CMFCCColorButton
+ m_Listbox_Statements_statments_control: CVSListBox
+ m_Combo_Statements_statments_control: CMFCCColorButton
+ m.Strings_Statements_syntax_var: std::vector<CString>
+ m_Colour_Statements_syntax_var: COLORREF
+ m.Strings_Statements_statments_var: std::vector<CString>
+ m_Colour_Statements_statments_var: COLORREF

+ CSmartColourStatements(std::vector<CString> syntax_string,
COLORREF syntax_colour, std::vector<CString> statement_string,
COLORREF statement_colour, CWnd* pParent = nullptr)
+ ~CSmartColourStatements()

-----
# DoDataExchange(CDataExchange* pDX): void
# OnInitDialog(): BOOL
# OnKillActive(): BOOL

-----
# DECLARE_DYNAMIC(CSmartColourStatements)
# DECLARE_MESSAGE_MAP()
```

MFC  
**CMFCPropertyPage**



**CSmartColourOther**

```

----- #ifdef AFX DESIGN TIME

+ enum { IDD = IDD_SMARCOLOUR_OTHER }

-----
+ m_pCombo_comments_type_control: CComboBox
+ m_pCombo_comments_colour_control: CMFCCColorButton
+ m_pCombo_numbers_colour_control: CMFCCColorButton
+ m_pCombo_highlight_colour_control: CMFCCColorButton
+ m_pCombo_fonts_control: CComboBox
+ m_pCombo_indent_control: CEdit
+ m_pCombo_comments_type_var: CString
+ m_pCombo_comments_colour_var: COLORREF
+ m_pCombo_numbers_colour_var: COLORREF
+ m_pCombo_highlight_colour_var: COLORREF
+ m_pCombo_fonts_var: CString
+ m_pCombo_indent_var: CString

+ CSmartColourOther(CString comments_type,
COLORREF comments_colour,COLORREF numbers_colour,
COLORREF highlight_colour, CString fonts,
CString indentSize, CWnd* pParent = nullptr)
+ ~CSmartColourOther()

-----
# DoDataExchange(CDataExchange* pDX): void
# OnInitDialog(): BOOL
# OnKillActive(): BOOL

-----
# DECLARE_DYNAMIC(CSmartColourOther)
# DECLARE_MESSAGE_MAP()

```

**CSmartColourBuiltin**

```

----- #ifdef AFX DESIGN TIME

+ enum { IDD = IDD_SMARCOLOUR_BUILTIN }

-----
+ m_Listbox_Builtin_Classes_control: CVSListBox
+ m_Combo_Builtin_Classes_colour_control: CMFCCColorButton
+ m_Listbox_Builtin_Functions_control: CVSListBox
+ m_Combo_Builtin_Functions_colour_control: CMFCCColorButton
+ m_Listbox_Builtin_Classes_var: std::vector<CString>
+ m_Combo_Builtin_Classes_colour_var: COLORREF
+ m_Listbox_Builtin_Functions_var: std::vector<CString>
+ m_Combo_Builtin_Functions_colour_var: COLORREF

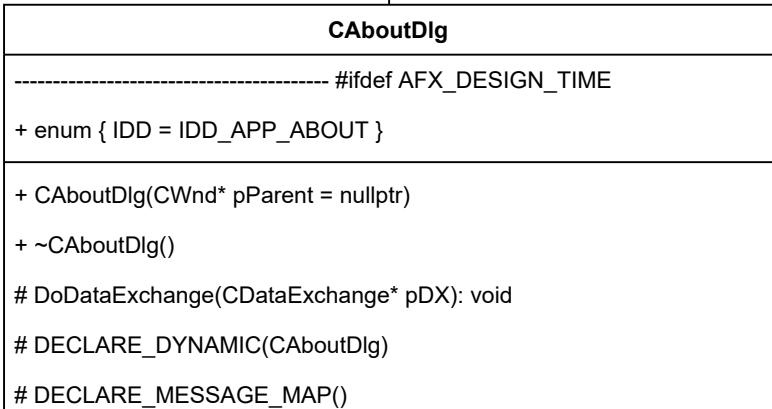
+ CSmartColourBuiltin(std::vector<CString> classes_string,
COLORREF classes_colour, std::vector<CString> functions_string,
COLORREF functions_colour, CWnd* pParent = nullptr)
+ ~CSmartColourBuiltin()

-----
# DoDataExchange(CDataExchange* pDX): void
# OnInitDialog(): BOOL
# OnKillActive(): BOOL

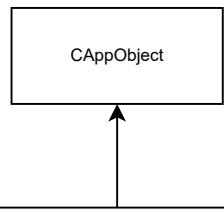
-----
# DECLARE_DYNAMIC(CSmartColourBuiltin)
# DECLARE_MESSAGE_MAP()

```

MFC  
**CDlgEx**



<b>CAppObject</b>
# ID: CString
# bounds: CRect
# active: BOOL
+ CAppObject(CRect bounds, CString ID, BOOL active = FALSE)
+ ~CAppObject()
-----
+ draw(CDC* pDc): void
-----
+ OnSize(UINT nType, int cx, int cy): void
+ OnLButtonUp(UINT nFlags, CPoint point): void
+ OnLButtonDown(UINT nFlags, CPoint point): void
+ OnLButtonDblClk(UINT nFlags, CPoint point): BOOL
+ OnRButtonUp(UINT nFlags, CPoint point): void
+ OnRButtonDown(UINT nFlags, CPoint point): void
+ OnMouseMove(UINT nFlags, CPoint point): BOOL
-----
+ OnRecieveText(CString text): int
+ OnRecieveBackspace(): void
+ OnRecieveReturn(): void
+ OnRecieveTab(): void
-----
+ OnKeyDown(UINT nChar, UINT nRepCnt, UINT nFlags): void
+ OnContextMenu(CWnd* pWnd, CPoint point): BOOL
-----
+ getID(): CString
+ setID(CString ID): void
+ getBounds(): CRect
+ checkPointInBounds(CPoint point): BOOL
+ setBounds(CRect bounds): void
+ setBounds(int x, int y, int cx, int cy): void
+ setPosition(CPoint point): type
+ move(int x, int y): void
+ isActive(): BOOL
+ setActive(): void



### CTextEditorObject

- activeLine: int

- mouseLine: int

- blockLine: int

- printLine: int

- caretPos: int

- defBoxHeight: int

- textExtent: CSize

- hlight: BOOL

- hlghting: BOOL

- hlghtStartP: CPoint

- hlghtEndP: CPoint

- startPos: int

- startLine: int

- recentHlight: type

- recentPos: int

- recentLine: int

- lines: std::vector<CTextLineObject\*>

- maxLines: int

- lineOffset: int

- lineNums: BOOL

- hoverSubLine: BOOL

- clickSubLine: BOOL

- sidebar: CRect

- brackets: std::vector<std::vector<int>>

- cursor\_arrow: int

- lMouseDown: BOOL

+ CTextEditorObject(CRect bounds, CString ID, BOOL lineNums, int maxLines = 0, int defBoxHeight = 0, BOOL active = FALSE, std::vector<CString> text = {L""}, std::vector<int> line = {1}, int lineOffset = 0)

+ ~CTextEditorObject()

+ draw(CDC\* pDc, CSize textExtent, int xScrollPosition, int returnNewLines, BOOL printing, CRect printAreaLength = CRect()): int

```
+ getCaretPos(): int
+ getCaretPoint(CSize caretSize): CPoint
+ getTextExtent(): CSize
+ setTextExtent(CSize size): void
+ getRecentPos(): int
+ pointHighlighted(CPoint point): BOOL
+ hasHighlight(): BOOL
+ hlightingOff(): void
+ getHighlightClippingRect(): CRect
+ getHighlightExactRgn(int x_offset, int y_offset): CRgn*
+ getStartLine(): int
+ isHlghtMultiline(): BOOL
+ lineHighlight(int line): int
+ iGetLineNum(int a = 0): std::vector <int>
+ sGetLineNum(int a = 0): CString
+ incrementSublines(int subline, int val): void
+ getActiveLine(): int
+ getBlockLine(): int
+ getPrintLine(): int
+ getLineText(int line = 0): CString
+ getLineBounds(int line = 0): CRect
+ getHighlightedText(): CString
+ getLineTextWidth(): int
+ getNumLines(): int
+ getBoxHeight(BOOL default = FALSE): int
+ setActive(BOOL active): void
+ move(int x, int y): void
+ setBounds(CRect bounds): void
+ getPrintBounds(int returnNewLines, int printAreaLength): std::tuple<CRect, int>
+ getCursorArrow(): int
-----
+ initialise(): void
-----
+ OnSize(UINT nType, int cx, int cy): void
+ OnLButtonUp(UINT nFlags, CPoint point): void
+ OnLButtonDown(UINT nFlags, CPoint point): void
+ OnLButtonDblClk(UINT nFlags, CPoint point): BOOL
+ OnRButtonUp(UINT nFlags, CPoint point): void
+ OnRButtonDown(UINT nFlags, CPoint point): void
+ OnMouseMove(UINT nFlags, CPoint point): BOOL
-----
+ OnRecieveText(CString text): int
+ OnRecieveBackspace(): void
+ OnRecieveReturn(): void
+ OnRecieveTab(): void
```

+ OnKeyDown(UINT nChar, UINT nRepCnt, UINT nFlags): void

+ OnContextMenu(CWnd\* pWnd, CPoint point): BOOL

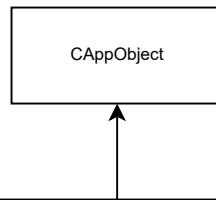
- setSidebar(): void

- setHighlighter(): void

- setBracketsNull(): void

- bracketContains(int value, int& itPos, int type, int side = 1): BOOL

- moveBrackets(int val, int index): void



```

CTextLineObject

- text: CString
- highlighter: Gdiplus::Rect
- highlight: BOOL
- hStart: int
- hEnd: int
- lineHighlight: BOOL
- numSubLines: int
- lineNums: std::vector<int>
- smartColour: BOOL

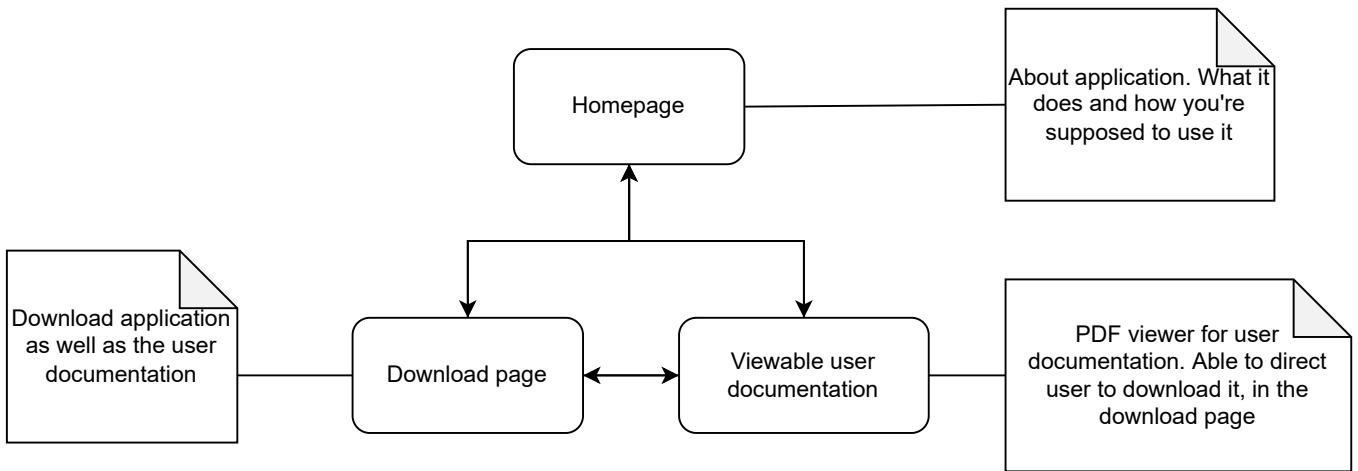
+ CTextLineObject(CRect bounds, CString ID, std::vector<int> lineNums, CString text = L"", BOOL active = FALSE)
+ ~CTextLineObject()

-----
+ draw(CDC* pDc, CSize textExtent, std::vector<std::vector<int>> brackets, int returnNewLines, BOOL printing, int printAreaLength = -1): int

-----
+ getText(): CString
+ getLength(): int
+ setText(CString text): void
+ concatenateString(CString text, int position): void
+ backspace(int position): void
+ isHighlighting(): BOOL
+ setHighlighting(BOOL val): void
+ setHighlighter(int pos1, int pos2): void
+ highlightLine(): void
+ isLineHighlighted(): BOOL
+ getHStart(): int
+ getHEnd(): int
+ getHighlightedText(): CString
+ getNumSubLines(): int
+ iGetLineNums(): std::vector<int>
+ sGetLineNums(): CString
+ addSublines(std::vector<int> subs): void
+ incrementLine(int subline, int val): void
+ setBounds(CRect bounds): void
+ move(int x, int y): void

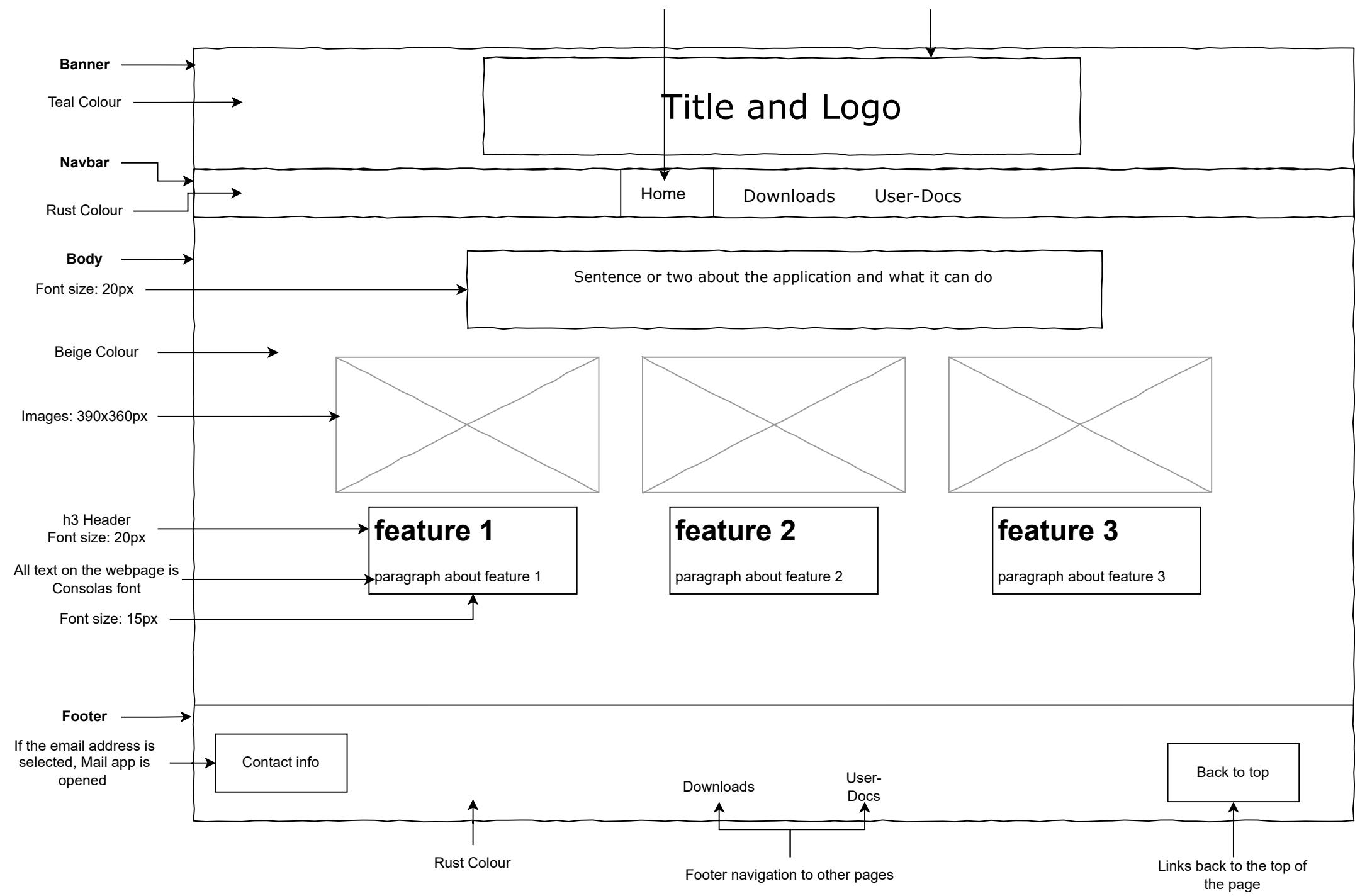
-----
- onlyNums(CString str): BOOL
- brkPosContains(std::vector<std::vector<int>> vector, int value, int side = 1): BOOL
  
```

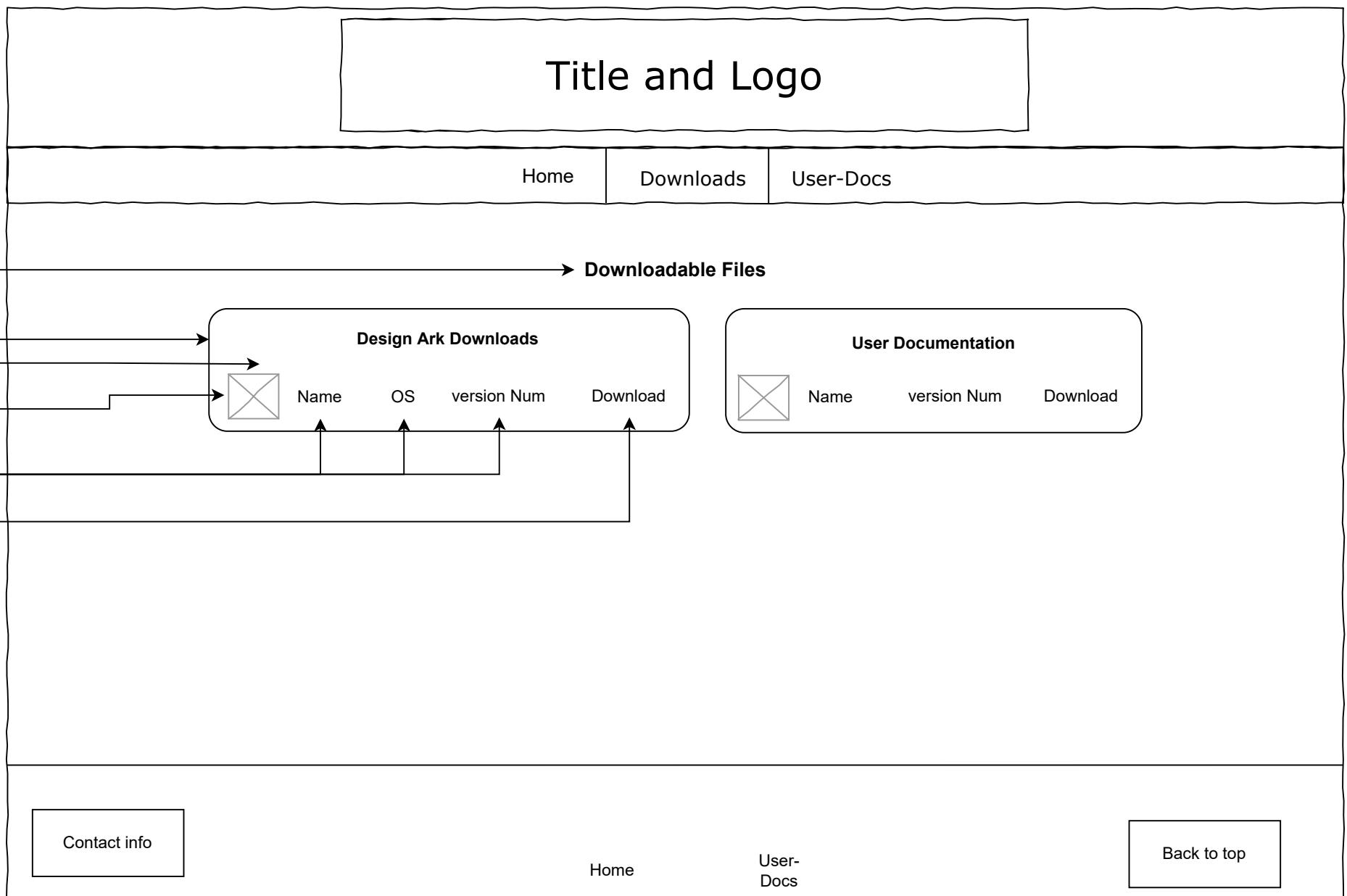
```
1 // To meet the SQA Advanced Higher Computing Science criteria,
2 // the program implements a insertion sort in all
3 // CSmartColour's property pages when closing them
4
5 // The following pseudocode is a design for CSmartColourTypes' sort
6 // It will show how it works in practise for the rest of the property
7 // sheets
8
9 FOR i IN RANGE(1, m_Listbox_Types_control.SizeOf()):
10
11     CREATE variables
12
13     WHILE pos > 0 AND m_Listbox_Types_control[pos-1] > currentValue:
14         SET m_Listbox_Types_control[pos] TO m_Listbox_Types_control[po
15         s - 1]
16         pos -= 1
17
18     m_Listbox_Types_control = currentValue
19
20
21.1 DECLARE CString currentValue INITIALLY AS m_Listbox_Types_control[i]
21.2 DECLARE INT pos INITIALLY AS i
```

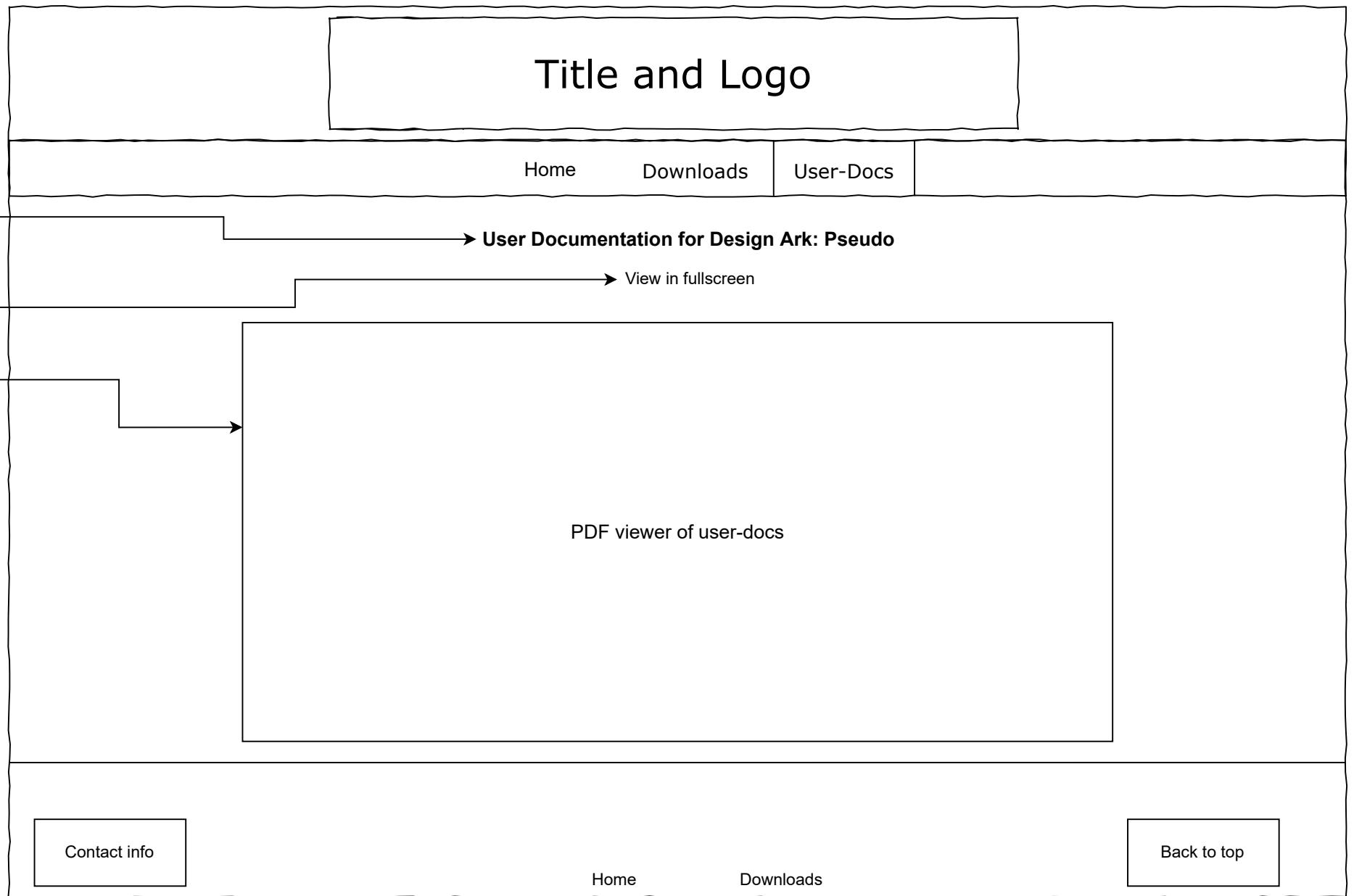


Current page is selected

Image: 530x91px







```
1 // DesignArk.h : main header file for the DesignArk application
2 //
3 #pragma once
4
5 #ifndef __AFXWIN_H__
6     #error "include 'pch.h' before including this file for PCH"
7 #endif
8
9
10 #include "resource.h"           // main symbols
11
12 /******
13 To find the number of lines in project, run
14
15 (gci -include *.h,*.cpp -recurse | select-string .).Count
16
17 in the code folder
18
19 (last count: 5298)
20
21 *****/
22
23
24
25 // CDesignArkApp:
26 // See DesignArk.cpp for the implementation of this class
27 //
28
29 class CDesignArkApp : public CWinAppEx
30 {
31 public:
32     // Public Constructors
33     CDesignArkApp() noexcept;
34     ~CDesignArkApp();
35
36     // Public Overrides
37     virtual BOOL InitInstance();
38     virtual int ExitInstance();
39
40     // Public Commands
41     afx_msg void OnSmartColour();
42
43     afx_msg void OnAppAbout();
44     afx_msg void OnOpenSite();
45     afx_msg void OnUserdocs();
46
47     // Public Implementations
48     UINT m_nAppLook;
49     BOOL m_bHiColorIcons;
50
51     CView* m_ActiveView;
52
53     // Public Resources
54     // SmartColour Preferences
55     std::vector<std::vector<CString>> smartColour_String;
56     std::vector<COLORREF> smartColour_Colour;
```

```
57
58     CString commentType;
59     COLORREF commentColour;
60     COLORREF numberColour;
61     COLORREF highlightColour;
62
63     // App work preferences
64     float zoom;
65     CString sFont;
66     int indentSize;
67
68 protected:
69     // Protected Implementation
70     virtual void PreLoadState();
71     virtual void LoadCustomState();
72     virtual void SaveCustomState();
73
74     DECLARE_MESSAGE_MAP()
75 };
76
77 extern CDesignArkApp theApp;
78
```

```
1 // DesignArk.cpp : Defines the class behaviors for the application.
2 //
3 //
4
5 #include "pch.h"
6 #include "framework.h"
7 #include "afxwinappex.h"
8 #include "afxdialogex.h"
9 #include "DesignArk.h"
10 #include "MainFrm.h"
11
12 #include "ChildFrm.h"
13 #include "CTextDocument.h"
14 #include "CTextDocView.h"
15
16 #include "CAboutDlg.h"
17 #include "CSmartColour.h"
18
19 #ifdef _DEBUG
20 #define new DEBUG_NEW
21#endif
22
23
24 // CDesignArkApp
25
26 BEGIN_MESSAGE_MAP(CDesignArkApp, CWinAppEx)
27     ON_COMMAND(ID_VIEW_SMARTCOLOURWINDOW,
28                 &CDesignArkApp::OnSmartColour) →
29     ON_COMMAND(ID_HELP_OPENWEBSITE, &CDesignArkApp::OnOpenSite) →
30     ON_COMMAND(ID_HELP_VIEWUSERDOCSONLINE,
31                 &CDesignArkApp::OnUserdocs) →
32     ON_COMMAND(ID_APP_ABOUT, &CDesignArkApp::OnAppAbout)
33
34     // Standard file based document commands
35     ON_COMMAND(ID_FILE_NEW, &CWinAppEx::OnFileNew)
36     ON_COMMAND(ID_FILE_OPEN, &CWinAppEx::OnFileOpen)
37     // Standard print setup command
38     ON_COMMAND(ID_FILE_PRINT_SETUP, &CWinAppEx::OnFilePrintSetup)
39
40 END_MESSAGE_MAP()
41
42 // CDesignArkApp construction
43 CDesignArkApp::CDesignArkApp() noexcept
44 {
45     m_bHiColorIcons = TRUE;
46     m_nAppLook = 0;
47
48     // support Restart Manager
49     m_dwRestartManagerSupportFlags =
50         AFX_RESTART_MANAGER_SUPPORT_ALL_ASPECTS; →
51 #ifdef _MANAGED
52     // If the application is built using Common Language Runtime →
53     // support (/clr):
54     //      1) This additional setting is needed for Restart Manager →
55     //          support to work properly.
56     //      2) In your project, you must add a reference to →
57 }
```

```
System.Windows.Forms in order to build.  
52     System::Windows::Forms::Application::SetUnhandledExceptionMode  ↪  
      (System::Windows::Forms::UnhandledExceptionMode::ThrowException  
n);  
53 #endif  
54  
55     SetAppID(_T("DesignArk.Verison.1.0"));  
56  
57     // TODO: add construction code here,  
58     // Place all significant initialization in InitInstance  
59     this->smartColour_String.resize(5);  
60     this->smartColour_Colour.resize(5);  
61  
62     this->smartColour_String[0] = std::vector<CString>{  
63         L"ELIF",  
64         L"ELSE",  
65         L"FOR",  
66         L"IF",  
67         L"IN",  
68         L"RETURN",  
69         L"WHILE"  
70     };  
71     this->smartColour_Colour[0] = COLORREF(RGB(128, 0, 128));  
72  
73     this->smartColour_String[1] = std::vector<CString>{  
74         L"AS",  
75         L"CREATE",  
76         L"DECLARE",  
77         L"DISPLAY",  
78         L"DO",  
79         L"FUNCTION",  
80         L"INITIALISE",  
81         L"INITIALLY",  
82         L"PRINT",  
83         L"PROCEDURE",  
84         L"SET",  
85         L"TO",  
86     };  
87     this->smartColour_Colour[1] = COLORREF(RGB(128, 0, 32));  
88  
89     this->smartColour_String[2] = std::vector<CString>{  
90         L"ARRAY",  
91         L"BOOLEAN",  
92         L"CLASS",  
93         L"CONST",  
94         L"DICTIONARY",  
95         L"FLOAT",  
96         L"INT",  
97         L"LIST",  
98         L"OVERLOAD"  
99         L"REAL",  
100        L"STRING",  
101        L"VIRTUAL",  
102        L"VOID"  
103    };  
104    this->smartColour_Colour[2] = COLORREF(RGB(0, 0, 255));
```

```
105
106     this->smartColour_String[3] = std::vector<CString>{
107         L"vector"
108     };
109     this->smartColour_Colour[3] = COLORREF(RGB(0, 255, 0));
110
111     this->smartColour_String[4] = std::vector<CString>{
112         L"RANDOM"
113     };
114     this->smartColour_Colour[4] = COLORREF(RGB(255, 0, 0));
115
116     this->commentType = L"//";
117     this->commentColour = COLORREF(RGB(255, 0, 0));
118
119     this->numberColour = COLORREF(RGB(0, 255, 0));
120     this->highlightColour = COLORREF(RGB(110, 192, 250));
121
122     this->zoom = 1.f;
123     this->sFont = L"Courier New";
124     this->indentSize = 4;
125
126     // Initialise the GDI stuff
127     Gdiplus::GdiplusStartupInput gdiplusStartupInput;
128     ULONG_PTR gdiplusToken;
129
130     Gdiplus::GdiplusStartup(&gdiplusToken, &gdiplusStartupInput,    ↪
131     NULL);
132 CDesignArkApp::~CDesignArkApp()
133 {
134 }
135
136 // The one and only CDesignArkApp object
137 CDesignArkApp theApp;
138
139 // CDesignArkApp initialisation
140 BOOL CDesignArkApp::InitInstance()
141 {
142     /*/////////////////////////////
143     AllocConsole();
144     FILE* fDummy;
145     freopen_s(&fDummy, "CONOUT$", "w", stdout);
146     freopen_s(&fDummy, "CONOUT$", "w", stderr);
147     freopen_s(&fDummy, "CONIN$", "r", stdin); // Console can be used
148     be debugging
149     //////////////////////////////*/
150
151     // Initialise the common controls
152     INITCOMMONCONTROLSEX InitCtrls;
153     InitCtrls.dwSize = sizeof(InitCtrls);
154     InitCtrls.dwICC = ICC_WIN95_CLASSES;
155     InitCommonControlsEx(&InitCtrls);
156
157     CWinAppEx::InitInstance();
158
159     // Initialize OLE libraries
```

```
159     if (!AfxOleInit())
160     {
161         AfxMessageBox(IDP_OLE_INIT_FAILED);
162         return FALSE;
163     }
164
165     AfxEnableControlContainer();
166     EnableTaskbarInteraction();
167
168     AfxInitRichEdit2(); // is required to use RichEdit control
169
170     // Register app with OS and load INI file options (including ↵
171     // MRU)
171     SetRegistryKey(_T("DesignArk"));
172     LoadStdProfileSettings(4); // Load standard INI file options ↵
172     (including MRU)
173
174     // Initialise managers
175     InitContextMenuManager();
176     InitKeyboardManager();
177     InitTooltipManager();
178     CMFCToolTipInfo ttParams;
179     ttParams.m_bVislManagerTheme = TRUE;
180     theApp.GetTooltipManager()->SetTooltipParams
180     (AFX_TOOLTIP_TYPE_ALL,
181      RUNTIME_CLASS(CMFCToolTipCtrl), &ttParams);
182
183     // Initialise document template
184     CMultiDocTemplate* pDocTemplate;
185     pDocTemplate = new CMultiDocTemplate(IDR_DesignArkTYPE,
186         RUNTIME_CLASS(CTextDocument),
187         RUNTIME_CLASS(CChildFrame), // custom MDI child frame
188         RUNTIME_CLASS(CTextDocView));
189     if (!pDocTemplate)
190         return FALSE;
191     this->AddDocTemplate(pDocTemplate);
192
193     // create main MDI Frame window
194     CMainFrame* pMainFrame = new CMainFrame;
195     if (!pMainFrame || !pMainFrame->LoadFrame(IDR_MAINFRAME))
196     {
197         delete pMainFrame;
198         return FALSE;
199     }
200     m_pMainWnd = pMainFrame;
201
202     // Parse command line for standard shell commands, DDE, file ↵
202     // open
203     CCommandLineInfo cmdInfo;
204     ParseCommandLine(cmdInfo);
205
206     // Dispatch commands specified on the command line. Will return ↵
206     // FALSE if
207     // app was launched with /RegServer, /Register, /Unregserver ↵
207     // or /Unregister.
208     if (!ProcessShellCommand(cmdInfo))
```

```
209         return FALSE;
210
211     // The main window has been initialized, so show and update it
212
213     //CRect client;
214     //pMainFrame->GetWindowRect(client);
215     //pMainFrame->MoveWindow(client.left, client.top, 750, 900);
216
217     pMainFrame->>ShowWindow(m_nCmdShow);
218     pMainFrame->UpdateWindow();
219
220     return TRUE;
221 }
222 int CDesignArkApp::ExitInstance()
223 {
224     //TODO: handle additional resources you may have added
225     AfxOleTerm(FALSE);
226
227     return CWinAppEx::ExitInstance();
228 }
229
230 // Public Commands
231 void CDesignArkApp::OnSmartColour()
232 {
233     CString tempIndentSize;
234     tempIndentSize.Format(L"%d", this->indentSize);
235     CSmartColour smartColour(this->smartColour_String, this-↗
236     >smartColour_Colour, this->commentType, this->commentColour, ↗
237     this->numberColour, this->highlightColour, this->sFont, ↗
238     tempIndentSize);
239
240     if (smartColour.DoModal() == IDOK) {
241
242         this->smartColour_String[0] = ((CSmartColourStatements*)↗
243             smartColour.GetPage(0))->m_Strings_Statements_syntax_var;
244         this->smartColour_Colour[0] = ((CSmartColourStatements*)↗
245             smartColour.GetPage(0))->m_Colour_Statements_syntax_var;
246
247         this->smartColour_String[1] = ((CSmartColourStatements*)↗
248             smartColour.GetPage(0))-↗
249             >m_Strings_Statements_statements_var;
250         this->smartColour_Colour[1] = ((CSmartColourStatements*)↗
251             smartColour.GetPage(0))-↗
252             >m_Colour_Statements_statements_var;
253
254         this->smartColour_String[2] = ((CSmartColourTypes*)↗
255             smartColour.GetPage(1))->m_listbox_types_strings;
256         this->smartColour_Colour[2] = ((CSmartColourTypes*)↗
257             smartColour.GetPage(1))->m_Combo_Types_colour_var;
258
259         this->smartColour_String[3] = ((CSmartColourBuiltin*)↗
260             smartColour.GetPage(2))->m_Listbox_Builtin_Classes_var;
261         this->smartColour_Colour[3] = ((CSmartColourBuiltin*)↗
262             smartColour.GetPage(2))-↗
263             >m_Combo_Builtin_Classes_colour_var;
```

```

...mes\source\repos\DesignArk\DesignArk\DesignArk.cpp 6
251     this->smartColour_String[4] = ((CSmartColourBuiltin*)    ↵
252         smartColour.GetPage(2))->m_Listbox_Builtin_Functions_var;
253     this->smartColour_Colour[4] = ((CSmartColourBuiltin*)    ↵
254         smartColour.GetPage(2))-    ↵
255             >m_Combo_Builtin_Functions_colour_var;
256
257     this->commentType = ((CSmartColourOther*) smartColour.GetPage    ↵
258         (3))->m_pCombo_comments_type_var;
259     this->commentColour = ((CSmartColourOther*)    ↵
260         smartColour.GetPage(3))->m_pCombo_comments_colour_var;
261
262     this->numberColour = ((CSmartColourOther*)    ↵
263         smartColour.GetPage(3))->m_pCombo_numbers_colour_var;
264
265     this->highlightColour = ((CSmartColourOther*)    ↵
266         smartColour.GetPage(3))->m_pCombo_highlight_colour_var;
267
268     this->sFont = ((CSmartColourOther*) smartColour.GetPage(3))-    ↵
269         >m_pCombo_fonts_var;
270
271     this->indentSize = _wtoi(((CSmartColourOther*)    ↵
272         smartColour.GetPage(3))->m_pCombo_indent_var);
273
274     // Will need to fixed if more views/docs are implemeted but    ↵
275         will do for v1.0
276     if (this->m_ActiveView) {
277         ((CTextDocView*)this->m_ActiveView)->refresh();
278     }
279 }
280
281 void CDesignArkApp::OnAppAbout() // This bit is not working
282 {
283     CAaboutDlg aboutDlg;
284     aboutDlg.DoModal(); // This line is triggering a debug assertion    ↵
285         failure
286 }
287 void CDesignArkApp::OnOpenSite()
288 {
289     ShellExecute(0, 0, L"http://localhost/DesignArk/", 0, 0,    ↵
290         SW_SHOW);
291 }
292 void CDesignArkApp::OnUserdocs()
293 {
294     ShellExecute(0, L"open", L"http://localhost/DesignArk/    ↵
295         userdocs.php", 0, 0, SW_SHOW);
296 }
297
298 // Protected Implementation
299 void CDesignArkApp::PreLoadState()
300 {
301     BOOL bNameValid;
302     CString strName;
303     bNameValid = strName.LoadString(IDS_EDIT_MENU);
304     ASSERT(bNameValid);
305     GetContextMenuManager()->AddMenu(strName, IDR_POPUP_EDIT);

```

```
294     bNameValid = strName.LoadString(IDS_EXPLORER);  
295     ASSERT(bNameValid);  
296     GetContextMenuManager() ->AddMenu(strName, IDR_POPUP_EXPLORER);  
297 }  
298 void CDesignArkApp::LoadCustomState()  
299 {  
300 }  
301 void CDesignArkApp::SaveCustomState()  
302 {  
303 }  
304  
305
```

```
1
2 // MainFrm.h : interface of the CMainFrame class
3 //
4
5 #pragma once
6
7 class CMainFrame : public CMDIFrameWndEx
8 {
9 public:
10     // Public Constructors
11     CMainFrame() noexcept;
12     virtual ~CMainFrame();
13
14     //Public Overrides
15     virtual BOOL PreCreateWindow(CREATESTRUCT& cs);
16     virtual BOOL LoadFrame(UINT nIDResource, DWORD dwDefaultStyle = ↵
17         WS_OVERLAPPEDWINDOW | FWS_ADDTOTITLE, CWnd* pParentWnd = ↵
18         nullptr, CCreateContext* pContext = nullptr);
19
20     // Public Implementations
21     DECLARE_DYNAMIC(CMainFrame)
22 #ifdef _DEBUG
23     virtual void AssertValid() const;
24     virtual void Dump(CDumpContext& dc) const;
25 #endif
26
27 protected:
28     // Protected Implementations
29     DECLARE_MESSAGE_MAP()
30
31     // Protected Resources
32     CMFCMenuBar      m_wndMenuBar;
33     CMFCToolBar      m_wndToolBar;
34     CMFCToolBarImages m_UserImages;
35
36     // Protected Message Handlers
37     afx_msg int OnCreate(LPCREATESTRUCT lpCreateStruct);
38     afx_msg void On WindowManager();
39     afx_msg void On ViewCustomize();
40     afx_msg LRESULT On ToolbarCreateNew(WPARAM wp, LPARAM lp);
41     afx_msg void On ApplicationLock(UINT id);
42     afx_msg void On UpdateApplicationLook(CCmdUI* pCmdUI);
43     virtual void On SetActiveView(CView* pViewNew, BOOL bNotify = TRUE);
44
45 private:
46     // Private Initialisers
47     int init_tabs();
48     int init_menubar();
49     int init_toolbar();
50     int init_statusbar();
51     int init_dockingwindows();
52     int init_customisation();
53     int init_userimages();
54     int init_commands();
```

54 };

55

```
1 // MainFrm.cpp : implementation of the CMainFrame class
2 //
3
4 #include "pch.h"
5 #include "framework.h"
6 #include "DesignArk.h"
7
8 #include "MainFrm.h"
9
10 #ifdef _DEBUG
11 #define new DEBUG_NEW
12 #endif
13
14 // CMainFrame
15
16
17 IMPLEMENT_DYNAMIC(CMainFrame, CMDIFrameWndEx)
18
19 const int iMaxUserToolbars = 10;
20 const UINT uiFirstUserToolBarId = AFX_IDW_CONTROLBAR_FIRST + 40;
21 const UINT uiLastUserToolBarId = uiFirstUserToolBarId +
22     iMaxUserToolbars - 1;
23
24 BEGIN_MESSAGE_MAP(CMainFrame, CMDIFrameWndEx)
25     ON_WM_CREATE()
26     ON_COMMAND(ID_WINDOW_MANAGER, &CMainFrame::On.WindowManager)
27     ON_COMMAND(ID_VIEW_CUSTOMIZE, &CMainFrame::On.ViewCustomize)
28     ON_REGISTERED_MESSAGE(AFX_WM_CREATETOOLBAR,
29         &CMainFrame::On.ToolbarCreateNew)
30     ON_COMMAND_RANGE(ID_VIEW_APPLOOK_WIN_2000,
31         ID_VIEW_APPLOOK_WINDOWS_7, &CMainFrame::On.ApplicationLook)
32     ON_UPDATE_COMMAND_UI_RANGE(ID_VIEW_APPLOOK_WIN_2000,
33         ID_VIEW_APPLOOK_WINDOWS_7,
34         &CMainFrame::On.UpdateApplicationLook)
35
36 END_MESSAGE_MAP()
37
38 static UINT indicators[] =
39 {
40     ID_SEPARATOR,           // status line indicator
41     ID_INDICATOR_CAPS,
42     ID_INDICATOR_NUM
43 };
44
45 // CMainFrame construction/destruction
46
47 CMainFrame::CMainFrame() noexcept
48 {
49     // TODO: add member initialization code here
50     theApp.m_nAppLook = theApp.GetInt(_T("ApplicationLook"),
51         ID_VIEW_APPLOOK_VS_2008);
52 }
53 CMainFrame::~CMainFrame()
54 {
55 }
56
57 BOOL CMainFrame::PreCreateWindow(CREATESTRUCT& cs)
```

```
51  {
52      if( !CMDIFrameWndEx::PreCreateWindow(cs) )
53          return FALSE;
54      // TODO: Modify the Window class or styles here by modifying
55      // the CREATESTRUCT cs
56
57      return TRUE;
58  }
59 BOOL CMainFrame::LoadFrame(UINT nIDResource, DWORD dwDefaultStyle, →
60     CWnd* pParentWnd, CCreateContext* pContext)
61 {
62     // base class does the real work
63
64     if( !CMDIFrameWndEx::LoadFrame(nIDResource, dwDefaultStyle, →
65         pParentWnd, pContext) )
66     {
67         return FALSE;
68     }
69
70     // enable customization button for all user toolbars
71     BOOL bNameValid;
72     CString strCustomize;
73     bNameValid = strCustomize.LoadString(IDS_TOOLBAR_CUSTOMIZE);
74     ASSERT(bNameValid);
75
76     for (int i = 0; i < iMaxUserToolbars; i++)
77     {
78         CMFCToolBar* pUserToolbar = GetUserToolBarByIndex(i);
79         if (pUserToolbar != nullptr)
80         {
81             pUserToolbar->EnableCustomizeButton(TRUE,
82                     ID_VIEW_CUSTOMIZE, strCustomize);
83         }
84     }
85
86
87 #ifdef _DEBUG
88 void CMainFrame::AssertValid() const
89 {
90     CMDIFrameWndEx::AssertValid();
91 }
92 void CMainFrame::Dump(CDumpContext& dc) const
93 {
94     CMDIFrameWndEx::Dump(dc);
95 }
96 #endif // _DEBUG
97
98 int CMainFrame::OnCreate(LPCREATESTRUCT lpCreateStruct)
99 {
100     if (CMDIFrameWndEx::OnCreate(lpCreateStruct) == -1)
101         return -1;
102
103     BOOL bNameValid;
```

```
104     CString strCustomize;
105
106     init_tabs();
107     init_menubar();
108     init_toolbar();
109     init_statusbar();
110     init_dockingwindows();
111     init_customisation();
112     init_userimages();
113     init_commands();
114
115     return 0;
116 }
117 void CMainFrame::On.WindowManager()
118 {
119     ShowWindowsDialog();
120 }
121 void CMainFrame::OnViewCustomize()
122 {
123     CMFCToolsCustomizeDialog* pDlgCust = new
124         CMFCToolsCustomizeDialog(this, TRUE /* scan menus */);
125     pDlgCust->EnableUserDefinedToolbars();
126     pDlgCust->Create();
127 }
128 LRESULT CMainFrame::OnToolbarCreateNew(WPARAM wp, LPARAM lp)
129 {
130     LRESULT lres = CMDIFrameWndEx::OnToolbarCreateNew(wp, lp);
131     if (lres == 0)
132     {
133         return 0;
134     }
135     CMFCToolBar* pUserToolbar = (CMFCToolBar*)lres;
136     ASSERT_VALID(pUserToolbar);
137
138     BOOL bNameValid;
139     CString strCustomize;
140     bNameValid = strCustomize.LoadString(IDS_TOOLBAR_CUSTOMIZE);
141     ASSERT(bNameValid);
142
143     pUserToolbar->EnableCustomizeButton(TRUE, ID_VIEW_CUSTOMIZE,
144                                         strCustomize);
145     return lres;
146 }
147 void CMainFrame::OnApplicationLook(UINT id)
148 {
149     CWaitCursor wait;
150     theApp.m_nAppLook = id;
151
152     switch (theApp.m_nAppLook)
153     {
154     case ID_VIEW_APPLOOK_WIN_2000:
155         CMFCVisualManager::SetDefaultManager(RUNTIME_CLASS
156             (CMFCVisualManager));
157         break;
```

```
157
158     case ID_VIEW_APPLOOK_OFF_XP:
159         CMFCVisualManager::SetDefaultManager(RUNTIME_CLASS
160             (CMFCVisualManagerOfficeXP));
161         break;
162
163     case ID_VIEW_APPLOOK_WIN_XP:
164         CMFCVisualManagerWindows::m_b3DTabsXPTHEME = TRUE;
165         CMFCVisualManager::SetDefaultManager(RUNTIME_CLASS
166             (CMFCVisualManagerWindows));
167         break;
168
169     case ID_VIEW_APPLOOK_OFF_2003:
170         CMFCVisualManager::SetDefaultManager(RUNTIME_CLASS
171             (CMFCVisualManagerOffice2003));
172         CDockingManager::SetDockingMode(DT_SMART);
173         break;
174
175     case ID_VIEW_APPLOOK_VS_2005:
176         CMFCVisualManager::SetDefaultManager(RUNTIME_CLASS
177             (CMFCVisualManagerVS2005));
178         CDockingManager::SetDockingMode(DT_SMART);
179         break;
180
181     case ID_VIEW_APPLOOK_VS_2008:
182         CMFCVisualManager::SetDefaultManager(RUNTIME_CLASS
183             (CMFCVisualManagerVS2008));
184         CDockingManager::SetDockingMode(DT_SMART);
185         break;
186
187     default:
188         switch (theApp.m_nAppLook)
189         {
190             case ID_VIEW_APPLOOK_OFF_2007_BLUE:
191                 CMFCVisualManagerOffice2007::SetStyle
192                     (CMFCVisualManagerOffice2007::Office2007_LunaBlue);
193                 break;
194
195             case ID_VIEW_APPLOOK_OFF_2007_BLACK:
196                 CMFCVisualManagerOffice2007::SetStyle
197                     (CMFCVisualManagerOffice2007::Office2007_ObsidianBlack);
198                 break;
199
200             case ID_VIEW_APPLOOK_OFF_2007_SILVER:
201                 CMFCVisualManagerOffice2007::SetStyle
202                     (CMFCVisualManagerOffice2007::Office2007_Silver);
203                 break;
204
205             case ID_VIEW_APPLOOK_OFF_2007_AQUA:
```

```
203         CMFCVisualManagerOffice2007::SetStyle  
204             (CMFCVisualManagerOffice2007::Office2007_Aqua);  
205     break;  
206 }  
207 CMFCVisualManager::SetDefaultManager(RUNTIME_CLASS  
208     (CMFCVisualManagerOffice2007));  
209 CDockingManager::SetDockingMode(DT_SMART);  
210 }  
211 RedrawWindow(nullptr, nullptr, RDW_ALLCHILDREN | RDW_INVALIDATE  
212     | RDW_UPDATENOW | RDW_FRAME | RDW_ERASE);  
213 theApp.WriteInt(_T("ApplicationLook"), theApp.m_nAppLook);  
214 }  
215 void CMainFrame::OnUpdateApplicationLook(CCmdUI* pCmdUI)  
216 {  
217     pCmdUI->SetRadio(theApp.m_nAppLook == pCmdUI->m_nID);  
218 }  
219  
220 void CMainFrame::OnSetActiveView(CView* pViewNew, BOOL bNotify)  
221 {  
222     CFrameWnd::SetActiveView(pViewNew, bNotify);  
223 }  
224  
225 // Private Initialisers  
226 int CMainFrame::init_tabs()  
227 {  
228     CMDITabInfo mdiTabParams;  
229     mdiTabParams.m_style = CMFCTabCtrl::STYLE_3D_ONENOTE;  
230     mdiTabParams.m_bActiveTabCloseButton = TRUE;  
231     mdiTabParams.m_bDocumentMenu = TRUE;  
232     mdiTabParams.m_bEnableTabSwap = TRUE;  
233     mdiTabParams.m_bFlatFrame = TRUE;  
234     mdiTabParams.m_bTabCloseButton = TRUE;  
235     mdiTabParams.m_bTabCustomTooltips = FALSE;  
236     mdiTabParams.m_bTabIcons = FALSE;  
237     //mdiTabParams.m_nTabBorderSize =  
238         CMFCVisualManager::GetMDITabsBordersSize;  
239     mdiTabParams.m_tabLocation =  
240         CMFCBaseTabCtrl::Location::LOCATION_TOP;  
241     this->EnableMDITabbedGroups(TRUE, mdiTabParams);  
242     return 0;  
243 }  
244 int CMainFrame::init_menubar()  
245 {  
246     if (!this->m_wndMenuBar.Create(this))  
247     {  
248         TRACE0("Failed to create menubar\n");  
249         return -1; // fail to create  
250     }  
251     this->m_wndMenuBar.SetPaneStyle(this->m_wndMenuBar.GetPaneStyle  
252         () | CBRS_SIZE_DYNAMIC | CBRS_TOOLTIPS | CBRS_FLYBY);
```

```
253     // prevent the menu bar from taking the focus on activation
254     CMFCPopupMenu::SetForceMenuFocus(FALSE);
255
256     return 0;
257 }
258 int CMainFrame::init_toolbar()
259 {
260     BOOL bNameValid;
261     if (!m_wndToolBar.CreateEx(this, TBSTYLE_FLAT, WS_CHILD |      ↵
262         WS_VISIBLE | CBRS_TOP | CBRS_GRIPPER | CBRS_TOOLTIPS |      ↵
263         CBRS_FLYBY | CBRS_SIZE_DYNAMIC) ||
264         !m_wndToolBar.LoadToolBar(theApp.m_bHiColorIcons ?      ↵
265             IDR_MAINFRAME_256 : IDR_MAINFRAME))
266     {
267         TRACE0("Failed to create toolbar\n");
268         return -1;           // fail to create
269     }
270
271     CString strToolBarName;
272     bNameValid = strToolBarName.LoadString(IDS_TOOLBAR_STANDARD);
273     ASSERT(bNameValid);
274     m_wndToolBar.SetWindowText(strToolBarName);
275
276     CString strCustomize;
277     bNameValid = strCustomize.LoadString(IDS_TOOLBAR_CUSTOMIZE);
278     ASSERT(bNameValid);
279     m_wndToolBar.EnableCustomizeButton(TRUE, ID_VIEW_CUSTOMIZE,      ↵
280         strCustomize);
281
282     // Allow user-defined toolbars operations:
283     InitUserToolbars(nullptr, uiFirstUserToolBarId,      ↵
284         uiLastUserToolBarId);
285
286     return 0;
287 }
288 int CMainFrame::init_statusbar()
289 {
290     if (!this->m_wndStatusBar.Create(this))
291     {
292         TRACE0("Failed to create status bar\n");
293         return -1;           // fail to create
294     }
295     this->m_wndStatusBar.SetIndicators(indicators, sizeof      ↵
296         (indicators) / sizeof(UINT));
297
298     return 0;
299 }
300 int CMainFrame::init_dockingwindows()
301 {
302     // TODO: Delete these five lines if you don't want the toolbar      ↵
303         and menubar to be dockable
304     this->DockPane(&this->m_wndMenuBar);
305     this->DockPane(&this->m_wndToolBar);
306     //this->m_wndMenuBar.EnableDocking(CBRS_ALIGN_ANY);
307     //this->m_wndToolBar.EnableDocking(CBRS_ALIGN_ANY);
308     //this->EnableDocking(CBRS_ALIGN_ANY);
```

```
302
303     // enable Visual Studio 2005 style docking window behavior
304     CDockingManager::SetDockingMode(DT_SMART);
305     // enable Visual Studio 2005 style docking window auto-hide      ↵
306     // behavior
307     this->EnableAutoHidePanes(CBRS_ALIGN_ANY);
308
309     return 0;
310 }
311 int CMainFrame::init_customisation()
312 {
313     BOOL bNameValid;
314     CString strCustomize;
315     bNameValid = strCustomize.LoadString(IDS_TOOLBAR_CUSTOMIZE);
316     ASSERT(bNameValid);
317
318     // set the visual manager and style based on persisted value
319     this->OnApplicationLook(theApp.m_nAppLook);
320
321     // Enable enhanced windows management dialog
322     this->EnableWindowsDialog(ID_WINDOW_MANAGER, ID_WINDOW_MANAGER,    ↵
323                               TRUE);
324
325     // Enable toolbar and docking window menu replacement
326     this->EnablePaneMenu(TRUE, ID_VIEW_CUSTOMIZE, strCustomize,        ↵
327                           ID_VIEW_TOOLBAR);
328
329     // enable quick (Alt+drag) toolbar customization
330     CMFCToolBar::EnableQuickCustomization();
331
332     // Switch the order of document name and application name on the    ↵
333     // window title bar. This
334     // improves the usability of the taskbar because the document    ↵
335     // name is visible with the thumbnail.
336     this->ModifyStyle(0, FWS_PREFIXTITLE);
337
338     return 0;
339 }
340 int CMainFrame::init_userimages()
341 {
342     // Load menu item image (not placed on any standard toolbars):
343     CMFCToolBar::AddToolBarForImageCollection(IDR_MENU_IMAGES,       ↵
344         theApp.m_bHiColorIcons ? IDR_MENU_IMAGES_24 : 0);
345
346     if (CMFCToolBar:: GetUserImages() == nullptr)
347     {
348         // load user-defined toolbar images
349         if (this->m_UserImages.Load(_T(".\\UserImages.bmp")))
350         {
351             CMFCToolBar::SetUserImages(&m_UserImages);
352         }
353     }
354
355     return 0;
356 }
357 int CMainFrame::init_commands()
```

```
352 {  
353     CList<UINT, UINT> lstBasicCommands;  
354  
355     lstBasicCommands.AddTail(ID_FILE_NEW);  
356     lstBasicCommands.AddTail(ID_FILE_OPEN);  
357     lstBasicCommands.AddTail(ID_FILE_SAVE);  
358     lstBasicCommands.AddTail(ID_FILE_PRINT);  
359     lstBasicCommands.AddTail(ID_APP_EXIT);  
360     lstBasicCommands.AddTail(ID_EDIT_CUT);  
361     lstBasicCommands.AddTail(ID_EDIT_PASTE);  
362     lstBasicCommands.AddTail(ID_EDIT_UNDO);  
363     lstBasicCommands.AddTail(ID_APP_ABOUT);  
364     lstBasicCommands.AddTail(ID_VIEW_STATUS_BAR);  
365     lstBasicCommands.AddTail(ID_VIEW_TOOLBAR);  
366     lstBasicCommands.AddTail(ID_VIEW_APPLOOK_OFF_2003);  
367     lstBasicCommands.AddTail(ID_VIEW_APPLOOK_VS_2005);  
368     lstBasicCommands.AddTail(ID_VIEW_APPLOOK_OFF_2007_BLUE);  
369     lstBasicCommands.AddTail(ID_VIEW_APPLOOK_OFF_2007_SILVER);  
370     lstBasicCommands.AddTail(ID_VIEW_APPLOOK_OFF_2007_BLACK);  
371     lstBasicCommands.AddTail(ID_VIEW_APPLOOK_OFF_2007_AQUA);  
372     lstBasicCommands.AddTail(ID_VIEW_APPLOOK_WINDOWS_7);  
373     lstBasicCommands.AddTail(ID_SORTING_SORTALPHABETIC);  
374     lstBasicCommands.AddTail(ID_SORTING_SORTBYTYPE);  
375     lstBasicCommands.AddTail(ID_SORTING_SORTBYACCESS);  
376     lstBasicCommands.AddTail(ID_SORTING_GROUPBYTYPE);  
377  
378     CMFCToolBar::SetBasicCommands(lstBasicCommands);  
379  
380     return 0;  
381 }  
382
```

```
1 // ChildFrm.h : interface of the CChildFrame class
2 //
3 //
4
5 #pragma once
6
7 class CChildFrame : public CMDIChildWndEx
8 {
9 public:
10     // Implementation
11     CChildFrame() noexcept;
12     virtual ~CChildFrame();
13     DECLARE_DYNCREATE(CChildFrame)
14
15     // Overrides
16     virtual BOOL PreCreateWindow(CREATESTRUCT& cs);
17
18 #ifdef _DEBUG
19     virtual void AssertValid() const;
20     virtual void Dump(CDumpContext& dc) const;
21 #endif
22 protected:
23     // Resources
24     CSplitterWndEx m_wndSplitter;
25
26     // Generated message map functions
27     DECLARE_MESSAGE_MAP()
28 };
```

```
1 // ChildFrm.cpp : implementation of the CChildFrame class
2 //
3 //
4 #include "pch.h"
5 #include "framework.h"
6 #include "DesignArk.h"
7
8 #include "ChildFrm.h"
9
10 #ifdef _DEBUG
11 #define new DEBUG_NEW
12 #endif
13
14 // CChildFrame
15
16
17 IMPLEMENT_DYNCREATE(CChildFrame, CMDIChildWndEx)
18
19 BEGIN_MESSAGE_MAP(CChildFrame, CMDIChildWndEx)
20 END_MESSAGE_MAP()
21
22 // CChildFrame construction/destruction
23
24 CChildFrame::CChildFrame() noexcept
25 {
26     // TODO: add member initialization code here
27 }
28
29 CChildFrame::~CChildFrame()
30 {
31 }
32
33
34 BOOL CChildFrame::PreCreateWindow(CREATESTRUCT& cs)
35 {
36     // TODO: Modify the Window class or styles here by modifying the →
37     // CREATESTRUCT cs
38     if( !CMDIChildWndEx::PreCreateWindow(cs) )
39         return FALSE;
40
41     return TRUE;
42 }
43
44 // CChildFrame diagnostics
45
46 #ifdef _DEBUG
47 void CChildFrame::AssertValid() const
48 {
49     CMDIChildWndEx::AssertValid();
50 }
51
52 void CChildFrame::Dump(CDumpContext& dc) const
53 {
54     CMDIChildWndEx::Dump(dc);
55 }
56 #endif // _DEBUG
```

```
56
57 // CChildFrame message handlers
58
```

```
1 //{{NO_DEPENDENCIES}}
2 // Microsoft Visual C++ generated include file.
3 // Used by DesignArk.rc
4 //
5 #define IDP_OLE_INIT_FAILED 100
6 #define IDD_SMARCOLOUR_TYPES 107
7 #define IDD_SMARCOLOUR_OPS 108
8 #define IDD_SMARCOLOUR_BUILTIN 109
9 #define IDD_SMARCOLOUR_OTHER 110
10 #define IDR_POPUP_EDIT 119
11 #define ID_STATUSBAR_PANE1 120
12 #define ID_STATUSBAR_PANE2 121
13 #define IDS_STATUS_PANE1 122
14 #define IDS_STATUS_PANE2 123
15 #define IDS_TOOLBAR_STANDARD 124
16 #define IDS_TOOLBAR_CUSTOMIZE 125
17 #define ID_VIEW_CUSTOMIZE 126
18 #define IDR_MAINFRAME 128
19 #define IDR_MAINFRAME_256 129
20 #define IDR_DesignArkTYPE 130
21 #define ID_WINDOW_MANAGER 131
22 #define ID_VIEW_FILEVIEW 133
23 #define ID_VIEW_CLASSVIEW 134
24 #define ID_PROPERTIES 135
25 #define ID_OPEN 136
26 #define ID_OPEN_WITH 137
27 #define ID_DUMMY_COMPILE 138
28 #define ID_CLASS_ADD_MEMBER_FUNCTION 139
29 #define ID_CLASS_ADD_MEMBER_VARIABLE 140
30 #define ID_CLASS_DEFINITION 141
31 #define ID_CLASS_PROPERTIES 142
32 #define ID_NEW_FOLDER 143
33 #define ID_SORT_MENU 144
34 #define ID_SORTING_GROUPBYTYPE 145
35 #define ID_SORTING_SORTALPHABETIC 146
36 #define ID_SORTING_SORTBYTYPE 147
37 #define ID_SORTING_SORTBYACCESS 148
38 #define ID_VIEW_PROPERTIESWND 150
39 #define ID_SORTPROPERTIES 151
40 #define ID_PROPERTIES1 152
41 #define ID_PROPERTIES2 153
42 #define ID_EXPAND_ALL 154
43 #define IDS_FILE_VIEW 155
44 #define IDS_CLASS_VIEW 156
45 #define IDS_PROPERTIES_WND 158
46 #define IDI_FILE_VIEW 161
47 #define IDI_FILE_VIEW_HC 162
48 #define IDI_CLASS_VIEW 163
49 #define IDI_CLASS_VIEW_HC 164
50 #define IDI_PROPERTIES_WND 167
51 #define IDI_PROPERTIES_WND_HC 168
52 #define IDR_EXPLORER 169
53 #define IDB_EXPLORER_24 170
54 #define IDR_SORT 171
55 #define IDB_SORT_24 172
56 #define IDR_POPUP_SORT 173
```

```
57 #define IDR_POPUP_EXPLORER 174
58 #define IDB_FILE_VIEW 175
59 #define IDB_FILE_VIEW_24 176
60 #define IDB_CLASS_VIEW 177
61 #define IDB_CLASS_VIEW_24 178
62 #define IDR_MENU_IMAGES 179
63 #define IDB_MENU_IMAGES_24 180
64 #define ID_TOOLS_MACRO 181
65 #define IDR_PROPERTIES 183
66 #define IDB_PROPERTIES_HC 184
67 #define IDR_THEME_MENU 200
68 #define ID_SET_STYLE 201
69 #define ID_VIEW_APPLOOK_WIN_2000 205
70 #define ID_VIEW_APPLOOK_OFF_XP 206
71 #define ID_VIEW_APPLOOK_WIN_XP 207
72 #define ID_VIEW_APPLOOK_OFF_2003 208
73 #define ID_VIEW_APPLOOK_VS_2005 209
74 #define ID_VIEW_APPLOOK_VS_2008 210
75 #define ID_VIEW_APPLOOK_OFF_2007_BLUE 215
76 #define ID_VIEW_APPLOOK_OFF_2007_BLACK 216
77 #define ID_VIEW_APPLOOK_OFF_2007_SILVER 217
78 #define ID_VIEW_APPLOOK_OFF_2007_AQUA 218
79 #define ID_VIEW_APPLOOK_WINDOWS_7 219
80 #define IDS_EXPLORER 305
81 #define IDS_EDIT_MENU 306
82 #define IDB_CARET 310
83 #define IDB_PNG1 329
84 #define IDB_ARROW_RIGHT_HOVER 334
85 #define IDB_BITMAP1 336
86 #define IDB_ARROW_RIGHT_CLICK 336
87 #define IDD_APP_ABOUT 343
88 #define IDC_MFCVSLISTBOX_TYPES 1051
89 #define IDC_SPLIT1 1052
90 #define IDC_COMBO_COLOUR_TYPES 1053
91 #define IDC_MFCVSLISTBOX_OPS 1064
92 #define IDC_COMBO_COLOUR_OPS 1065
93 #define IDC_MFCVSLISTBOX_BUILTIN_CLASSES 1066
94 #define IDC_SMARCOLOUR_OTHERS_NUMBERS_COLOUR 1066
95 #define IDC_MFCVSLISTBOX_OPS2 1066
96 #define IDC_MFCVSLISTBOX_BUILTIN_FUNCTIONS 1067
97 #define IDC_COMBO_COLOUR_OPS2 1067
98 #define IDC_SMARCOLOUR_OTHERS_NUMBERS_COLOUR2 1067
99 #define IDC_COMBO_COLOUR_BUILTIN_FUNCTIONS 1068
100 #define IDC_COMBO_COLOUR_BUILTIN_CLASSES 1069
101 #define IDC_SMARCOLOUR_OTHERS_COMMENT_COMBO 1070
102 #define IDC_SMARCOLOUR_OTHERS_COMMENT_COMBO2 1071
103 #define IDC_SMARCOLOUR_OTHERS_FONT 1071
104 #define IDC_SMARCOLOUR_OTHER_COMMENTS_COLOUR 1074
105 #define IDC_SMARCOLOR_OTHER_INDENT 1083
106 #define ID_VIEW_SMARCOLOURWINDOW 32771
107 #define ID_VIEW_ 32772
108 #define ID_HELP_OPENWEBSITE 32773
109 #define ID_HELP_VIEWUSERDOCSONLINE 32776
110
111 // Next default values for new objects
112 //
```

```
113 #ifdef APSTUDIO_INVOKED
114 #ifndef APSTUDIO_READONLY_SYMBOLS
115 #define _APS_NEXT_RESOURCE_VALUE      345
116 #define _APS_NEXT_COMMAND_VALUE       32777
117 #define _APS_NEXT_CONTROL_VALUE        1084
118 #define _APS_NEXT_SYMED_VALUE         310
119 #endif
120#endif
121
```

```
C:\Users\james\source\repos\DesignArk\DesignArk\pch.h 1
1 // pch.h: This is a precompiled header file.
2 // Files listed below are compiled only once, improving build      ↵
3 // performance for future builds.                                ↵
4 // This also affects IntelliSense performance, including code      ↵
5 // completion and many code browsing features.                  ↵
6 // However, files listed here are ALL re-compiled if any one of them ↵
7 // is updated between builds.                                     ↵
8 // Do not add files here that you will be updating frequently as this ↵
9 // negates the performance advantage.
10 // add headers that you want to pre-compile here
11 #include "framework.h"
12
13 #include <iostream>
14 #include <fstream>
15 #include <sstream>
16 #include <map>
17 #include <string>
18 #include <vector>
19 #include <memory>
20 #include <math.h>
21 #include <shellapi.h>
22 #include <gdiplus.h>
23 #include <stdio.h>
24
25 #endif //PCH_H
26
27 using namespace std;
28
```

C:\Users\james\source/repos\DesignArk\DesignArk\pch.cpp 1

---

```
1 // pch.cpp: source file corresponding to the pre-compiled header
2
3 #include "pch.h"
4
5 // When you are using pre-compiled headers, this source file is      ↵
6     necessary for compilation to succeed.
```

```
1 #pragma once
2
3 #ifndef VC_EXTRALEAN
4 #define VC_EXTRALEAN           // Exclude rarely-used stuff from Windows headers
5 #endif
6
7 #include "targetver.h"
8
9 #define _ATL_CSTRING_EXPLICIT_CONSTRUCTORS     // some CString constructors will be explicit
10
11 // turns off MFC's hiding of some common and often safely ignored warning messages
12 #define _AFX_ALL_WARNINGS
13
14 #include <afxwin.h>             // MFC core and standard components
15 #include <afxext.h>             // MFC extensions
16
17
18 #include <afxdisp.h>            // MFC Automation classes
19
20
21
22 #ifndef _AFX_NO_OLE_SUPPORT
23 #include <afxdtctl.h>           // MFC support for Internet Explorer
24     4 Common Controls
25 #endif
26 #ifndef _AFX_NO_AFXCMN_SUPPORT
27 #include <afxcmn.h>             // MFC support for Windows Common Controls
28 #endif // _AFX_NO_AFXCMN_SUPPORT
29
30 #include <afxcontrolbars.h>      // MFC support for ribbons and control bars
31
32
33
34
35
36
37
38
39 #ifdef _UNICODE
40 #if defined _M_IX86
41 #pragma comment(linker,"/manifestdependency:\"type='win32' name='Microsoft.Windows.Common-Controls' version='6.0.0.0' processorArchitecture='x86' publicKeyToken='6595b64144ccf1df' language='*'\"")
42 #elif defined _M_X64
43 #pragma comment(linker,"/manifestdependency:\"type='win32' name='Microsoft.Windows.Common-Controls' version='6.0.0.0' processorArchitecture='amd64' publicKeyToken='6595b64144ccf1df' language='*'\"")
44 #else
```

```
...james\source\repos\DesignArk\DesignArk\framework.h 2
45 #pragma comment(linker,"/manifestdependency:type='win32'
    name='Microsoft.Windows.Common-Controls' version='6.0.0.0'
    processorArchitecture='*' publicKeyToken='6595b64144ccf1df'
    language='*'\"")
46 #endif
47 #endif
48
49
50
```

```
...james\source\repos\DesignArk\DesignArk\targetver.h 1
1 #pragma once
2
3 // Including SDKDDKVer.h defines the highest available Windows      ↵
4 // platform.
5 // If you wish to build your application for a previous Windows      ↵
6 // platform, include WinSDKVer.h and      ↵
7 // set the _WIN32_WINNT macro to the platform you wish to support      ↵
8 // before including SDKDDKVer.h.
9
8 #include <SDKDDKVer.h>
```

```
1
2 #pragma once
3 #include "CAppObject.h"
4 #include "CTextHandler.h"
5 #include "CTextEditorObject.h"
6
7 class CDocView
8 {
9 public:
10     // Public constructors
11     CDocView();
12     ~CDocView();
13
14 protected:
15
16     // Protected Commands
17     virtual void updateWindow(BOOL caret = TRUE, BOOL window = TRUE) ↵
18         = 0;
19
20     // Protected resources
21     CTextHandler textInput;
22     CFont fFont;
23
24     std::map<CString, CBitmap*> m_pBitmaps;
25 };
26
27
```

```
1
2 #include "pch.h"
3 #include "framework.h"
4
5 #ifndef SHARED_HANDLERS
6 #include "DesignArk.h"
7 #endif
8
9 #include "CDocView.h"
10
11
12 // Public constructors
13 CDocView::CDocView()
14 {
15     this->m_pBitmaps[L"caret"] = new CBitmap();
16     this->m_pBitmaps[L"caret"]->LoadBitmap(IDB_CARET);
17 }
18 CDocView::~CDocView()
19 {
20
21     for (auto& it : this->m_pBitmaps) {
22         delete it.second;
23     }
24 }
25
26 void CDocView::updateWindow(BOOL caret, BOOL window)
27 {
28 }
```

29

```
1
2 #pragma once
3 #include "CDocView.h"
4 #include "CTextDocument.h"
5
6 #define PRINT_SIZE 5
7
8 #define INITIALISE 1000 // USED FOR DRAWING ENTIRE VIEW
9 #define TEXT 1001 // USED FOR TEXT EDITS - NEW TEXT, ↵
    BACKSPACE ON TEXT
10 #define RETURN_BACK 1002 // USED FOR ADDING OR REMOVING ↵
    LINES WITH BACKSPACE OR RETURN
11 #define EDITOR_EDIT 1003 // USED FOR ADDING OR REMOVING ↵
    EDITORS
12 #define HIGHLIGHTING 1004 // USED FOR WHEN HIGHLIGHTING TEXT
13 #define SELECT_LINE 1007 // USED WHEN A DIFFERENT LINE IS ↵
    SELECTED
14 #define REMOVE_HIGHLIGHT 1008 // USED WHEN A HIGHLIGHT IS BEING ↵
    DELETED
15 #define CARET_MOVE 1009 // USED WHEN THE CARET IS MOVED ↵
    WITH THE ARROW KEYS
16 #define ADD_RGN_SIDEBAR_BTN 1010 // USED TO ADD THE AREA OF THE ↵
    SIDEBAR BUTTON TO THE REGION
17 #define ADD_RGN_ACTIVE_LINE 1011
18 #define H_SCROLL 1012
19 #define PASTE 1013
20
21
22 class CTextDocView :
23     public CScrollView, public CDocView
24 {
25 protected:
26     // Public constructors
27     CTextDocView() noexcept;
28     virtual ~CTextDocView();
29
30     DECLARE_DYNCREATE(CTextDocView)
31
32     // Public attributes
33     CTextDocument* GetDocument() const;
34     CSize GetDocSize();
35     int getActiveEditor();
36
37     // Public implementations
38 #ifdef _DEBUG
39     virtual void AssertValid() const;
40     virtual void Dump(CDumpContext& dc) const;
41 #endif
42     void refresh(BOOL caret = TRUE, BOOL window = TRUE);
43
44 protected:
45
46     // Protected implementations
47     virtual void updateWindow(BOOL caret = TRUE, BOOL window = ↵
        TRUE);
48     void ResyncScrollSizes(BOOL docSize = TRUE, BOOL reposition = ↵
```

```
48     FALSE);  
49  
50     // Protected overrides  
51     virtual void OnDraw(CDC* pDC);  
52     afx_msg BOOL OnEraseBkgnd(CDC* pDC);  
53  
54     virtual BOOL PreCreateWindow(CREATESTRUCT& cs);  
55     virtual BOOL OnSetCursor(CWnd* pWnd, UINT nHitTest, UINT message);  
56     virtual void OnActivateView(BOOL bActivate, CView* pActivateView, CView* pDeactiveView);  
57  
58     virtual BOOL OnPreparePrinting(CPrintInfo* pInfo);  
59     virtual void OnBeginPrinting(CDC* pDC, CPrintInfo* pInfo);  
60     virtual void OnEndPrinting(CDC* pDC, CPrintInfo* pInfo);  
61     virtual void OnPrepareDC(CDC* pDC, CPrintInfo* pInfo = (CPrintInfo*)0);  
62     virtual void OnPrint(CDC* pDC, CPrintInfo* pInfo);  
63  
64     // Protected message handlers  
65     afx_msg int OnCreate(LPCREATESTRUCT lpCS);  
66     virtual void OnInitialUpdate();  
67     afx_msg void OnSize(UINT nType, int cx, int cy);  
68  
69     afx_msg void OnFilePrintPreview();  
70  
71     afx_msg void OnHScroll(UINT nSBCode, UINT nPos, CScrollBar* pScrollBar);  
72     afx_msg void OnLButtonUp(UINT nFlags, CPoint point);  
73     afx_msg void OnLButtonDown(UINT nFlags, CPoint point);  
74     afx_msg void OnLButtonDblClk(UINT nFlags, CPoint point);  
75     afx_msg void OnMouseMove(UINT nFlags, CPoint point);  
76     afx_msg BOOL OnMouseWheel(UINT nFlags, short zDelta, CPoint pt);  
77     afx_msg void OnRButtonUp(UINT nFlags, CPoint point);  
78     afx_msg void OnRButtonDown(UINT nFlags, CPoint point);  
79     afx_msg void OnKeyDown(UINT nChar, UINT nRepCnt, UINT nFlags);  
80     afx_msg void OnKeyUp(UINT nChar, UINT nRepCnt, UINT nFlags);  
81  
82     virtual void OnContextMenu(CWnd* pWnd, CPoint point);  
83  
84     void OnCopy();  
85     void OnPaste();  
86     void OnCut();  
87  
88     DECLARE_MESSAGE_MAP()  
89  
90 private:  
91  
92     // Private Methods  
93     void expandLineBounds();  
94     void selectClippingRgn(int nAction, int type = -1);  
95  
96     void drawHeader(CDC* pDC);  
97  
98     // Private Resources  
99     int activeEditor;
```

```
100
101     int recentEditor; // Used to store the last editor that was used to change lines
102     int recentLine; // Used to store the last line that was used, to be used in combination with 'recentEditor'
103     int recentBlockLine;
104     BOOL recentHlght;
105     int recentPrintZoom;
106
107     CRgn* rgn;
108     CPrintInfo* pInfo;
109     BOOL printing;
110     int printIterator;
111     int returnNewLines;
112 };
113
114 #ifndef _DEBUG // debug version in DesignArkView.cpp
115 inline CTextDocument* CTextDocView::GetDocument() const
116 {
117     return reinterpret_cast<CTextDocument*>(m_pDocument);
118 }
119#endif
```

```
1 // CTextDocView.cpp : implementation of the CTextDocView class
2 //
3 //
4
5 #include "pch.h"
6 #include "framework.h"
7 // SHARED_HANDLERS can be defined in an ATL project implementing preview, thumbnail
8 // and search filter handlers and allows sharing of document code with that project.
9 #ifndef SHARED_HANDLERS
10 #include "DesignArk.h"
11 #endif
12
13 #include "CTextDocument.h"
14 #include "CTextDocView.h"
15
16 #include "CTextEditorObject.h"
17
18 #ifdef _DEBUG
19 #define new DEBUG_NEW
20 #endif
21
22
23 // CTextDocView
24 IMPLEMENT_DYNCREATE(CTextDocView, CScrollView)
25
26 BEGIN_MESSAGE_MAP(CTextDocView, CScrollView)
27
28     ON_WM_ERASEBKGND()
29
30     ON_WM_CREATE()
31     ON_WM_SETCURSOR()
32     ON_WM_SIZE()
33
34     // Standard printing commands
35     ON_COMMAND(ID_FILE_PRINT, &CScrollView::OnFilePrint)
36     ON_COMMAND(ID_FILE_PRINT_DIRECT, &CScrollView::OnFilePrint)
37     ON_COMMAND(ID_FILE_PRINT_PREVIEW,
38                 &CTextDocView::OnFilePrintPreview)
39
40     ON_WM_HSCROLL()
41
42     ON_WM_LBUTTONDOWN()
43     ON_WM_LBUTTONDBLCLK()
44     ON_WM_MOUSEMOVE()
45     ON_WM_MOUSEWHEEL()
46     ON_WM_RBUTTONDOWN()
47     ON_WM_RBUTTONDBLCLK()
48
49     ON_WM_KEYDOWN()
50     ON_WM_KEYUP()
51
52     ON_WM_CONTEXTMENU()
```

```
54     ON_COMMAND(ID_EDIT_COPY, &CTextDocView::OnCopy)
55     ON_COMMAND(ID_EDIT_PASTE, &CTextDocView::OnPaste)
56     ON_COMMAND(ID_EDIT_CUT, &CTextDocView::OnCut)
57
58 END_MESSAGE_MAP()
59
60 // CTextDocView construction/destruction
61
62 CTextDocView::CTextDocView() noexcept
63 {
64     SetScrollSizes(MM_TEXT, CSize(0, 0));
65     this->rgn = new CRgn();
66     this->recentBlockLine = 0;
67
68     this->printIterator = 0;
69     this->returnNewLines = 0;
70     this->recentPrintZoom = theApp.zoom;
71     this->recentHight = FALSE;
72     this->printing = FALSE;
73 }
74 CTextDocView::~CTextDocView()
75 {
76     delete this->rgn;
77 }
78
79 // Public attributes
80 CTextDocument* CTextDocView::GetDocument() const // non-debug ↵
version is inline
81 {
82     ASSERT(m_pDocument->IsKindOf(RUNTIME_CLASS(CDocument)));
83     return (CTextDocument*)m_pDocument;
84 }
85 CSize CTextDocView::GetDocSize()
86 {
87     int sizeX = 0, sizeY = 0;
88
89     for (int i = 0; i < this->GetDocument()->objects.GetSize(); i + ↵
+) {
90
91         if (sizeX < this->GetDocument()->objects[i]- ↵
>getLineTextWidth()) {
92             sizeX = this->GetDocument()->objects[i]- ↵
>getLineTextWidth();
93         }
94
95         sizeY += this->GetDocument()->objects[i]->getBounds ↵
().Height() + (this->GetDocument()->objects[i]- ↵
>getBoxHeight());
96     }
97
98     CRect client;
99     GetClientRect(&client);
100
101    sizeX += (client.Width() * 0.1) + static_cast<int>(this- ↵
>GetDocument()->lineOffset * theApp.zoom); ↵
102 }
```

```
103     sizeY += client.Height() -  
104         2*(this->GetDocument()->objects[this->GetDocument()->  
105            >objects.GetUpperBound()]->getBoxHeight());  
106     return CSize(sizeX, sizeY);  
107 }  
108  
109 int CTextDocView::getActiveEditor()  
110 {  
111     return this->activeEditor;  
112 }  
113  
114 // Public implementations  
115 #ifdef _DEBUG  
116 void CTextDocView::AssertValid() const  
117 {  
118     CView::AssertValid();  
119 }  
120 void CTextDocView::Dump(CDumpContext& dc) const  
121 {  
122     CView::Dump(dc);  
123 }  
124 #endif //_DEBUG  
125 void CTextDocView::refresh(BOOL caret, BOOL window)  
126 {  
127     if (caret) {  
128         this->CreateSolidCaret(this->GetDocument()->objects[this->  
129             >activeEditor]->getBoxHeight() * 0.05625, this->  
130             >GetDocument()->objects[this->activeEditor]->getBoxHeight()  
131             * 0.8);  
132     }  
133     if (window) {  
134         this->updateWindow(FALSE, TRUE);  
135     }  
136 }  
137  
138 // Protected implementations  
139 void CTextDocView::updateWindow(BOOL caret, BOOL window)  
140 {  
141     if (window) {  
142         CRect client;  
143         this->GetClientRect(client);  
144         this->RedrawWindow(client);  
145     }  
146     else if (caret) {  
147         this->SetCaretPos(this->GetDocument()->objects[this->  
148             >activeEditor]->getCaretPoint(CSize(this->GetDocument()->  
149             >objects[this->activeEditor]->getBoxHeight() * 0.05,  
150             this->GetDocument()->objects[this->activeEditor]->  
151             getBoxHeight() * 0.8)) - CSize(this->GetScrollPosition  
152             ()));  
153         this->>ShowCaret();  
154     }  
155 }  
156  
157 void CTextDocView::ResyncScrollSizes(BOOL docSize, BOOL reposition)
```

```
150 {
151     CSize sizeDoc;
152     if (docSize) {
153         // Primary stuff with scroll bars
154         CCClientDC dc(NULL);
155         OnPrepareDC(&dc);
156         sizeDoc = this->GetDocSize();
157         dc.LPtodP(&sizeDoc);
158         SetScrollSizes(MM_TEXT, sizeDoc);
159     }
160
161     // Secondary stuff with objects and realignment
162     CRect client;
163     GetClientRect(&client); // Find current window size
164
165     BOOL redraw = FALSE;
166
167     int tempX = this->GetScrollPosition().x;
168
169     // Is the position of the cursor on the screen
170     int activeRight = this->GetDocument()->objects[this-
171         >activeEditor]->getBounds().left + static_cast<int>(this-
172         >GetDocument()->lineOffset * theApp.zoom) + this->GetDocument(
173             )->objects[this->activeEditor]->getCaretPos() * this-
174             >GetDocument()->objects[this->activeEditor]->getTextExtent
175             ().cx + 7;
176
177     // If the position of the cursor is beyond the edge of the
178     // window...
179     if (tempX + client.Width() < activeRight && client.Width() > 0)
180     {
181
182         POINT scrollpt;
183         scrollpt.x = activeRight - client.Width() + 5; // Move the
184             X to where the scroll is
185         scrollpt.y = this->GetScrollPosition().y; // Keep Y the
186             same
187
188         this->ScrollToPosition(scrollpt);
189         redraw = TRUE; // We need to redraw the window now
190
191         for (int i = 0; i < this->GetDocument()->objects.GetSize(); i++)
192             { // Resize all the objects so that the text fits
193                 inside
194
195                     this->GetDocument()->objects[i]->OnSize(0, this-
196                         >GetScrollPosition().x + client.Width(), this-
197                         >GetScrollPosition().y + client.Height());
198
199                 }
200             }
201
202             // Is the upper bounds of the active line
203             int activeBottom = this->GetDocument()->objects[this-
204                 >activeEditor]->getLineBounds(this->GetDocument()->objects
205                     [this->activeEditor]->getActiveLine()).top;
```

```
191
192     if (this->GetScrollPosition().y + client.Height() <
193         activeBottom && client.Height() > 0) {
194
195         POINT pt;
196         pt.x = this->GetScrollPosition().x;
197         pt.y = activeBottom - client.Height();
198
199         this->ScrollToPosition(pt);
200         redraw = TRUE;
201     } // If the cursor is under the screen, scroll down
202
203     if (!docSize) {
204         if (this->GetScrollPosition().x > activeRight &&
205             client.Width() > 0) {
206
207             POINT pt;
208             pt.x = activeRight - this->GetDocument()->lineOffset - 7;
209             pt.y = this->GetScrollPosition().y;
210
211             this->ScrollToPosition(pt);
212             redraw = TRUE;
213         }
214         if (this->GetScrollPosition().y > activeBottom - this-
215             >GetDocument()->defBoxHeight && client.Height() > 0) {
216
217             POINT pt;
218             pt.x = this->GetScrollPosition().x;
219             pt.y = activeBottom - this->GetDocument()->
220                 defBoxHeight;
221
222             this->ScrollToPosition(pt);
223             redraw = TRUE;
224         }
225     }
226
227     if (reposition) {
228
229         this->OnSize(0, client.Width(), client.Height());
230
231         CPoint point;
232         point.x = 0;
233         point.y = 0;
234
235         for (int i = 0; i < this->GetDocument()->objects.GetSize(); i++)
236             {
237
238                 this->GetDocument()->objects[i]->setPosition(point);
239                 point.y = this->GetDocument()->objects[i]->getBounds
240                     ().bottom + this->GetDocument()->objects[i]-
241                         >getBoxHeight();
242
243             }
244     }
245
246     if (tempX != this->GetScrollPosition().x) {
```

```
239         this->expandLineBounds();
240     }
241
242     this->updateWindow(FALSE, redraw);
243 }
244
245 // Protected overrides
246 void CTextDocView::OnDraw(CDC* pDC)
247 {
248     CTextDocument* pDoc = this->GetDocument();
249     ASSERT_VALID(pDoc);
250     if (!pDoc)
251         return;
252
253     if (!this->printing) {
254         this->fFont.CreateFont(static_cast<int>(pDoc->defBoxHeight * theApp.zoom), 0, 0, 0, FW_NORMAL, FALSE, FALSE, 0, ANSI_CHARSET, OUT_DEFAULT_PRECIS, CLIP_DEFAULT_PRECIS, DEFAULT_QUALITY, 0, theApp.sFont);
255         pDC->SelectObject(&this->fFont);
256
257     } // If not printing, create a font here
258
259     BOOL finished = FALSE;
260     int i = 0; // Iterator
261     if (this->printing) { // If we are printing
262         i = this->printIterator; // Set the printIterator to where we left off
263     }
264
265
266     while (!finished && i < pDoc->objects.GetSize()) {
267
268         if (this->printing) { // If we are printing, do this
269
270             int bottom = std::get<0>(pDoc->objects[i]->getPrintBounds(returnNewLines, this->pInfo->m_rectDraw.Width())).bottom; // TODO : Remove this bit
271
272             if (bottom < this->pInfo->m_rectDraw.bottom) { // If the object will completley fit on the print page
273
274                 this->printIterator++;
275                 returnNewLines = pDoc->objects[i]->draw(pDC, CSize (), this->GetScrollPosition().x, returnNewLines, this->printing, this->pInfo->m_rectDraw); // Draw object on page as normal
276             }
277             else { // If not, ie is on the next page in any way
278
279                 // If the object should be printed on the next page completley
280                 if (std::get<0>(pDoc->objects[i]->getPrintBounds (returnNewLines, this->pInfo->m_rectDraw.Width ())) .top + pDoc->objects[i]->getBoxHeight() >=
```

```
281                     this->pInfo->m_rectDraw.bottom) {  
282                         pDoc->objects[i]->setPosition(CPoint(this-  
283 >GetDocument()->objects[i]->getBounds().left,  
284 this->pInfo->m_rectDraw.top - (pDoc->objects[i]-  
285 >getPrintLine() * pDoc->objects[i]->getBoxHeight  
286 (())));  
287                 }  
288             else { // If part of the object should be printed  
289             on the current page, and the rest on the next  
290                     returnNewLines = pDoc->objects[i]->draw(pDC,  
291 CSIZE(), this->GetScrollPosition().x,  
292 returnNewLines, this->printing, this->pInfo-  
293 >m_rectDraw);  
294                     pDoc->objects[i]->setPosition(CPoint(pDoc-  
295 >objects[i]->getBounds().left, this->pInfo-  
296 >m_rectDraw.top - ((pDoc->objects[i]->getPrintLine()  
297 () + returnNewLines) * pDoc->objects[i]-  
298 >getBoxHeight()));  
299                 }  
300             }  
301         for (int j = i + 1; j < pDoc->objects.GetSize(); j+  
302 +) {  
303             pDoc->objects[j]->move(0, pDoc->objects[j - 1]-  
304 >getBounds().bottom + pDoc->objects[j - 1]-  
305 >getBoxHeight() - pDoc->objects[j]->getBounds  
306 ().top);  
307         } // Move the objects relative to the previous  
308         object  
309     this->drawHeader(pDC); // Draw the header for the  
310         next page  
311     finished = TRUE;  
312     }  
313     else {  
314         returnNewLines = pDoc->objects[i]->draw(pDC, CSIZE(),  
315 this->GetScrollPosition().x, this->printing,  
316 returnNewLines);  
317     }  
318     i++;  
319 }  
320 if (this->printing) {  
321     if (i == pDoc->objects.GetSize()) {  
322         this->drawHeader(pDC);  
323     }  
324 }
```

```
316
317     if (!this->printing) {
318         this->updateWindow(TRUE, FALSE); // DO NOT CHANGE SECOND ↵
319         this->fFont.DeleteObject();
320     }
321
322     //pDc->FrameRgn(this->rgn, new CBrush(RGB(255, 0, 0)), -1, -1);
323     this->rgn = new CRgn();
324 }
325 BOOL CTextDocView::OnEraseBkgnd(CDC* pDC)
326 {
327     pDC->SelectClipRgn(this->rgn);
328     CView::OnEraseBkgnd(pDC);
329     //this->rgn = new CRgn();
330
331     return 0;
332 }
333
334 BOOL CTextDocView::PreCreateWindow(CREATESTRUCT& cs)
335 {
336     return CView::PreCreateWindow(cs);
337 }
338 BOOL CTextDocView::OnSetCursor(CWnd* pWnd, UINT nHitTest, UINT ↵
339     message)
340 {
341     return TRUE;
342 }
343 void CTextDocView::OnActivateView(BOOL bActivate, CView* ↵
344     pActivateView, CView* pDeactiveView)
345 {
346     CView::OnActivateView(bActivate, pActivateView, pDeactiveView);
347     theApp.m_ActiveView = this;
348     this->refresh();
349 }
350 BOOL CTextDocView::OnPreparePrinting(CPrintInfo* pInfo)
351 {
352     return this->DoPreparePrinting(pInfo);
353 }
354 void CTextDocView::OnBeginPrinting(CDC* pDc, CPrintInfo* pInfo)
355 {
356     this->printing = TRUE;
357
358     this->recentPrintZoom = theApp.zoom; // Save the current zoom ↵
359     so we can resize after the printing
360     theApp.zoom = 5; // TODO : Fix this so that it isn't constant
361
362     this->fFont.CreateFont(static_cast<int>(this->GetDocument()->defBoxHeight * theApp.zoom), 0, 0, 0, FW_NORMAL, FALSE, ↵
363     FALSE, 0, ANSI_CHARSET, OUT_DEFAULT_PRECIS, ↵
364     CLIP_DEFAULT_PRECIS, DEFAULT_QUALITY, 0, theApp.sFont);
365     pDc->SelectObject(&this->fFont);
366 }
367 }
```

```
365 void CTextDocView::OnEndPrinting(CDC* pDC, CPrintInfo* pInfo)
366 {
367     theApp.zoom = this->recentPrintZoom;
368
369     CRect client;
370     this->GetClientRect(&client);
371     this->OnSize(PRINT_SIZE, client.Width(), client.Height());
372
373     this->GetDocument()->objects[0]->setPosition(CPoint(0, 0));
374
375     for (int i = 1; i < this->GetDocument()->objects.GetSize(); i++)
376     {
377         this->GetDocument()->objects[i]->move(0, this->GetDocument()-
378             ()->objects[i - 1]->getBounds().bottom + this-
379             >GetDocument()->objects[i - 1]->getBoxHeight() - this-
380             >GetDocument()->objects[i]->getBounds().top);
381     } // Move the resized objects
382
383     this->printIterator = 0;
384     this->returnNewLines = 0;
385     pInfo->m_bContinuePrinting = FALSE;
386
387     this->printing = FALSE;
388
389     this->fFont.DeleteObject();
390     CScrollView::OnEndPrinting(pDC, pInfo);
391 }
392
393 void CTextDocView::OnPrepareDC(CDC* pDC, CPrintInfo* pInfo)
394 {
395     UNREFERENCED_PARAMETER(pInfo);
396
397     if (pInfo->m_nCurPage == 1) { // If we have just started
398         // printing, start the process of finding the length of the
399         // document
400
401         CRect client;
402         this->GetClientRect(&client);
403         this->OnSize(PRINT_SIZE, client.Width(), client.Height());
404         // Resize the document and stuff
405
406         pInfo->m_rectDraw.top += 600; // Create a space for the
407         // headers
408         pInfo->m_rectDraw.bottom -= 300; // Create a space for the
409         // margin
410
411         this->GetDocument()->objects[0]->move(0, pInfo-
412             >m_rectDraw.top);
413
414         for (int i = 1; i < this->GetDocument()->objects.GetSize(); i++)
415         {
416             this->GetDocument()->objects[i]->move(0, this-
```

```

...\\source\\repos\\DesignArk\\DesignArk\\CTextDocView.cpp 10
        >GetDocument() ->objects[i-1] ->getBounds().bottom + ↵
        this->GetDocument() ->objects[i-1] ->getBoxHeight() - ↵
        this->GetDocument() ->objects[i] ->getBounds().top);
410    } // Move the resized objects
411
412    std::tuple<CRect, int> printData;
413    std::get<0>(printData) = CRect();
414    std::get<1>(printData) = 0;
415
416    CSize textExtent = pDC->GetTextExtent(L"A");
417
418    for (int i = 0; i < this->GetDocument() ->objects.GetSize(); ↵
419        i++) {
420        this->GetDocument() ->objects[i] ->setTextExtent ↵
        (textExtent);
421        printData = this->GetDocument() ->objects[i] -> ↵
        >getPrintBounds(std::get<1>(printData), pInfo- ↵
        >m_rectDraw.Width());
422    } // Find the size of the document
423
424    pInfo->SetMaxPage(std::ceil(float(std::get<0>
425        (printData).bottom) / float(pInfo- ↵
        >m_rectDraw.bottom))); // Finally, set the number of ↵
        pages in the document
426
427    this->pInfo = pInfo; // Save the print info
428    this->OnDraw(pDC);
429
430    if (pInfo->GetMaxPage() == pInfo->m_nCurPage) { // Once we have ↵
        finished printing, resize and reposition all the editors
431    theApp.zoom = this->recentPrintZoom;
432
433    CRect client;
434    this->GetClientRect(&client);
435    this->OnSize(PRINT_SIZE, client.Width(), client.Height());
436
437    this->GetDocument() ->objects[0] ->setPosition(CPoint(0, 0));
438
439    for (int i = 1; i < this->GetDocument() ->objects.GetSize(); ↵
        i++) {
440        this->GetDocument() ->objects[i] ->move(0, this- ↵
        >GetDocument() ->objects[i-1] ->getBounds().bottom + ↵
        this->GetDocument() ->objects[i-1] ->getBoxHeight() - ↵
        this->GetDocument() ->objects[i] ->getBounds().top);
441    } // Move the resized objects
442
443    this->printIterator = 0;
444    this->returnNewLines = 0;
445    pInfo->m_bContinuePrinting = FALSE;
446    }
447 }
448
449 // Protected message handlers
450 int CTextDocView::OnCreate(LPCREATESTRUCT lpcs)

```

```
451 {
452     CView::OnCreate(lpcs);
453
454     ::SetCursor(AfxGetApp() ->LoadStandardCursor(IDC_IBEAM));
455
456     this->activeEditor = 0;
457
458     this->recentEditor = this->activeEditor;
459     this->recentLine = this->GetDocument()->objects[this->activeEditor]->getActiveLine(); ↵
460
461     return 0;
462 }
463 void CTextDocView::OnInitialUpdate()
464 {
465     ResyncScrollSizes();
466     CScrollView::OnInitialUpdate();
467 }
468 void CTextDocView::OnSize(UINT nType, int cx, int cy)
469 {
470     CScrollView::OnSize(nType, cx, cy);
471
472     if (nType != PRINT_SIZE) {
473         ResyncScrollSizes();           // ensure that scroll info is ↵
474                                     up-to-date
475     }
476
477     cx += this->GetScrollPosition().x;
478     cy += this->GetScrollPosition().y;
479
480     CRect newBounds;
481
482     for (int i = 0; i < this->GetDocument()->objects.GetSize(); i + ↵
483     +) {
484
485         this->GetDocument()->objects[i]->OnSize(nType, max(cx, ↵
486             this->GetDocument()->objects[i]->getLineTextWidth() + ↵
487             static_cast<int>(this->GetDocument()->lineOffset * ↵
488             theApp.zoom)), cy);
489
490         newBounds = this->GetDocument()->objects[i]->getBounds();
491         newBounds.right = max(cx, this->GetDocument()->objects[i]->getLineTextWidth() + static_cast<int>(this->GetDocument() ->lineOffset * theApp.zoom));
492
493         this->GetDocument()->objects[i]->setBounds(newBounds);
494     }
495 }
496
497 void CTextDocView::OnFilePrintPreview()
498 {
```

```
499 #ifndef SHARED_HANDLERS
500     AFXPrintPreview(this);
501 #endif
502 }
503
504 void CTextDocView::OnHScroll(UINT nSBCode, UINT nPos, CScrollBar* pScrollBar)
505 {
506     CScrollView::OnHScroll(nSBCode, nPos, pScrollBar);
507
508     this->expandLineBounds();
509
510     //this->selectClippingRgn(H_SCROLL);
511     this->updateWindow(FALSE, TRUE);
512 }
513 void CTextDocView::OnLButtonUp(UINT nFlags, CPoint point)
514 {
515     point += this->GetScrollPosition();
516
517     BOOL redraw = this->GetDocument()->objects[this->activeEditor]->
518         OnLButtonUp(nFlags, point);
519
520     std::vector<int> lineNums = this->GetDocument()->objects[this->
521         activeEditor]->iGetLineNum();
522
523     if (lineNums[0] != 0) {
524
525         lineNums.push_back(1);
526
527         BOOL iUpdate = FALSE;
528         int i = this->activeEditor+1;
529
530         BOOL exists = FALSE;
531         int pos = 1;
532
533         int lineNumSize = lineNums.size();
534
535         // TODO : Possibly change to a binary search to make
536         // searching more efficient
537         while (!iUpdate && i < this->GetDocument()->objects.GetSize())
538             { // Iterate through all editors under one that was
539             // clicked
540
541             if (lineNums == this->GetDocument()->objects[i]->
542                 iGetLineNum(1)) { // If the editor requested to be
543                 made has already been made, go to that editor and
544                 don't create a new one
545
546                 this->GetDocument()->objects[this->activeEditor]->
547                     setActive(FALSE);
548                 this->activeEditor = i;
549                 this->GetDocument()->objects[this->activeEditor]->
550                     setActive(TRUE);
551
552                 iUpdate = TRUE;
553                 exists = TRUE;
554             }
555         }
556     }
557 }
```

```
544         }
545
546         else { // If this editor does not match, we must check ↵
547             // to see if the requested editor should go before this ↵
548             // one, or if we should keep searching
549
550             BOOL jUpdate = FALSE;
551             int j = 0;
552
553             while (!jUpdate && j < min(lineNumSize, this-      ↵
554             >GetDocument()->objects[i]->iGetLineNum(1).size    ↵
555             ()) { // Iterate through the line numbers'        ↵
556                 sublines
557
558                 if (lineNums[j] < this->GetDocument()->objects    ↵
559                 [i]->iGetLineNum(1)[j]) { // If this passes, the    ↵
560                     editor should be above it
561
562                     jUpdate = TRUE;
563                     iUpdate = TRUE;
564
565                     pos = i;
566
567                 }
568
569                 i++;
570             }
571
572             if (!iUpdate) {
573                 pos = this->GetDocument()->objects.GetSize();
574             }
575
576             if (!exists) {
577
578                 this->GetDocument()->objects[this->activeEditor]-    ↵
579                 >setActive(FALSE); // Set old editor not active
580
581                 int oldEditor = this->activeEditor;
582                 this->activeEditor = pos; // Change position for    ↵
583                 editing
584
585                 CRect aboveRect = this->GetDocument()->objects[this-    ↵
586                 >activeEditor - 1]->getBounds();
587
588                 CRect editorBounds;
589                 editorBounds.top = aboveRect.bottom + this->GetDocument()    ↵
590                 ()->objects[this->activeEditor - 1]->getBoxHeight();
591                 editorBounds.bottom = editorBounds.top + this-
```

```

...\\source\\repos\\DesignArk\\DesignArk\\CTextDocView.cpp 14
    >GetDocument()->objects[this->activeEditor - 1]->
    >getBoxHeight();
588     editorBounds.left = aboveRect.left;
589     editorBounds.right = aboveRect.right;
590
591     std::vector<CString> text = { L"" };
592
593     text[0] = L"// TODO : Add a subline";
594
595     this->GetDocument()->objects.InsertAt(
596         this->activeEditor,
597         new CTextEditorObject(editorBounds, L"0", TRUE, 0, this-
598             >GetDocument()->objects[this->activeEditor - 1]->
599             getBoxHeight(TRUE), TRUE, text, lineNums,
600             this->GetDocument()->lineOffset)); // Add a new line
601
602     CString id;
603
604     for (int i = this->activeEditor + 1; i < this->GetDocument()->objects.GetSize(); i++) {
605
606         id.Format(L"%d", i + 1);
607
608         this->GetDocument()->objects[i]->move(0, this->GetDocument()->objects[this->activeEditor]->getBounds().Height() + this->GetDocument()->objects[this->activeEditor - 1]->getBoxHeight());
609         this->GetDocument()->objects[i]->setID(id);
610     } // Move all lines and reset their IDs
611
612     this->selectClippingRgn(EDITOR_EDIT); // Will draw everything under and including the added editor
613
614     // We should have something like this, but it works without it for some reason so it's getting left out for now
615     /*CRgn* tempRgn = this->rgn;
616     this->rgn = new CRgn();
617
618     this->setClippingRgn(SELECT_LINE);
619     this->rgn->CombineRgn(this->rgn, tempRgn, RGN_OR);*/
620
621     this->ResyncScrollSizes();
622 }
623     redraw = TRUE;
624 }
625     this->updateWindow(FALSE, redraw);
626 }
627 void CTextDocView::OnLButtonDown(UINT nFlags, CPoint point)
628 {
629     point += this->GetScrollPosition();

```

```
630
631     if (this->recentEditor != this->activeEditor) {
632         this->recentEditor = this->activeEditor;
633     }
634     if (this->recentLine != this->GetDocument()->objects[this-
635 >recentEditor]->getActiveLine()) {
636         this->recentLine = this->GetDocument()->objects[this-
637 >recentEditor]->getActiveLine();
638     }
639     BOOL update = FALSE;
640     int i = 0;
641     while (!update && i < this->GetDocument()->objects.GetSize()) {
642         if (this->GetDocument()->objects[i]->checkPointInBounds
643             (point) || point.y < this->GetDocument()->objects[i]-
644 >getBounds().bottom + this->GetDocument()->objects[i]-
645 >getBoxHeight()) {
646             if (i != this->activeEditor) {
647                 this->GetDocument()->objects[this->activeEditor]->
648 setActive(FALSE);
649                 this->activeEditor = i;
650                 this->GetDocument()->objects[this->activeEditor]->
651 setActive(TRUE);
652             }
653             i++;
654         }
655         if (!update) {
656             this->GetDocument()->objects[this->activeEditor]->setActive
657             (FALSE);
658             this->activeEditor = this->GetDocument()->objects.GetSize()-
659             1;
660             this->GetDocument()->objects[this->activeEditor]->setActive
661             (TRUE);
662         }
663         BOOL oldHlght = this->recentHlght;
664         if (this->recentHlght != this->GetDocument()->objects[this-
665 >recentEditor]->hasHighlight()) {
666             this->recentHlght = this->GetDocument()->objects[this-
667 >recentEditor]->hasHighlight();
668         }
669         BOOL sidebarbtn = this->GetDocument()->objects[this-
670 >activeEditor]->OnLButtonDown(nFlags, point);
671         this->selectClippingRgn(SELECT_LINE); // THIS NEEDS TO GO
BEFORE THE ADD_RGN_SIDEBAR_BTN
672         this->recentBlockLine = this->GetDocument()->objects[this-
```

```
    >recentEditor]->getBlockLine();
672     int oldEditor = this->activeEditor;
673     this->recentEditor = this->activeEditor;
674
675     if (sidebarbtn) {
676
677         if (this->recentLine == this->recentBlockLine && !oldHlght ↵
678             && !this->GetDocument()->objects[oldEditor]->hasHighlight ↵
679             ()) {
680             this->selectClippingRgn(ADD_RGN_SIDEBAR_BTN, 0);
681         }
682         else {
683             this->selectClippingRgn(ADD_RGN_SIDEBAR_BTN, 1);
684         }
685     }
686     this->updateWindow(FALSE, TRUE);
687 }
688 void CTextDocView::OnLButtonDblClk(UINT nFlags, CPoint point)
689 {
690     point += this->GetScrollPosition();
691
692     this->GetDocument()->objects[this->activeEditor]->OnLButtonDblClk(nFlags, point); ↵
693
694     for (int i = 0; i < this->GetDocument()->objects.GetSize(); i+ ↵
695         +) {
696         if (i != this->activeEditor) {
697
698             this->GetDocument()->objects[i]->hlghtingOff();
699         }
700     }
701     this->updateWindow(FALSE, TRUE);
702 }
703 void CTextDocView::OnMouseMove(UINT nFlags, CPoint point)
704 {
705     point += this->GetScrollPosition();
706
707     int old_cursor = this->GetDocument()->objects[this->activeEditor]->getCursorArrow(); ↵
708
709     BOOL objectRedraw = this->GetDocument()->objects[this->activeEditor]->OnMouseMove(nFlags, point); ↵
710
711     if (nFlags == VK_LEFT || objectRedraw) {
712
713         for (int i = 0; i < this->GetDocument()->objects.GetSize(); ↵
714             i++) {
715             if (i != this->activeEditor) {
716
717                 this->GetDocument()->objects[i]->hlghtingOff();
718             }
719         }
720
721         this->selectClippingRgn(HIGHLIGHTING);
```

```
720
721     if (this->recentBlockLine != this->GetDocument()->objects [this->recentEditor]->getBlockLine() && this-
722         >recentBlockLine != -1) {
723         this->selectClippingRgn(ADD_RGN_SIDEBAR_BTN, 1);
724         this->recentEditor = this->activeEditor;
725         this->recentBlockLine = this->GetDocument()->objects [this->recentEditor]->getBlockLine();
726     }
727     if (objectRedraw && nFlags != VK_LEFT && this->GetDocument ()->objects[this->activeEditor]->getBlockLine() != -1) {
728         this->recentBlockLine = this->GetDocument()->objects [this->activeEditor]->getBlockLine();
729         this->selectClippingRgn(ADD_RGN_SIDEBAR_BTN, 1);
730     }
731     this->updateWindow(FALSE, TRUE);
732
733     int new_cursor = this->GetDocument()->objects[this->activeEditor]->getCursorArrow();
734
735     if (new_cursor != old_cursor) {
736
737         if (new_cursor == 0) {
738
739             ::SetCursor(AfxGetApp()->LoadStandardCursor (IDC_IBEAM));
740         }
741         else if (new_cursor == 1) {
742
743             ::SetCursor(AfxGetApp()->LoadStandardCursor (IDC_ARROW));
744         }
745         else {
746
747             ::SetCursor(AfxGetApp ()->LoadStandardCursor (IDC_HAND));
748         }
749     }
750 }
751
752 BOOL CTextDocView::OnMouseWheel(UINT nFlags, short zDelta, CPoint pt)
753 {
754     if (nFlags == MK_CONTROL) {
755
756         BOOL adjust = TRUE;
757
758         if ((theApp.zoom <= 0.75 && zDelta < 0) || (theApp.zoom >=
759             2.8 && zDelta > 0)) {
760             adjust = FALSE;
761         }
762
763         if (adjust) {
764
765             theApp.zoom += zDelta / 1000.f;
766             this->ResyncScrollSizes(TRUE, TRUE);
767         }
768     }
769 }
```

```
766         }
767
768         this->refresh(TRUE, FALSE);
769
770         return TRUE;
771     }
772     else {
773         return CScrollView::OnMouseWheel(nFlags, zDelta, pt);
774     }
775 }
776
777 void CTextDocView::OnRButtonUp(UINT nFlags, CPoint point)
778 {
779     point += this->GetScrollPosition();
780
781     this->GetDocument()->objects[this->activeEditor]->OnRButtonUp(nFlags, point);
782
783
784     if (point.x - this->GetScrollPosition().x > this->GetDocument()-
785         ()->lineOffset) {
786         this->ClientToScreen(&point);
787         OnContextMenu(this, point);
788     }
789
790     void CTextDocView::OnRButtonDown(UINT nFlags, CPoint point)
791 {
792     point += this->GetScrollPosition();
793
794     if (this->recentEditor != this->activeEditor) {
795         this->recentEditor = this->activeEditor;
796     }
797     if (this->recentLine != this->GetDocument()->objects[this-
798         >recentEditor]->getActiveLine()) {
799         this->recentLine = this->GetDocument()->objects[this-
800             >recentEditor]->getActiveLine();
801     }
802
803     BOOL update = FALSE;
804     int i = 0;
805     while (!update && i < this->GetDocument()->objects.GetSize()) {
806
807         if (this->GetDocument()->objects[i]->checkPointInBounds
808             (point) || point.y < this->GetDocument()->objects[i]-
809             >getBounds().bottom + this->GetDocument()->objects[i]-
810             >getBoxHeight()) {
811             if (i != this->activeEditor) {
812
813                 this->GetDocument()->objects[this->activeEditor]->
814                     setActive(FALSE);
815                 this->activeEditor = i;
816                 this->GetDocument()->objects[this->activeEditor]->
817                     setActive(TRUE);
818             }
819             update = TRUE;
820         }
821     }
822 }
```

---

...\\source\\repos\\DesignArk\\DesignArk\\CTextDocView.cpp

```

813         i++;
814     }
815
816     if (!update) {
817
818         this->GetDocument()->objects[this->activeEditor]->setActive FALSE;
819         this->activeEditor = this->GetDocument()->objects.GetSize() - 1;
820         this->GetDocument()->objects[this->activeEditor]->setActive TRUE;
821     }
822
823     if (this->recentHlght != this->GetDocument()->objects[this->recentEditor]->hasHighlight()) {
824         this->recentHlght = this->GetDocument()->objects[this->recentEditor]->hasHighlight();
825     }
826
827     this->GetDocument()->objects[this->activeEditor]->OnRButtonDown (nFlags, point);
828
829     this->selectClippingRgn(SELECT_LINE); // THIS NEEDS TO GO BEFORE THE ADD_RGN_SIDEBAR_BTN
830
831     this->recentEditor = this->activeEditor;
832
833     this->updateWindow(FALSE, TRUE);
834 }
835 void CTextDocView::OnKeyDown(UINT nChar, UINT nRepCnt, UINT nFlags)
836 {
837     BOOL redraw = TRUE;
838
839     switch (nChar) {
840
841     case VK_RETURN: {
842
843         // TODO : Make this more efficient (but make it work first)
844
845         BOOL move = TRUE, increment = TRUE;
846
847         std::vector<int> line = this->GetDocument()->objects[this->activeEditor]->iGetLineNum(this->GetDocument()->objects[this->activeEditor]->getActiveLine() + 1);
848         line.push_back(1);
849
850         for (int i = this->activeEditor + 1; i < this->GetDocument()->objects.GetSize(); i++) {
851
852             if (this->GetDocument()->objects[i]->iGetLineNum(1) == line) {
853
854                 if (this->GetDocument()->objects[this->activeEditor]->getCaretPos() == this->GetDocument()->objects[this->activeEditor]->getLineText().GetLength()) {

```

```
855                     increment = FALSE;
856
857                 }
858             else if (this->GetDocument()->objects[this-
859 >activeEditor]->getCaretPos() != 0) {
860
861                     move = FALSE;
862
863                     // SEND A WARNING MESSAGE
864                     cout << "UNABLE TO PROCESS COMMAND" << endl;
865                 }
866             }
867         }
868     if (move) {
869
870         if (this->GetDocument()->objects[this->activeEditor]->
871 >hasHighlight()) {
872
873             this->OnKeyDown(VK_BACK, 1, 0);
874         }
875
876         this->GetDocument()->objects[this->activeEditor]->
877 >OnRecieveReturn(); // Send the return message to the
878 active editor so it can handle itself
879
880         // Move editors and increment sublines
881         for (int i = this->activeEditor + 1; i < this-
882 >GetDocument()->objects.GetSize(); i++) { // Iterate
883 through all the editors underneath the active editor
884
885             this->GetDocument()->objects[i]->move(0, this-
886 >GetDocument()->objects[this->activeEditor]->
887 >getBoxHeight()); // Move the editors down the
888 appropriate length
889
890             if (this->GetDocument()->objects[this-
891 >activeEditor]->iGetLineNum(1).size() < this-
892 >GetDocument()->objects[i]->iGetLineNum(1).size())
893             { // Check is the line to be changed is bigger,
894 as any line that is the same size or smaller than
895 the avtive editor will never need changed
896
897                 if (this->GetDocument()->objects[i]->
898 iGetLineNum(1)[this->GetDocument()->objects[this-
899 >activeEditor]->iGetLineNum(1).size() - 1] > /*Get
900 line 1's number of the iterated editor and take
901 the index that is the same as the last index in
902 the active editor line number's*/
903                     this->GetDocument()->objects[this-
904 >activeEditor]->iGetLineNum(this->GetDocument()->
905 >objects[this->activeEditor]->getActiveLine
906 ()).back()) /*Get the active line's number and
907 take the last index*/ {
908
909                 // Compare the numbers. See if the editor
910 needs to be incremented by having larger
```

```
    appropriate index
887
888            this->GetDocument()->objects[i]->incrementSublines(this->GetDocument()->objects
889            [this->activeEditor]->iGetLineNum(1).size() - 1,
890            1);
891        }
892        else if (this->GetDocument()->objects[i]->iGetLineNum(1)[this->GetDocument()->objects[this-
893            >activeEditor]->iGetLineNum(1).size() - 1] ==
894            this->GetDocument()->objects[this->activeEditor]->iGetLineNum(this->GetDocument()->objects[this-
895            >activeEditor]->getActiveLine().back()) {
896            // Compare the same numbers as in the previous statement but check for equality
897            if (increment) { // Check that an increment is appropriate
898                this->GetDocument()->objects[i]->incrementSublines(this->GetDocument()->objects
899                [this->activeEditor]->iGetLineNum(1).size() - 1,
900                1);
901            }
902            this->ResyncScrollSizes();
903            this->selectClippingRgn(RETURN_BACK);
904        }
905
906        break;
907    }
908
909    case VK_BACK: {
910
911        BOOL move = TRUE;
912        BOOL removeEditor = FALSE;
913
914        if (this->GetDocument()->objects[this->activeEditor]->getCaretPos() == 0 || this->GetDocument()->objects[this-
915            >activeEditor]->isHlghtMultiline()) { // If the caret is at the start of the line
916
917            std::vector<std::vector<int>> lines = { };
918
919            // Get the line number of the active line
920            if (this->GetDocument()->objects[this->activeEditor]->getCaretPos() == 0) {
921                lines.push_back(this->GetDocument()->objects[this->activeEditor]->iGetLineNum(this->GetDocument()->objects[this-
922                    >activeEditor]->getActiveLine() + 1));
923            }
924        }
925    }
926}
```

```
922         }
923
924         if (this->GetDocument()->objects[this->activeEditor]->
925             isHlghtMultiline()) {
926             BOOL first = FALSE;
927             BOOL last = FALSE;
928             int i = 0;
929
930             while (!last && i < this->GetDocument()->objects
931                   [this->activeEditor]->getNumLines()) {
932
933                 switch (this->GetDocument()->objects[this-
934                   >activeEditor]->lineHighlight(i)) {
935
936                     case 0:
937                         if (first) {
938                             last = TRUE;
939                         }
940                         break;
941
942                     case 1:
943                         if (first) {
944                             lines.push_back(this->GetDocument()->
945                               objects[this->activeEditor]->iGetLineNum(i + 1));
946                             last = TRUE;
947                         }
948                         else {
949                             first = TRUE;
950                         }
951                         break;
952
953                     case 2:
954                         if (!first) {
955                             first = TRUE;
956                         }
957
958                         lines.push_back(this->GetDocument()->
959                           objects[this->activeEditor]->iGetLineNum(i + 1));
960                         break;
961
962                     }
963
964                     for (auto& it : lines) {
965                         it.push_back(1);
966                     }
967
968                     // Check if the line wanting to delete has a subline
969                     for (int i = this->activeEditor + 1; i < this-
970                           >GetDocument()->objects.GetSize(); i++) {
971
972                         for (auto& line : lines) {
```

```
972             if (this->GetDocument()->objects[i]->iGetLineNum(1) == line) { →
973                 move = FALSE;
975
976                 // SEND A WARNING MESSAGE
977                 cout << "UNABLE TO PROCESS COMMAND" << endl; →
978             }
979         }
980     }
981
982     // If the line we want to delete does not have a
983     // subline...
984     if (move) {
985
986         // and we are not on the first line
987         if (this->GetDocument()->objects[this->activeEditor]->getActiveLine() != 0) {
988
989             // Move all the editors up one line space and
990             // increment them appropriatley
991             for (int i = this->activeEditor + 1; i < this->GetDocument()->objects.GetSize(); i++) {
992                 this->GetDocument()->objects[i]->move(0, -this->GetDocument()->objects[this->activeEditor]->getBoxHeight());
993
994                 if (this->GetDocument()->objects[this->activeEditor]->hasHighlight()) {
995
996                     if (this->GetDocument()->objects[i]->iGetLineNum(1)[this->GetDocument()->objects[this->activeEditor]->iGetLineNum(1).size() - 1] >
997                         this->GetDocument()->objects[this->activeEditor]->iGetLineNum(this->GetDocument()->objects[this->activeEditor]->getStartLine()).back()
998                     {
999                         this->GetDocument()->objects[i]->incrementSublines(this->GetDocument()->objects[this->activeEditor]->iGetLineNum(1).size() - 1, -1);
1000
1001                     else if (this->GetDocument()->objects[i]->iGetLineNum(1)[this->GetDocument()->objects[this->activeEditor]->iGetLineNum(1).size() - 1] >
1002                         this->GetDocument()->objects[this->activeEditor]->iGetLineNum(this->GetDocument()->objects[this->activeEditor]->getActiveLine())
1003                         () . back ()) {
1004
1005                         this->GetDocument()->objects[i]->incrementSublines(this->GetDocument()->objects
```

```

...\\source\\repos\\DesignArk\\DesignArk\\CTextDocView.cpp 24
    [this->activeEditor]->iGetLineNum(1).size() - 1, ↵
    -1);
1005         }
1006     }
1007 }
1008
1009 // and there is only one line, but it is not the ↵
1010 // first editor
1011 else if (this->GetDocument()->objects[this- ↵
1012 >activeEditor]->getNumLines() == 1 && this- ↵
1013 >activeEditor != 0 && !this->GetDocument()- ↵
1014 >objects[this->activeEditor]->hasHighlight()) {
1015
1016     // this means that the user want to delete this ↵
1017     // editor
1018
1019     this->selectClippingRgn(EDITOR_EDIT); // Will ↵
1020     draw everything under and including the removed ↵
1021     editor
1022
1023     // Remove the editor and move all the editors ↵
1024     back
1025
1026     for (int i = this->activeEditor + 1; i < this- ↵
1027 >GetDocument()->objects.GetSize(); i++) {
1028         this->GetDocument()->objects[i]->move(0, - ↵
1029         (this->GetDocument()->objects[this->activeEditor]- ↵
1030         >getBounds().Height() + this->GetDocument()- ↵
1031         >objects[this->activeEditor]->getBoxHeight()));
1032         }
1033         this->GetDocument()->objects.RemoveAt(this- ↵
1034         >activeEditor);
1035
1036         std::vector<int> oldLine = lines[0];
1037         lines.pop_back();
1038         lines.pop_back();
1039
1040         // TODO : Fix this so the parent line of the ↵
1041         // deleted editor becomes the active editor
1042         if (lines.size() > 0) {
1043
1044             BOOL found = FALSE;
1045             int i = 0;
1046
1047             while (!found) {
1048
1049                 found = TRUE;
1050             }
1051
1052             this->activeEditor--;
1053         }
1054
1055         else {
1056
1057             this->activeEditor--;
1058         }

```

```
1045
1046             this->GetDocument()->objects[this-
1047                 >activeEditor]->setActive(TRUE);
1048
1049
1050         // Adds the line that needs to be selected to the region
1051         CRgn rgn2;
1052         rgn2.CreateRectRgn( this->GetDocument()->objects[this->activeEditor]->getBounds().left,
1053                             this->GetDocument()->objects[this->activeEditor]->getLineBounds(this->GetDocument()->objects[this->activeEditor]->getActiveLine()).top,
1054                             this->GetDocument()->objects[this->activeEditor]->getBounds().right,
1055                             this->GetDocument()->objects[this->activeEditor]->getLineBounds(this->GetDocument()->objects[this->activeEditor]->getActiveLine()).bottom);
1056
1057         this->rgn->CombineRgn(this->rgn, &rgn2, RGN_OR);
1058     }
1059     else if(!this->GetDocument()->objects[this->activeEditor]->hasHighlight()) {
1060         move = FALSE;
1061     }
1062 }
1063 }
1064 if (move && !removeEditor) {
1065
1066     if(this->GetDocument()->objects[this->activeEditor]->hasHighlight()) {
1067
1068         if (this->GetDocument()->objects[this->activeEditor]->isHghtMultiline()) {
1069             this->selectClippingRgn(REMOVE_HIGHLIGHT);
1070             this->GetDocument()->objects[this->activeEditor]->OnRecieveBackspace();
1071             this->selectClippingRgn(ADD_RGN_ACTIVE_LINE);
1072         }
1073     else {
1074         this->GetDocument()->objects[this->activeEditor]->OnRecieveBackspace();
1075         this->selectClippingRgn(TEXT);
1076         this->selectClippingRgn(ADD_RGN_ACTIVE_LINE);
1077     }
1078
1079     if (this->GetDocument()->objects.GetSize() > this->activeEditor + 1) {
1080
1081         int dy = this->GetDocument()->objects[this->activeEditor + 1]->getBounds().top - (this->GetDocument()->objects[this->activeEditor]->
```

---

26

```

...\\source\\repos\\DesignArk\\DesignArk\\CTextDocView.cpp
    >getBounds().bottom + this->GetDocument()->objects[&this->activeEditor]->getBoxHeight());
1082
1083     for (int i = this->activeEditor + 1; i < this->GetDocument()->objects.GetSize(); i++) {
1084
1085         this->GetDocument()->objects[i]->move(0, -dy);
1086     }
1087 }
1088 }
1089 else {
1090
1091     if (this->GetDocument()->objects[this->activeEditor]->getCaretPos() == 0) {
1092         // Send backspace to editor AFTER setting the clipping region here
1093         this->selectClippingRgn(RETURN_BACK);
1094         this->GetDocument()->objects[this->activeEditor]->OnRecieveBackspace();
1095     }
1096     else {
1097         // Send backspace to editor BEFORE setting the clipping region here
1098         this->GetDocument()->objects[this->activeEditor]->OnRecieveBackspace();
1099         this->selectClippingRgn(TEXT);
1100     }
1101 }
1102 this->ResyncScrollSizes();
1103 }
1104
1105     break;
1106 }
1107
1108 case VK_TAB:
1109     this->GetDocument()->objects[this->activeEditor]->OnRecieveTab();
1110     break;
1111
1112 default:
1113     if (this->textInput.OnKeyDown(nChar, nRepCnt, nFlags)) {
1114         if (this->GetDocument()->objects[this->activeEditor]->hasHighlight()) {
1115
1116             this->OnKeyDown(VK_BACK, 1, 0);
1117         }
1118     else {
1119
1120         // Set the clipping region for basic text input, with no highlighting
1121         this->selectClippingRgn(TEXT);
1122     }
1123
1124     this->GetDocument()->objects[this->activeEditor]->OnRecieveText(this->textInput.RecieveText());

```

```
1125
1126
1127
1128         this->ResyncScrollSizes();
1129     }
1130     else {
1131         int oldLine = this->GetDocument()->objects[this-
1132             >activeEditor]->getActiveLine();
1133         int oldCaretPos = this->GetDocument()->objects[this-
1134             >activeEditor]->getCaretPos();
1135
1136         this->GetDocument()->objects[this->activeEditor]->
1137             OnKeyDown(nChar, nRepCnt, nFlags);
1138
1139         if (oldLine != this->GetDocument()->objects[this-
1140             >activeEditor]->getActiveLine()) {
1141             this->recentLine = oldLine;
1142             this->selectClippingRgn(SELECT_LINE);
1143         }
1144         else if (oldCaretPos != this->GetDocument()->objects
1145             [this->activeEditor]->getCaretPos()) {
1146             this->selectClippingRgn(CARET_MOVE);
1147             redraw = FALSE;
1148         }
1149     }
1150
1151     if (redraw) {
1152         this->updateWindow(FALSE, TRUE);
1153     }
1154 }
1155 void CTextDocView::OnKeyUp(UINT nChar, UINT nRepCnt, UINT nFlags)
1156 {
1157     this->textInput.OnKeyUp(nChar, nRepCnt, nFlags);
1158 }
1159
1160 void CTextDocView::OnContextMenu(CWnd* pWnd, CPoint point)
1161 {
1162     point -= this->GetScrollPosition();
1163
1164     this->GetDocument()->objects[this->activeEditor]->OnContextMenu(pWnd, point);
1165 }
1166
1167 void CTextDocView::OnCopy()
1168 {
1169     if (this->GetDocument()->objects[this->activeEditor]->
1170         hasHighlight()) {
1171         if (!OpenClipboard())
1172             AfxMessageBox(_T("Cannot open the Clipboard"));
1173         return;
```

```
1174         }
1175
1176         // Remove the current Clipboard contents
1177         if (!EmptyClipboard())
1178         {
1179             AfxMessageBox(_T("Cannot empty the Clipboard"));
1180             return;
1181         }
1182
1183         // Get the currently selected data
1184         CStringA text(this->GetDocument()->objects[this-
1185             >activeEditor]->getHighlightedText()); ↗
1186
1187         const char* x = text;
1188
1189         HGLOBAL hGlob = GlobalAlloc(GMEM_FIXED, text.GetLength() * ↗
1190             2);
1191
1192         strcpy_s((char*)hGlob, text.GetLength() * 2, x);
1193
1194         // For the appropriate data formats...
1195         if (::SetClipboardData(CF_TEXT, hGlob) == NULL)
1196         {
1197             CString msg;
1198             msg.Format(_T("Unable to set Clipboard data, error: % ↗
1199                 d"), GetLastError());
1200             AfxMessageBox(msg);
1201             CloseClipboard();
1202             GlobalFree(hGlob);
1203             return;
1204         }
1205     void CTextDocView::OnPaste()
1206     {
1207         HANDLE hGlob;
1208
1209         // If the application is in edit mode,
1210         // get the clipboard text.
1211
1212         if (!IsClipboardFormatAvailable(CF_TEXT)) {
1213             return;
1214         }
1215
1216         if (!OpenClipboard()) {
1217             AfxMessageBox(_T("Cannot open the Clipboard"));
1218             return;
1219         }
1220
1221         hGlob = GetClipboardData(CF_TEXT);
1222         if (hGlob != NULL)
1223         {
1224             if (this->GetDocument()->objects[this->activeEditor]-<↗
1225                 >hasHighlight()) {
1226                 this->GetDocument()->objects[this->activeEditor]-<↗
```

```
1226         }
1227
1228         this->recentLine = this->GetDocument()->objects[this-
1229             >activeEditor]->getActiveLine();
1230
1231         int carriageReturn = this->GetDocument()->objects[this-
1232             >activeEditor]->OnRecieveText(CString((char*)GlobalLock
1233                 (hGlob)), TRUE);
1234
1235         for (int i = this->activeEditor + 1; i < this->GetDocument()
1236             ()->objects.GetSize(); i++) {
1237
1238             this->GetDocument()->objects[i]->move(0, this-
1239                 >GetDocument()->objects[this->activeEditor]-
1240                     >getBoxHeight() * carriageReturn);
1241
1242             GlobalUnlock(hGlob);
1243         }
1244
1245         AfxMessageBox(_T("Clipboard is Empty"));
1246
1247         return;
1248     }
1249
1250 }
1251 void CTextDocView::OnCut()
1252 {
1253     this->OnCopy();
1254
1255     if (this->GetDocument()->objects[this->activeEditor]-
1256         >hasHighlight()) {
1257         this->OnKeyDown(VK_BACK, 1, 0);
1258     }
1259
1260 void CTextDocView::expandLineBounds()
1261 {
1262     CRect client;
1263     this->GetClientRect(&client);
1264
1265     CRect newBounds;
1266
1267     for (int i = 0; i < this->GetDocument()->objects.GetSize(); i+
1268         +) {
1269
1270         newBounds = this->GetDocument()->objects[i]->getBounds();
1271         newBounds.right = max(client.right + this-
1272             >GetScrollPosition().x, this->GetDocument()->objects[i]-
1273                 >getLineTextWidth() + static_cast<int>(this->GetDocument()
1274                     ()->lineOffset * theApp.zoom));
```

```
1270         this->GetDocument()->objects[i]->setBounds(newBounds);
1271     }
1272 }
1273 }
1274
1275 void CTextDocView::selectClippingRgn(int nAction, int type)
1276 {
1277     int x1 = 0, y1 = 0, x2 = 0, y2 = 0;
1278
1279     switch (nAction) {
1280
1281     case TEXT: { // Working
1282
1283         if (this->recentHlght) {
1284             x2 = this->GetDocument()->objects[this->activeEditor]->
1285                 getBounds().left + this->GetDocument()->lineOffset + (
1286                     ((this->GetDocument())->objects[this->activeEditor]->
1287                         getLineText(this->GetDocument())->objects[this-
1288                             >activeEditor]->getActiveLine() + 1).GetLength() + 1) *
1289                     this->GetDocument()->objects[this->activeEditor]->
1290                         getTextExtent().cx) - this->GetScrollPosition().x;
1291         }
1292         else {
1293             x2 = this->GetDocument()->objects[this->activeEditor]->
1294                 getLineBounds(this->GetDocument())->objects[this-
1295                     >activeEditor]->getActiveLine().right - this->
1296                         GetScrollPosition().x;
1297         }
1298
1299         int drawpos = this->GetDocument()->objects[this-
1300             >activeEditor]->getCaretPos();
1301
1302         int i = this->GetDocument()->objects[this->activeEditor]->
1303             getCaretPos();
1304         BOOL end = FALSE;
1305
1306         while (i >= 0 && !end) {
1307
1308             if (this->GetDocument())->objects[this->activeEditor]->
1309                 getLineText().Mid(i, 1) == L" " || i == 0) {
1310                 drawpos = i;
1311                 end = TRUE;
1312             }
1313
1314             i--;
1315         }
1316
1317         x1 = this->GetDocument()->objects[this->activeEditor]->
1318             getBounds().left + this->GetDocument()->lineOffset +
1319             (drawpos * this->GetDocument())->objects[this-
1320                 >activeEditor]->getTextExtent().cx) - this->
1321                     GetScrollPosition().x;
1322
1323         y1 = this->GetDocument())->objects[this->activeEditor]->
1324             getLineBounds(this->GetDocument())->objects[this-
1325                 >activeEditor]->getActiveLine().top - this->
1326                     GetScrollPosition().y;
```

...\\source\\repos\\DesignArk\\DesignArk\\CTextDocView.cpp

---

```

1307         y2 = this->GetDocument()->objects[this->activeEditor]->getLineBounds(this->GetDocument()->objects[this->activeEditor]->getActiveLine()).bottom - this->GetScrollPosition().y;
1308
1309         VERIFY(this->rgn->CreateRectRgn(x1, y1, x2, y2));
1310         break;
1311     }
1312
1313     case RETURN_BACK: // Working
1314
1315         x1 = this->GetDocument()->objects[this->activeEditor]->getBounds().left - this->GetScrollPosition().x;
1316         y1 = this->GetDocument()->objects[this->activeEditor]->getLineBounds(this->GetDocument()->objects[this->activeEditor]->getActiveLine()-1).top - this->GetScrollPosition().y;
1317         x2 = this->GetDocument()->objects[this->GetDocument()->objects.GetSize() - 1]->getBounds().right - this->GetScrollPosition().x;
1318         y2 = this->GetDocument()->objects[this->GetDocument()->objects.GetSize() - 1]->getBounds().bottom + this->GetDocument()->objects[this->activeEditor]->getBoxHeight() - this->GetScrollPosition().y;
1319         VERIFY(this->rgn->CreateRectRgn(x1, y1, x2, y2));
1320         break;
1321
1322     case EDITOR_EDIT: { // Working
1323         x1 = this->GetDocument()->objects[this->activeEditor]->getBounds().left - this->GetScrollPosition().x;
1324         y1 = this->GetDocument()->objects[this->activeEditor]->getBounds().top - this->GetScrollPosition().y;
1325         x2 = this->GetDocument()->objects[this->activeEditor]->getBounds().right - this->GetScrollPosition().x;
1326         y2 = this->GetDocument()->objects[this->GetDocument()->objects.GetSize() - 1]->getBounds().bottom - this->GetScrollPosition().y;
1327         VERIFY(this->rgn->CreateRectRgn(x1, y1, x2, y2));
1328
1329         CRgn* rgn1 = new CRgn();
1330         CRect tempRect = this->GetDocument()->objects[this->recentEditor]->getLineBounds(this->GetDocument()->objects[this->recentEditor]->getActiveLine());
1331
1332         x1 = tempRect.left;
1333         y1 = tempRect.top;
1334         x2 = tempRect.right;
1335         y2 = tempRect.bottom;
1336
1337         VERIFY(rgn1->CreateRectRgn(x1, y1, x2, y2));
1338         this->rgn->CombineRgn(this->rgn, rgn1, RGN_OR);
1339
1340         this->selectClippingRgn(ADD_RGN_SIDEBAR_BTN, 1);
1341
1342         break;
1343     }

```

```
1344
1345     case HIGHLIGHTING: { // Working
1346         CRect temp = this->GetDocument()->objects[this-
1347             >activeEditor]->getHighlightClippingRect();
1348         x1 = temp.left - this->GetScrollPosition().x;
1349         y1 = temp.top - this->GetScrollPosition().y;
1350         x2 = temp.right - this->GetScrollPosition().x;
1351         y2 = temp.bottom - this->GetScrollPosition().y;
1352         VERIFY(this->rgn->CreateRectRgn(x1, y1, x2, y2));
1353         break;
1354     }
1355
1356     case SELECT_LINE: { // Working
1357         if (this->GetDocument()->objects[this->activeEditor]-
1358             >getActiveLine() != this->recentLine) {
1359             CRgn* rgn1 = new CRgn();
1360             CRgn* rgn2 = new CRgn();
1361             CRgn* tempRgn = new CRgn();
1362
1363             CRect temp = this->GetDocument()->objects[this-
1364                 >recentEditor]->getLineBounds(this->recentLine);
1365
1366             x1 = temp.left - this->GetScrollPosition().x;
1367             y1 = temp.top - this->GetScrollPosition().y;
1368             x2 = temp.right - this->GetScrollPosition().x;
1369             y2 = temp.bottom - this->GetScrollPosition().y;
1370
1371             VERIFY(rgn1->CreateRectRgn(x1, y1, x2, y2));
1372             x1 += 1;
1373             y1 += 1;
1374             x2 -= 1;
1375             y2 -= 1;
1376
1377             VERIFY(tempRgn->CreateRectRgn(x1, y1, x2, y2));
1378
1379             int nCombineResult = rgn1->CombineRgn(rgn1, tempRgn,
1380                 RGN_XOR);
1381             ASSERT(nCombineResult != ERROR && nCombineResult !=
1382                 NULLREGION);
1383
1384             tempRgn = new CRgn();
1385
1386             x1 = this->GetCaretPos().x;
1387             x2 = x1 + this->GetDocument()->objects[0]-
1388                 >getTextExtent().cx*0.25;
1389
1390             VERIFY(tempRgn->CreateRectRgn(x1, y1, x2, y2));
1391
1392             nCombineResult = rgn1->CombineRgn(rgn1, tempRgn,
1393                 RGN_XOR);
1394             ASSERT(nCombineResult != ERROR && nCombineResult !=
1395                 NULLREGION);
1396
1397             tempRgn = new CRgn();
```

```
1392
1393     temp = this->GetDocument()->objects[this-
1394         >activeEditor]->getLineBounds(this->GetDocument()->
1395             objects[this->activeEditor]->getActiveLine());
1396
1397     x1 = temp.left - this->GetScrollPosition().x;
1398     y1 = temp.top - this->GetScrollPosition().y;
1399     x2 = temp.right - this->GetScrollPosition().x;
1400     y2 = temp.bottom - this->GetScrollPosition().y;
1401
1402     VERIFY(rgn2->CreateRectRgn(x1, y1, x2, y2));
1403
1404     x1 += 1;
1405     y1 += 1;
1406     x2 -= 1;
1407     y2 -= 1;
1408
1409     VERIFY(tempRgn->CreateRectRgn(x1, y1, x2, y2));
1410
1411     nCombineResult = rgn2->CombineRgn(rgn2, tempRgn,
1412         RGN_XOR);
1413     ASSERT(nCombineResult != ERROR && nCombineResult !=
1414         NULLREGION);
1415
1416     VERIFY(this->rgn->CreateRectRgn(0, 0, 0, 0)); // Must
1417         initialise the rgn first
1418     nCombineResult = this->rgn->CombineRgn(rgn1, rgn2,
1419         RGN_OR);
1420
1421     ASSERT(nCombineResult != ERROR && nCombineResult !=
1422         NULLREGION);
1423
1424     if (this->recentHlght) {
1425
1426         CRgn* rgn3 = this->GetDocument()->objects[this-
1427             >recentEditor]->getHighlightExactRgn(this-
1428                 >GetScrollPosition().x, this->GetScrollPosition()
1429                     .y);
1430
1431         nCombineResult = this->rgn->CombineRgn(this->rgn,
1432             rgn3, RGN_OR);
1433
1434         ASSERT(nCombineResult != ERROR && nCombineResult !=
1435             NULLREGION);
1436     }
1437
1438     else if (this->recentHlght) {
1439
1440         CRect temp = this->GetDocument()->objects[this-
1441             >activeEditor]->getLineBounds(this->GetDocument()->
1442                 objects[this->activeEditor]->getActiveLine());
1443
1444         x1 = temp.left - this->GetScrollPosition().x;
1445         y1 = temp.top - this->GetScrollPosition().y;
```

```
1434         x2 = temp.right - this->GetScrollPosition().x;
1435         y2 = temp.bottom - this->GetScrollPosition().y;
1436
1437         VERIFY(this->rgn->CreateRectRgn(x1, y1, x2, y2));
1438
1439         CRgn* rgn3 = this->GetDocument()->objects[this-
1440             >recentEditor]->getHighlightExactRgn(this-
1441             >GetScrollPosition().x, this->GetScrollPosition().y);
1442
1443         int nCombineResult = this->rgn->CombineRgn(this->rgn,
1444             rgn3, RGN_OR);
1445
1446     else {
1447
1448         x1 = (this->GetDocument()->objects[this->activeEditor]->
1449             getRecentPos() - 0.5) * this->GetDocument()->objects[
1450                 [this->activeEditor]->getTextExtent().cx ;
1451
1452         y1 = this->GetDocument()->objects[this->activeEditor]->
1453             getLineBounds(this->GetDocument()->objects[this-
1454                 >activeEditor]->getActiveLine()).top;
1455
1456         x2 = x1 + this->GetDocument()->objects[this-
1457             >activeEditor]->getTextExtent().cx;
1458
1459         y2 = this->GetDocument()->objects[this->activeEditor]->
1460             getLineBounds(this->GetDocument()->objects[this-
1461                 >activeEditor]->getActiveLine()).bottom;
1462
1463         this->rgn->CreateRectRgn(x1, y1, x2, y2);
1464     }
1465
1466     break;
1467 }
1468
1469 case REMOVE_HIGHLIGHT: // Working
1470     x1 = this->GetDocument()->objects[this->activeEditor]->
1471         getBounds().left - this->GetScrollPosition().x;
1472
1473     y1 = this->GetDocument()->objects[this->activeEditor]->
1474         getLineBounds(min(this->GetDocument()->objects[this-
1475             >activeEditor]->getActiveLine(), this->GetDocument()->
1476             objects[this->activeEditor]->getStartLine())).top -
1477             this->GetScrollPosition().y;
1478
1479     x2 = this->GetDocument()->objects[this->GetDocument()->
1480         objects.GetSize() - 1]->getBounds().right - this-
1481         >GetScrollPosition().x;
1482
1483     y2 = this->GetDocument()->objects[this->GetDocument()->
1484         objects.GetSize() - 1]->getBounds().bottom - this-
1485         >GetScrollPosition().y;
1486
1487     VERIFY(this->rgn->CreateRectRgn(x1, y1, x2, y2));
1488     break;
1489
1490 case CARET_MOVE: // Working
1491
1492     x1 = this->GetCaretPos().x - this->GetDocument()->objects
```

```
[this->activeEditor]->getTextExtent().cx - this->GetScrollPosition().x;
1470    y1 = this->GetDocument()->objects[this->activeEditor]->getLineBounds(this->GetDocument()->objects[this->activeEditor]->getActiveLine()).top - this->GetScrollPosition().y;
1471    x2 = this->GetCaretPos().x + this->GetDocument()->objects[this->activeEditor]->getLineBounds(this->GetDocument()->objects[this->activeEditor]->getActiveLine()).bottom - this->GetScrollPosition().y;
1472
1473    VERIFY(this->rgn->CreateRectRgn(x1, y1, x2, y2));
1474    break;
1475
1476
1477 case ADD_RGN_SIDEBAR_BTN: { // Working
1478
1479     if (type != 0 && type != 1) {
1480         throw("ERROR : type error. type must be 0 or 1");
1481     }
1482
1483     Gdiplus::Bitmap bitmap(L"res/arrow_right_click.bmp");
1484
1485     x1 = this->GetDocument()->objects[this->recentEditor]->getBounds().left + static_cast<int>(this->GetDocument()->defBoxHeight * theApp.zoom) / 2 - (bitmap.GetWidth() * theApp.zoom) / 2;
1486     y1 = this->GetDocument()->objects[this->recentEditor]->getLineBounds(this->recentBlockLine).top + static_cast<int>(this->GetDocument()->defBoxHeight * theApp.zoom) / 2 - (bitmap.GetHeight() * theApp.zoom) / 2 - this->GetScrollPosition().y;
1487     x2 = x1 + (bitmap.GetWidth() * theApp.zoom);
1488     y2 = y1 + (bitmap.GetHeight() * theApp.zoom) - this->GetScrollPosition().y;
1489
1490     if (type == 0) {
1491         VERIFY(this->rgn->CreateRectRgn(x1, y1, x2, y2));
1492     }
1493     else {
1494         CRgn* rgn1 = new CRgn();
1495         VERIFY(rgn1->CreateRectRgn(x1, y1, x2, y2));
1496         this->rgn->CombineRgn(this->rgn, rgn1, RGN_OR);
1497     }
1498
1499     break;
1500 }
1501 case ADD_RGN_ACTIVE_LINE: {
1502     CRgn* rgn1 = new CRgn();
1503     CRect temp = this->GetDocument()->objects[this->recentEditor]->getLineBounds(this->GetDocument()->objects[this->activeEditor]->getActiveLine()));
1504
1505     x1 = temp.left - this->GetScrollPosition().x;
```

```

...\\source\\repos\\DesignArk\\DesignArk\\CTextDocView.cpp
1506         y1 = temp.top - this->GetScrollPosition().y;
1507         x2 = temp.right - this->GetScrollPosition().x;
1508         y2 = temp.bottom - this->GetScrollPosition().y;
1509
1510         VERIFY(rgn1->CreateRectRgn(x1, y1, x2, y2));
1511
1512         this->rgn->CombineRgn(this->rgn, rgn1, RGN_OR);
1513
1514         break;
1515     }
1516
1517     case H_SCROLL:
1518
1519         x1 = 0;
1520         y1 = this->GetDocument()->objects[0]->getBounds().top;
1521         x2 = this->GetScrollPosition().x + this->GetDocument()->lineOffset;
1522         y2 = this->GetDocument()->objects[this->GetDocument()->objects.GetSize() - 1]->getBounds().bottom;
1523
1524         VERIFY(this->rgn->CreateRectRgn(x1, y1, x2, y2));
1525         break;
1526
1527     case PASTE:
1528
1529         x1 = this->GetDocument()->objects[this->activeEditor]->getBounds().left - this->GetScrollPosition().x;
1530         y1 = this->GetDocument()->objects[this->activeEditor]->getLineBounds(this->GetDocument()->objects[this->activeEditor]->getActiveLine()).top - this->GetScrollPosition().y;
1531         x2 = this->GetDocument()->objects[this->activeEditor]->getBounds().right - this->GetScrollPosition().x;
1532         y2 = this->GetDocument()->objects[this->GetDocument()->objects.GetSize() - 1]->getBounds().bottom - this->GetScrollPosition().y;
1533
1534         VERIFY(this->rgn->CreateRectRgn(x1, y1, x2, y2));
1535         break;
1536     }
1537 //cout << x1 << ", " << y1 << ", " << x2 << ", " << y2 << endl;// KEEP THIS FOR DEBUGGING
1538 }
1539
1540 void CTextDocView::drawHeader(CDC* pDc)
1541 {
1542     pDc->TextOut(this->GetDocument()->objects[0]->getLineBounds().left, 400 - this->GetDocument()->objects[0]->getTextExtent().cy,
1543                     this->GetDocument()->filename, this->GetDocument()->filename.getLength());
1544
1545     CString page;
1546     page.Format(L"%d", this->pInfo->m_nCurPage);
1547
1548     pDc->TextOut(this->pInfo->m_rectDraw.Width() - (this-

```

```
    >GetDocument()->objects[0]->getLineBounds().left - this->
    >GetDocument()->objects[0]->getBounds().left) / 2 -
    page.GetLength() * this->GetDocument()->objects[0]-
    >getTextExtent().cx, 400 - this->GetDocument()->objects[0]-
    >getTextExtent().cy, page, page.GetLength());
1549
1550     pDc->MoveTo(CPoint((this->GetDocument()->objects[0]-
    >getLineBounds().left - this->GetDocument()->objects[0]-
    >getBounds().left)/2, 400));
1551     pDc->LineTo(CPoint(this->pInfo->m_rectDraw.Width() - (this-
    >GetDocument()->objects[0]->getLineBounds().left - this-
    >GetDocument()->objects[0]->getBounds().left) / 2, 400));
1552 }
1553 }
```

```
1 // CTextDocument.h : interface of the CTextDocument class
2 //
3 //
4 #pragma once
5 #include "CTextEditorObject.h"
6
7 class CTextDocument : public CDocument
8 {
9 public:
10     // Public constructors
11     CTextDocument() noexcept;
12     virtual ~CTextDocument();
13
14     DECLARE_DYNAMIC(CTextDocument);
15
16     // Public overrides
17     virtual BOOL OnNewDocument();
18     virtual void Serialize(CArchive& ar);
19
20     virtual void OnFileSave();
21     virtual void OnFileOpen();
22
23 #ifdef SHARED_HANDLERS
24     virtual void OnDrawThumbnail(CDC& dc, LPRECT lprcBounds);
25     virtual void InitializeSearchContent();
26 #endif // !SHARED_HANDLERS
27
28     // Public implementations
29 #ifdef _DEBUG
30     virtual void AssertValid() const;
31     virtual void Dump(CDumpContext& dc) const;
32 #endif
33
34     // Document Data
35     CArray<CTextEditorObject*> objects;
36     int defBoxHeight;
37     int lineOffset;
38
39     CString filename;
40
41 protected:
42     // Protected message handlers
43     DECLARE_MESSAGE_MAP()
44
45     // Protected commands
46 #ifdef SHARED_HANDLERS
47     void SetSearchContent(const CString& value);
48 #endif // !SHARED_HANDLERS
49
50 };
51
```

```
1 // CTextDocument.cpp : implementation of the CTextDocument class
2 //
3 //
4 #include "pch.h"
5 #include "framework.h"
6 // SHARED_HANDLERS can be defined in an ATL project implementing
7 // preview, thumbnail
8 // and search filter handlers and allows sharing of document code
9 // with that project.
10 #ifndef SHARED_HANDLERS
11 #include "DesignArk.h"
12 #endif
13
14 #include "CTextDocument.h"
15 #include "ChildFrm.h"
16 #include "CTextDocView.h"
17
18 #include <propkey.h>
19
20 #ifdef _DEBUG
21 #define new DEBUG_NEW
22 #endif
23
24 // CTextDocument
25 IMPLEMENT_DYNCREATE(CTextDocument, CDocument)
26
27 BEGIN_MESSAGE_MAP(CTextDocument, CDocument)
28     ON_COMMAND(ID_FILE_SAVE, &CTextDocument::OnFileSave)
29     ON_COMMAND(ID_FILE_OPEN, &CTextDocument::OnFileOpen)
30 END_MESSAGE_MAP()
31
32 // Public constructors
33 CTextDocument::CTextDocument() noexcept
34 {
35     this->SetTitle(L"Pseudocode Editor");
36     std::vector<CString> text = { L"" };
37
38     this->defBoxHeight = 20;
39     this->lineOffset = 100;
40     this->filename = L"local";
41
42     CRect editorBounds;
43     editorBounds.top = 0;
44     editorBounds.bottom = this->defBoxHeight * theApp.zoom *
45         (text.size());
46     editorBounds.left = 0;
47     editorBounds.right = 1000;
48
49     std::vector <int> lineNums = { 1 };
50
51     this->objects.Add(new CTextEditorObject(editorBounds, L"1",
52         TRUE, 0, this->defBoxHeight * theApp.zoom, TRUE, text,
53         lineNums, this->lineOffset));
54 }
```

```
52 CTextDocument::~CTextDocument()
53 {
54     for (int i = 0; i < this->objects.GetSize(); i++) {
55         delete this->objects[i];
56     }
57 }
58
59 // Public overrides
60 BOOL CTextDocument::OnNewDocument()
61 {
62     if (!CDocument::OnNewDocument())
63         return FALSE;
64
65     // TODO: add reinitialization code here
66     // (SDI documents will reuse this document)
67
68     return TRUE;
69 }
70 void CTextDocument::Serialize(CArchive& ar)
71 {
72     if (ar.IsStoring())
73     {
74
75     }
76     else
77     {
78
79     }
80 }
81
82 void CTextDocument::OnFileSave()
83 {
84     const TCHAR szFilter[] = _T("Text Files (*.txt) | *.txt | All File | *.* | ");
85
86     CFileDialog fileDialog(FALSE, _T("txt"), NULL, OFN_HIDEREADONLY | OFN_FILEMUSTEXIST, szFilter);
87
88     if (fileDialog.DoModal() == IDOK) {
89
90         CFile file;
91
92         if (file.Open(fileDialog.GetFolderPath() + L"\\" + fileDialog.GetFileName(), CFile::modeCreate | CFile::modeWrite)) {
93
94             CString data = L"";
95
96             int max_size = this->objects[0]->sGetLineNum(this->objects[0]->getNumLines()).GetLength();
97
98             for (int i = 1; i < this->objects.GetSize(); i++) {
99                 if (this->objects[i]->sGetLineNum(1).GetLength() > max_size) {
100                     max_size = this->objects[i]->sGetLineNum(this->objects[i]->getNumLines()).GetLength();
101
102                 }
103             }
104
105             file.Write(data, max_size);
106
107             file.Close();
108
109         }
110
111     }
112 }
```

```
101         }
102     }
103
104     for (int i = 0; i < this->objects.GetSize(); i++) {
105         for (int j = 1; j < this->objects[i]->getNumLines() ↪
106             + 1; j++) {
107             for (int k = 0; k < (max_size - this->objects ↪
108 [i]->sGetLineNum(j).GetLength()); k++) {
109                 data.Append(L" ");
110             }
111             data.Append(this->objects[i]->sGetLineNum(j));
112             data.Append(L"      ");
113             data.Append(this->objects[i]->getLineText(j));
114             data.Append(L"\n");
115         }
116         data.Append(L"\n\n");
117     }
118
119     file.Write(data.GetBuffer(), data.GetLength()*2);
120
121     this->filename = fileDialog.GetFileName();
122 }
123 }
124 }
125 void CTextDocument::OnFileOpen()
126 {
127     const TCHAR szFilter[] = _T("Text Files (*.txt)|*.txt| All File|*.*|"); // Set a filter for file types
128
129     CFileDialog fileDialog(TRUE, _T("*.txt"), NULL, OFN_HIDEREADONLY ↪
130     | OFN_FILEMUSTEXIST, szFilter); // Instantiate file dialog box
131
132     if (fileDialog.DoModal() == IDOK) { // Open the file dialog and wait for the user to be done
133
134         CString filePath = fileDialog.GetFolderPath() + L"\\" + fileDialog.GetFileName(); // Create the file path
135         CDocument* newDoc = theApp.OpenDocumentFile(filePath); // Open a new document for the file
136
137         CFile file; // Open the file pathway
138
139         if (file.Open(filePath, CFile::modeRead)) { // Read through the file
140
141             // Read the text from the file
142
143             int iFileSize = file.GetLength(); // Getting the content length
144             BYTE* pData = new BYTE[iFileSize]; // To save the data in
145
146             file.Read(pData, iFileSize); // Reading file content
```

```
146         pData[iFileSize] = '\0';           // Add last character ↵
147             as NULL
148
149         file.Close(); // Close the file
150
151         CString character;
152         CString data;
153
154         int k = 0;
155
156         for (int i = 0; i < iFileSize; i++) { // Iterate through ↵
157             all the readable characters
158
159             character.Format(L"%C", pData[i]); // Format each ↵
160                 character correctly
161             data += character; // Add the character to the ↵
162                 current line (data)
163
164             if (character == L"\n" || i == iFileSize-1) { // Once ↵
165                 we have received a new space or have gotten to the ↵
166                 end of the file
167
168             int j = 0;
169             BOOL end = FALSE; // Iterators
170
171             int space = 0;
172             BOOL startNum = FALSE;
173
174             CString subnum = L"";  

175             std::vector<int>writingLineNum = {};
176
177             while (j < data.GetLength() && !end) { // ↵
178                 Iterate through the collected line
179
180                 CString active = data.Mid(j, 1); // Is the ↵
181                     secondary iterated character
182
183                 if (active == L"0" ||
184                     active == L"1" ||
185                     active == L"2" ||
186                     active == L"3" ||
187                     active == L"4" ||
188                     active == L"5" ||
189                     active == L"6" ||
190                     active == L"7" ||
191                     active == L"8" ||
192                     active == L"9") { // Once we have found ↵
193                         a number ie, where the line number is in the data
194
195                         j++;
196                         startNum = TRUE; // This tells us that ↵
197                         we have found start of the line number
198
199                         subnum.Append(active);
200
201 }
```

```

...source\repos\DesignArk\DesignArk\CTextDocument.cpp 5
192             else if (active == L"." && space == 0) { // ↵
Once we have reached the next subline
193                 j++;
194                 writingLineNum.push_back(_ttoi(subnum)); ↵

195                     subnum = L"";
196                 }
197                 else if (active == L" " && space < 4/*There ↵
should be 4 spaces between the end of the line ↵
number and the start of the text*/) { // If we have ↵
not reached the start of the actual text, either ↵
before or after the subline
198
199                 j++;
200
201                     if (startNum) { // If we have found the ↵
start of the line number we can start incrementing ↵
space
202
203                     space++;
204                 }
205             }
206             else { // This means that we have gotten to ↵
the start of the text
207
208                 if (subnum != L "") {
209                     writingLineNum.push_back(_ttoi
(subnum));
210                     }
211                     subnum = L "";
212                     end = TRUE;
213                 }
214             }
215
216
217             BOOL newEdit = FALSE;
218
219             if (writingLineNum.size() > 0) { // If there is ↵
not a gap in the line numbers
220
221                 std::vector<int>writingTemp = ↵
writingLineNum;
222                     writingTemp.pop_back(); // Remove the last ↵
subline in order to search for a parent line
223
224                 std::vector<int>itTemp;
225                 int i = ((CTextDocument*)newDoc)-
>objects.GetSize() - 1; // Find the number of ↵
objects created so far
226                 BOOL fin = FALSE; // Iterators
227
228                 while (i >= 0 && !fin) { // Iterate through ↵
all the objects in the new doc, in reverse
229
230                     itTemp = ((CTextDocument*)newDoc)-
>objects[i]->iGetLineNum(1);

```

```

...source\repos\DesignArk\DesignArk\CTextDocument.cpp 6
231             itTemp.pop_back(); // Get the line ↵
232             number and remove the recent subline
233             if (writingTemp == itTemp) { // Check if ↵
234             this is the editor we are searching
235             fin = TRUE; // If it is, say we ↵
236             found it with this
237             }
238             i--;
239             }
240             }
241             if (!fin) {
242             newEdit = TRUE; // If we did not find an ↵
243             editor that our line belongs to, create a new one
244             }
245             if (newEdit) {
246             // If we get here, it means that the new ↵
247             line should be made in a new editor
248             CRect prevBounds = ((CTextDocument*) ↵
249             newDoc)->objects[k]->getBounds(); // Find the ↵
250             bounds of the editor that lies above the one we are ↵
251             about to create
252             CRect bounds = prevBounds; // Create ↵
253             bounds for the new object
254             bounds.top = prevBounds.bottom + ↵
255             ((CTextDocument*) newDoc)->objects[k]->getBoxHeight ↵
256             (); // Move the top
257             bounds.bottom = bounds.top + ↵
258             ((CTextDocument*) newDoc)->objects[k]->getBoxHeight ↵
259             (); // Move the bottom
260             CString id;
261             id.Format(_T("%d", k + 1);
262             ((CTextDocument*) newDoc)->objects[k]->OnRecieveBackspace();
263             ((CTextDocument*) newDoc)->objects.Add( ↵
264             new CTextEditorObject(bounds, id, ↵
265             TRUE, 0, ((CTextDocument*) newDoc)->defBoxHeight, ↵
266             FALSE, ↵
267             std::vector<CString>{data.Mid
(j)}, writingLineNum,
                this->lineOffset)
            );
268             ((CTextDocument*) newDoc)->objects[k + 1]->OnRecieveReturn(TRUE);
269             ((CTextDocument*) newDoc)->objects[k]->setActive(FALSE);

```

```
268                     k++;
269                     //((CTextDocument*)newDoc)->objects[k]->
270                     >OnRecieveReturn(TRUE);
271                     }
272                     else {
273                         ((CTextDocument*)newDoc)->objects[k]->
274                         >OnRecieveText(data.Mid(j), TRUE); // Send the data
275                         through to the current editor
276                         //((CTextDocument*)newDoc)->objects[k]->
277                         >OnRecieveReturn(TRUE);
278                     }
279                     }
280                 }
281             }
282             ((CTextDocument*)newDoc)->objects[k]->OnRecieveBackspace();
283             ((CTextDocument*)newDoc)->objects[k]->setActive(FALSE);
284             ((CTextDocument*)newDoc)->objects[0]->setActive(TRUE);
285
286             ((CTextDocView*)theApp.m_ActiveView)->refresh();
287
288             ((CTextDocument*)newDoc)->filename =
289                 openFileDialog.GetFileName();
290             }
291         }
292
293 #ifdef SHARED_HANDLERS
294
295 // Support for thumbnails
296 void CTextDocument::OnDrawThumbnail(CDC& dc, LPRECT lprcBounds)
297 {
298     // Modify this code to draw the document's data
299     dc.FillSolidRect(lprcBounds, RGB(0, 0, 255));
300
301     CString strText = _T("TODO: implement thumbnail drawing here");
302     LOGFONT lf;
303
304     CFont* pDefaultGUIFont = CFont::FromHandle((HFONT)
305         GetStockObject(DEFAULT_GUI_FONT));
306     pDefaultGUIFont->GetLogFont(&lf);
307     lf.lfHeight = 36;
308
309     CFont fontDraw;
310     fontDraw.CreateFontIndirect(&lf);
311
312     CFont* pOldFont = dc.SelectObject(&fontDraw);
313     dc.DrawText(strText, lprcBounds, DT_CENTER | DT_WORDBREAK);
314     dc.SelectObject(pOldFont);
315
316 // Support for Search Handlers
```

```
317 void CTextDocument::InitializeSearchContent()
318 {
319     CString strSearchContent;
320     // Set search contents from document's data.
321     // The content parts should be separated by ";"
322
323     // For example: strSearchContent = _T
324     //                 ("point;rectangle;circle;ole object;");
325     SetSearchContent(strSearchContent);
326
327 void CTextDocument::SetSearchContent(const CString& value)
328 {
329     if (value.IsEmpty())
330     {
331         RemoveChunk(PKEY_Search_Contents.fmtid,
332                     PKEY_Search_Contents.pid);
333     }
334     else
335     {
336         CMFCFilterChunkValueImpl *pChunk = nullptr;
337         ATLTRY(pChunk = new CMFCFilterChunkValueImpl);
338         if (pChunk != nullptr)
339         {
340             pChunk->SetTextValue(PKEY_Search_Contents, value,
341                                   CHUNK_TEXT);
342             SetChunkValue(pChunk);
343         }
344     }
345 #endif // SHARED_HANDLERS
346
347 // Public implementations
348 #ifdef _DEBUG
349 void CTextDocument::AssertValid() const
350 {
351     CDocument::AssertValid();
352 }
353
354 void CTextDocument::Dump(CDumpContext& dc) const
355 {
356     CDocument::Dump(dc);
357 }
358 #endif //_DEBUG
359
360
```

```
1 #pragma once
2
3
4 class CTextHandler
5 {
6 public:
7     // Public constructors
8     CTextHandler();
9     ~CTextHandler();
10
11    // Public commands
12    CString RecieveText();
13    void DestroyText();
14
15    // Public message handlers
16    BOOL OnKeyDown(UINT nChar, UINT nRepCnt, UINT nFlags);
17    BOOL OnKeyUp(UINT nChar, UINT nRepCnt, UINT nFlags);
18
19 private:
20     // Private resources
21     CString text;
22     BOOL shft;
23 };
24
25
```

```
1 #include "pch.h"
2 #include "CTextHandler.h"
3
4 // Public constructors
5 CTextHandler::CTextHandler()
6 {
7     this->shft = FALSE;
8 }
9 CTextHandler::~CTextHandler()
10 {
11 }
12
13 // Public commands
14 CString CTextHandler::RecieveText()
15 {
16     CString temp = this->text;
17     this->text = "";
18     return temp;
19 }
20 void CTextHandler::DestroyText()
21 {
22     this->text = "";
23 }
24
25 // Public message handlers
26 BOOL CTextHandler::OnKeyDown(UINT nChar, UINT nRepCnt, UINT nFlags)
27 {
28     BOOL caps = GetKeyState(VK_CAPITAL);
29
30     switch (nChar) {
31
32         case VK_SPACE:
33             this->text.Append(L" ");
34             break;
35
36         case VK_SHIFT:
37             this->shft = TRUE;
38             break;
39
40         case 0x30:
41             if (this->shft) {
42                 this->text.Append(L"0");
43             }
44             else {
45                 this->text.Append(L"1");
46             }
47             break;
48
49         case 0x31:
50             if (this->shft) {
51                 this->text.Append(L"!");
52             }
53             else {
54                 this->text.Append(L"1");
55             }
56             break;
57 }
```

```
57
58     case 0x32:
59         if (this->shft) {
60             this->text.Append(L"\\");
61         }
62         else {
63             this->text.Append(L"2");
64         }
65         break;
66
67     case 0x33:
68         if (this->shft) {
69             this->text.Append(L"\u00a3");
70         }
71         else {
72             this->text.Append(L"3");
73         }
74         break;
75
76     case 0x34:
77         if (this->shft) {
78             this->text.Append(L"\u00a3");
79         }
80         else {
81             this->text.Append(L"4");
82         }
83         break;
84
85     case 0x35:
86         if (this->shft) {
87             this->text.Append(L"\u00a3");
88         }
89         else {
90             this->text.Append(L"5");
91         }
92         break;
93
94     case 0x36:
95         if (this->shft) {
96             this->text.Append(L"\u00a3");
97         }
98         else {
99             this->text.Append(L"6");
100        }
101        break;
102
103    case 0x37:
104        if (this->shft) {
105            this->text.Append(L"\u00a3");
106        }
107        else {
108            this->text.Append(L"7");
109        }
110        break;
111
112    case 0x38:
```

```
...\source\repos\DesignArk\DesignArk\CTextHandler.cpp
113         if (this->shft) {
114             this->text.Append(L"*");
115         }
116     else {
117         this->text.Append(L"8");
118     }
119     break;
120
121 case 0x39:
122     if (this->shft) {
123         this->text.Append(L"(");
124     }
125     else {
126         this->text.Append(L"9");
127     }
128     break;
129
130 case 0x41:
131     if (!(this->shft && caps) && (this->shft || caps)) {
132         this->text.Append(L"A");
133     }
134     else {
135         this->text.Append(L"a");
136     }
137     break;
138
139 case 0x42:
140     if (!(this->shft && caps) && (this->shft || caps)) {
141         this->text.Append(L"B");
142     }
143     else {
144         this->text.Append(L"b");
145     }
146     break;
147
148 case 0x43:
149     if (!(this->shft && caps) && (this->shft || caps)) {
150         this->text.Append(L"C");
151     }
152     else {
153         this->text.Append(L"c");
154     }
155     break;
156
157 case 0x44:
158     if (!(this->shft && caps) && (this->shft || caps)) {
159         this->text.Append(L"D");
160     }
161     else {
162         this->text.Append(L"d");
163     }
164     break;
165
166 case 0x45:
167     if (!(this->shft && caps) && (this->shft || caps)) {
168         this->text.Append(L"E");
```

```
169         }
170     else {
171         this->text.Append(L"e");
172     }
173     break;
174
175 case 0x46:
176     if (!(this->shft && caps) && (this->shft || caps)) {
177         this->text.Append(L"F");
178     }
179     else {
180         this->text.Append(L"f");
181     }
182     break;
183
184 case 0x47:
185     if (!(this->shft && caps) && (this->shft || caps)) {
186         this->text.Append(L"G");
187     }
188     else {
189         this->text.Append(L"g");
190     }
191     break;
192
193 case 0x48:
194     if (!(this->shft && caps) && (this->shft || caps)) {
195         this->text.Append(L"H");
196     }
197     else {
198         this->text.Append(L"h");
199     }
200     break;
201
202 case 0x49:
203     if (!(this->shft && caps) && (this->shft || caps)) {
204         this->text.Append(L"I");
205     }
206     else {
207         this->text.Append(L"i");
208     }
209     break;
210
211 case 0x4A:
212     if (!(this->shft && caps) && (this->shft || caps)) {
213         this->text.Append(L"J");
214     }
215     else {
216         this->text.Append(L"j");
217     }
218     break;
219
220 case 0x4B:
221     if (!(this->shft && caps) && (this->shft || caps)) {
222         this->text.Append(L"K");
223     }
224     else {
```

```
225         this->text.Append(L"k");
226     }
227     break;
228
229 case 0x4C:
230     if (!(this->shft && caps) && (this->shft || caps)) {
231         this->text.Append(L"L");
232     }
233     else {
234         this->text.Append(L"l");
235     }
236     break;
237
238 case 0x4D:
239     if (!(this->shft && caps) && (this->shft || caps)) {
240         this->text.Append(L"M");
241     }
242     else {
243         this->text.Append(L"m");
244     }
245     break;
246
247 case 0x4E:
248     if (!(this->shft && caps) && (this->shft || caps)) {
249         this->text.Append(L"N");
250     }
251     else {
252         this->text.Append(L"n");
253     }
254     break;
255
256 case 0x4F:
257     if (!(this->shft && caps) && (this->shft || caps)) {
258         this->text.Append(L"O");
259     }
260     else {
261         this->text.Append(L"o");
262     }
263     break;
264
265 case 0x50:
266     if (!(this->shft && caps) && (this->shft || caps)) {
267         this->text.Append(L"P");
268     }
269     else {
270         this->text.Append(L"p");
271     }
272     break;
273
274 case 0x51:
275     if (!(this->shft && caps) && (this->shft || caps)) {
276         this->text.Append(L"Q");
277     }
278     else {
279         this->text.Append(L"q");
280     }
```

```
281         break;
282
283     case 0x52:
284         if (!(this->shft && caps) && (this->shft || caps)) {
285             this->text.Append(L"R");
286         }
287         else {
288             this->text.Append(L"r");
289         }
290         break;
291
292     case 0x53:
293         if (!(this->shft && caps) && (this->shft || caps)) {
294             this->text.Append(L"S");
295         }
296         else {
297             this->text.Append(L"s");
298         }
299         break;
300
301     case 0x54:
302         if (!(this->shft && caps) && (this->shft || caps)) {
303             this->text.Append(L"T");
304         }
305         else {
306             this->text.Append(L"t");
307         }
308         break;
309
310     case 0x55:
311         if (!(this->shft && caps) && (this->shft || caps)) {
312             this->text.Append(L"U");
313         }
314         else {
315             this->text.Append(L"u");
316         }
317         break;
318
319     case 0x56:
320         if (!(this->shft && caps) && (this->shft || caps)) {
321             this->text.Append(L"V");
322         }
323         else {
324             this->text.Append(L"v");
325         }
326         break;
327
328     case 0x57:
329         if (!(this->shft && caps) && (this->shft || caps)) {
330             this->text.Append(L"W");
331         }
332         else {
333             this->text.Append(L"w");
334         }
335         break;
336
```

```
337     case 0x58:
338         if (!(this->shft && caps) && (this->shft || caps)) {
339             this->text.Append(L"X");
340         }
341         else {
342             this->text.Append(L"x");
343         }
344         break;
345
346     case 0x59:
347         if (!(this->shft && caps) && (this->shft || caps)) {
348             this->text.Append(L"Y");
349         }
350         else {
351             this->text.Append(L"y");
352         }
353         break;
354
355     case 0x5A:
356         if (!(this->shft && caps) && (this->shft || caps)) {
357             this->text.Append(L"Z");
358         }
359         else {
360             this->text.Append(L"z");
361         }
362         break;
363
364     case VK_MULTIPLY:
365         this->text.Append(L"*");
366         break;
367
368     case VK_ADD:
369         this->text.Append(L"+");
370         break;
371
372     case VK_SUBTRACT:
373         this->text.Append(L"-");
374         break;
375
376     case VK_DIVIDE:
377         this->text.Append(L"/");
378         break;
379
380     case VK_OEM_1:
381         if (this->shft) {
382             this->text.Append(L":");
383         }
384         else {
385             this->text.Append(L";");
386         }
387         break;
388
389     case VK_OEM_PLUS:
390         if (this->shft) {
391             this->text.Append(L"+");
392         }
```

```
393         else {
394             this->text.Append(L"=");
395         }
396         break;
397
398     case VK_OEM_COMMA:
399         if (this->shft) {
400             this->text.Append(L"<");
401         }
402         else {
403             this->text.Append(L",");
404         }
405         break;
406
407     case VK_OEM_MINUS:
408         if (this->shft) {
409             this->text.Append(L"_");
410         }
411         else {
412             this->text.Append(L"-");
413         }
414         break;
415
416     case VK_OEM_PERIOD:
417         if (this->shft) {
418             this->text.Append(L">");
419         }
420         else {
421             this->text.Append(L".");
422         }
423         break;
424
425     case VK_OEM_2:
426         if (this->shft) {
427             this->text.Append(L"?");
428         }
429         else {
430             this->text.Append(L"/");
431         }
432         break;
433
434     case VK_OEM_3:
435         if (this->shft) {
436             this->text.Append(L"@");
437         }
438         else {
439             this->text.Append(L"'");
440         }
441         break;
442
443     case VK_OEM_4:
444         if (this->shft) {
445             this->text.Append(L"{");
446         }
447         else {
448             this->text.Append(L"[");
```

```
449         }
450         break;
451
452     case VK_OEM_5:
453         if (this->shft) {
454             this->text.Append(L" | ");
455         }
456         else {
457             this->text.Append(L"\\\" );
458         }
459         break;
460
461     case VK_OEM_6:
462         if (this->shft) {
463             this->text.Append(L"} ");
464         }
465         else {
466             this->text.Append(L"] ");
467         }
468         break;
469
470     default:
471         return FALSE;
472
473     }
474     return TRUE;
475 }
476 BOOL CTextHandler::OnKeyUp(UINT nChar, UINT nRepCnt, UINT nFlags)
477 {
478     if (nChar == VK_SHIFT) {
479         this->shft = FALSE;
480     }
481     return 0;
482 }
```

```
1
2 #pragma once
3 #include "DesignArk.h"
4
5 class CAppObject
6 {
7 public:
8     // Public constructors
9     CAppObject(CRect bounds, CString ID, BOOL active = FALSE);
10    ~CAppObject();
11
12    // Public commands
13    virtual void draw(CDC* pDC);
14
15    // Public message handlers
16    virtual void OnSize(UINT nType, int cx, int cy);
17
18    virtual BOOL OnLButtonUp(UINT nFlags, CPoint point);
19    virtual BOOL OnLButtonDown(UINT nFlags, CPoint point);
20    virtual void OnLButtonDblClk(UINT nFlags, CPoint point);
21    virtual void OnRButtonUp(UINT nFlags, CPoint point);
22    virtual void OnRButtonDown(UINT nFlags, CPoint point);
23    virtual BOOL OnMouseMove(UINT nFlags, CPoint point);
24
25    virtual int OnRecieveText(CString text);
26    virtual void OnRecieveBackspace();
27    virtual void OnRecieveReturn();
28    virtual void OnRecieveTab();
29
30    virtual void OnKeyDown(UINT nChar, UINT nRepCnt, UINT nFlags);
31
32    virtual BOOL OnContextMenu(CWnd* pWnd, CPoint point);
33
34    // Public getters, setters and checkers
35    CString getID();
36    void setID(CString ID);
37
38    CRect getBounds();
39    BOOL checkPointInBounds(CPoint point);
40
41    virtual void setBounds(CRect bounds);
42    virtual void setBounds(int x, int y, int cx, int cy);
43    virtual void setPosition(CPoint point);
44    virtual void move(int x, int y);
45
46    BOOL isActive();
47    virtual void setActive(BOOL val);
48
49 protected:
50    CString ID;
51    CRect bounds;
52    BOOL active;
53};
```

```
1 #include "pch.h"
2 #include "CAppObject.h"
3
4 // Public constructors
5 CAppObject::CAppObject(CRect bounds, CString ID, BOOL active)
6 {
7     this->bounds = bounds;
8     this->ID = ID;
9     this->active = active;
10 }
11 CAppObject::~CAppObject()
12 {
13 }
14
15 void CAppObject::draw(CDC* pDC)
16 {
17 }
18
19 // Public message handlers
20 void CAppObject::OnSize(UINT nType, int cx, int cy)
21 {
22 }
23
24 BOOL CAppObject::OnLButtonUp(UINT nFlags, CPoint point)
25 {
26     return 0;
27 }
28 BOOL CAppObject::OnLButtonDown(UINT nFlags, CPoint point)
29 {
30     return 0;
31 }
32 void CAppObject::OnLButtonDblClk(UINT nFlags, CPoint point)
33 {
34 }
35 void CAppObject::OnRButtonUp(UINT nFlags, CPoint point)
36 {
37 }
38 void CAppObject::OnRButtonDown(UINT nFlags, CPoint point)
39 {
40 }
41 BOOL CAppObject::OnMouseMove(UINT nFlags, CPoint point)
42 {
43     return TRUE;
44 }
45 int CAppObject::OnRecieveText(CString text)
46 {
47     return 0;
48 }
49 void CAppObject::OnRecieveBackspace()
50 {
51 }
52 void CAppObject::OnRecieveReturn()
53 {
54 }
55 void CAppObject::OnRecieveTab()
56 {
```

```
57 }
58
59 void CAppObject::OnKeyDown(UINT nChar, UINT nRepCnt, UINT nFlags)
60 {
61 }
62
63 BOOL CAppObject::OnContextMenu(CWnd* pWnd, CPoint point)
64 {
65     return 0;
66 }
67
68 // Public getters, setters and checkers
69 CString CAppObject::getID()
70 {
71     return this->ID;
72 }
73 void CAppObject::setID(CString ID)
74 {
75     this->ID = ID;
76 }
77
78 CRect CAppObject::getBounds()
79 {
80     return this->bounds;
81 }
82 BOOL CAppObject::checkPointInBounds(CPoint point)
83 {
84     if (this->bounds.left <= point.x && bounds.right >= point.x &&
85         this->bounds.top <= point.y && this->bounds.bottom >=
86             point.y) {
87         return TRUE;
88     }
89     return FALSE;
90 }
91
92 void CAppObject::setBounds(CRect bounds)
93 {
94     this->bounds = bounds;
95 }
96 void CAppObject::setBounds(int x, int y, int cx, int cy)
97 {
98     this->bounds.left = x;
99     this->bounds.top = y;
100    this->bounds.right = x + cx;
101    this->bounds.bottom = y + cy;
102 }
103 void CAppObject::setPosition(CPoint point)
104 {
105     this->move(point.x - this->bounds.left, point.y - this-
106                 >bounds.top);
107 }
108 void CAppObject::move(int x, int y)
109 {
110     this->bounds.left += x;
111     this->bounds.right += x;
```

```
111     this->bounds.top += y;
112     this->bounds.bottom += y;
113 }
114
115 BOOL CAppObject::isActive()
116 {
117     return this->active;
118 }
119 void CAppObject::setActive(BOOL val)
120 {
121     this->active = val;
122 }
123
```

```
1 #pragma once
2 #include "CTextLineObject.h"
3
4
5 class CTextEditorObject :
6     public CAppObject
7 {
8 public:
9     // Public constructors
10    CTextEditorObject(CRect bounds, CString ID, BOOL lineNums, int maxLines = 0, int defBoxHeight = 0, BOOL active = FALSE, std::vector<CString> text = {L""}, std::vector<int> line = {1}, int lineOffset = 0);
11    ~CTextEditorObject();
12
13    // Public implementations
14    int draw(CDC* pDc, CSize textExtent, int xScrollPosition, int returnNewLines, BOOL printing, CRect printAreaLength = CRect());
15
16    // Public getters, setters and checkers
17    int getCaretPos();
18    CPoint getCaretPoint(CSize caretSize);
19    CSize getTextExtent();
20    void setTextExtent(CSize size);
21
22    int getRecentPos();
23
24    BOOL pointHighlighted(CPoint point);
25    BOOL hasHighlight();
26    void hlightingOff();
27    CRect getHighlightClippingRect();
28    CRgn* getHighlightExactRgn(int x_offset, int y_offset);
29    int getStartLine();
30    BOOL isHlghtMultiline();
31    int lineHighlight(int line);
32    // RETURNS 0 for none
33    //          1 for partially highlighted
34    //          2 for line highlighted
35
36    std::vector<int> iGetLineNum(int a = 0);
37    CString sGetLineNum(int a = 0);
38
39    void incrementSublines(int subline, int val);
40    int getActiveLine();
41    int getBlockLine();
42    int getPrintLine();
43    CString getLineText(int line = 0);
44    CRect getLineBounds(int line = 0);
45    CString getHighlightedText();
46    int getLineTextWidth();
47    int getNumLines();
48    int getBoxHeight(BOOL default = FALSE);
49
50    virtual void setActive(BOOL active);
51    virtual void move(int x, int y);
```

```
52     virtual void setBounds(CRect bounds);
53     std::tuple<CRect, int> getPrintBounds(int returnNewLines, int ↵
54         printAreaLength);
55     int getCursorArrow();
56
57     void initialise();
58
59     // Public message handlers
60     virtual void OnSize(UINT nType, int x, int y);
61
62     virtual BOOL OnLButtonUp(UINT nFlags, CPoint point);
63     virtual BOOL OnLButtonDown(UINT nFlags, CPoint point);
64     virtual void OnLButtonDblClk(UINT nFlags, CPoint point);
65     virtual void OnRButtonUp(UINT nFlags, CPoint point);
66     virtual void OnRButtonDown(UINT nFlags, CPoint point);
67     virtual BOOL OnMouseMove(UINT nFlags, CPoint point);
68
69     virtual int OnRecieveText(CString text, BOOL open = FALSE);
70     virtual void OnRecieveBackspace();
71     virtual void OnRecieveTab();
72     virtual void OnRecieveReturn(BOOL open = FALSE);
73
74     virtual void OnKeyDown(UINT nChar, UINT nRepCnt, UINT nFlags);
75     virtual BOOL OnContextMenu(CWnd* pWnd, CPoint point);
76
77 private:
78
79     // Private Implementations
80     void setSidebar();
81     void setHighlighter();
82     void setBracketsNull();
83
84     // Private resources
85
86     // Positioning
87     int activeLine;
88     int mouseLine;
89     int blockLine;
90     int printLine;
91     int caretPos;
92
93     // Sizes
94     int defBoxHeight;
95     CSize textExtent;
96
97     // Highlights
98     BOOL hlight;           // Used to tell the editor that there is ↵
99         a highlight
100    BOOL hlighting;        // Used to tell that the user is      ↵
101        currently highlighting
102    CPoint hlightStartP;   // Stores the start of the highlight
103    CPoint hlightEndP;    // Stores the end/current mouse pos for ↵
104        the highlighting
105    int startPos;          // Stores the original starting pos in ↵
106        terms of character spaces for the highlight
```

```
...urce\repos\DesignArk\DesignArk\CTextEditorObject.h 3
103     int startLine;           // Stores the original starting line for the highlight
104
105     // Recent
106     BOOL recentHlght;
107     int recentPos;
108     int recentLine;
109
110     // Line stuff
111     std::vector<CTextLineObject*> lines;
112     int maxLines;
113     int lineOffset;
114     BOOL lineNums;
115     BOOL hoverSubLine;
116     BOOL clickSubLine;
117     CRect sidebar;
118
119     // Editing
120     std::vector<std::vector<int>> brackets;
121
122     /*
123
124     Brackets contain the positions and types of all the auto completed brackets currently in the editor.
125     Auto-brackets must always be on the same line and it will only be auto for as long the user is editing that line.
126
127     Each record in brackets contain three integers.
128
129     [x][0] contains the position of the first bracket.
130     [x][1] contains the position of the second bracket.
131     [x][2] contains the type of bracket that has been saved.
132
133     if [x][2] == 0 : type = (
134         1 : type = {
135             2 : type = [
136                 3 : type = "
137                 4 : type = '
138
139     */
140
141
142     BOOL bracketContains(int value, int& itPos, int type, int side = 1);
143     void moveBrackets(int val, int index);
144
145     // Stuff for view
146     int cursor_arrow;
147     BOOL lMouseDown;
148 };
149
150
```

```
1 // CTextEditorObject.cpp : implementation of the CTextEditorObject class
2 // 
3 // 
4 #include "pch.h"
5 #include "framework.h"
6 // SHARED_HANDLERS can be defined in an ATL project implementing preview, thumbnail
7 // and search filter handlers and allows sharing of document code with that project.
8 #ifndef SHARED_HANDLERS
9 #include "DesignArk.h"
10 #endif
11 
12 #include "CTextEditorObject.h"
13 
14 // Public contructors
15 CTextEditorObject::CTextEditorObject(CRect bounds, CString ID,
16                                     BOOL lineNums, int maxLines, int defBoxHeight,
17                                     BOOL active,
18                                     std::vector<CString> text, std::vector<int> line, int lineOffset)
19 : CAppObject(bounds, ID, active)
20 {
21     this->activeLine = 1;
22     this->mouseLine = 0;
23     this->blockLine = 0;
24     this->printLine = 0;
25     this->caretPos = text[text.size() - 1].GetLength();
26     this->maxLines = maxLines;
27 
28     this->defBoxHeight = defBoxHeight;
29 
30     this->hlght = FALSE;
31     this->hlghting = FALSE;
32     this->hlghtStartP.x = 0; this->hlghtStartP.y = 0;
33     this->hlghtEndP.x = 0; this->hlghtEndP.y = 0;
34 
35     this->lineNums = lineNums;
36     this->lineOffset = lineOffset;
37     this->hoverSubLine = FALSE;
38     this->clickSubLine = FALSE;
39 
40     int i = 1;
41     CString id;
42     CRect lineBounds = bounds;
43     std::vector<int> lineNumbers = line;
44 
45     if (this->lineNums) {
46 
47         lineBounds.left += (this->lineOffset * theApp.zoom);
48     }
49 
50     lineBounds.bottom = lineBounds.top + static_cast<int>(this-
```

```
...ce\repos\DesignArk\DesignArk\CTextEditorObject.cpp 2
    >defBoxHeight * theApp.zoom);
52
53     for (auto& text : text) {
54
55         id.Format(L"%d", i-1);
56
57         this->lines.push_back(new CTextLineObject(lineBounds, id, ↵
58             lineNumbers, text, FALSE));
59         lineBounds.top += static_cast<int>(this->defBoxHeight * ↵
60             theApp.zoom);
61         lineBounds.bottom += static_cast<int>(this->defBoxHeight * ↵
62             theApp.zoom);
63
64         lineNumbers[lineNumbers.size()-1]++;
65         i++;
66     }
67
68     this->lines[0]->setActive(TRUE);
69
70     this->cursor_arrow = 0;
71     this->lMouseDown = FALSE;
72
73 }
74 CTextEditorObject::~CTextEditorObject()
75 {
76     for (auto& it : this->lines) {
77         delete it;
78     }
79 }
80
81 // Public implementations
82 int CTextEditorObject::draw(CDC* pDc, CSize textExtent, int ↵
83     xScrollPosition, int returnNewLines, BOOL printing, CRect ↵
84     printAreaLength)
85 {
86     this->textExtent = pDc->GetTextExtent(L"A", 1);
87
88     if (printing) {
89         this->setBracketsNull();
90     }
91
92     if (!printing) {
93         for (auto& it : this->lines) {
94             returnNewLines = it->draw(pDc, this->textExtent, ↵
95                 this->brackets, returnNewLines, printing);
96         }
97
98         // Prints the sidebar
99         if (!printing) {
100             this->sidebar.left = xScrollPosition;
```

```

...ce\repos\DesignArk\DesignArk\CTextEditorObject.cpp 3
101         this->sidebar.right = xScrollPosition +
102             static_cast<int>(this->defBoxHeight * theApp.zoom); ↵
103     }
104     else {
105         this->sidebar.left = 0;
106         this->sidebar.right = 0;
107     }
108     if (!printing) {
109
110         CPen penWhite;
111         penWhite.CreatePen(PS_SOLID, 0, RGB(255, 255, 255));
112         CBrush brushWhite;
113         brushWhite.CreateSolidBrush(RGB(255, 255, 255));
114
115         CPen* pOldPen = pDc->SelectObject(&penWhite);
116         CBrush* oldBrush = pDc->SelectObject(&brushWhite);
117
118         pDc->Rectangle(CRect(this->sidebar.left, this- ↵
119             >sidebar.top + (returnNewLines * this->defBoxHeight * ↵
120                 theApp.zoom), this->sidebar.left + (this->lineOffset ↵
121                 * theApp.zoom), this->sidebar.bottom + 2)); ↵
122
123         pDc->SelectObject(pOldPen);
124         pDc->SelectObject(oldBrush);
125
126         pDc->RoundRect(this->sidebar, CPoint(8 * theApp.zoom, 8 ↵
127             * theApp.zoom));
128
129         pDc->SetTextColor(RGB(50, 50, 150));
130
131         if (this->blockLine != 0) {
132
133             Gdiplus::Graphics g(pDc->GetSafeHdc());
134
135             if (this->lMouseDown) {
136                 Gdiplus::Bitmap bitmap(L"res/ ↵
137                     arrow_right_click.bmp"); ↵
138
139                 Gdiplus::Rect expansionRect(this->sidebar.left ↵
140                     + static_cast<int>(this->defBoxHeight * ↵
141                         theApp.zoom) / 2 - (bitmap.GetWidth() * ↵
142                             theApp.zoom) / 2, this->lines[this->blockLine - ↵
143                                 1]->getBounds().top + static_cast<int>(this- ↵
144                                     >defBoxHeight * theApp.zoom) / 2 - ↵
145                                         (bitmap.GetHeight() * theApp.zoom) / 2, ↵
146                                         (bitmap.GetWidth() * theApp.zoom), ↵
147                                         (bitmap.GetHeight() * theApp.zoom));
148
149                 g.DrawImage(&bitmap, expansionRect);
150             }
151             else {
152                 Gdiplus::Bitmap bitmap(L"res/arrow_right.bmp"); ↵
153
154                 Gdiplus::Rect expansionRect(this->sidebar.left ↵

```

```

...ce\repos\DesignArk\DesignArk\CTextEditorObject.cpp 4
    + static_cast<int>(this->defBoxHeight *
theApp.zoom) / 2 - (bitmap.GetWidth() *
theApp.zoom) / 2, this->lines[this->blockLine - 1]-
>getBounds().top + static_cast<int>(this-
>defBoxHeight * theApp.zoom) / 2 -
(bitmap.GetHeight() * theApp.zoom) / 2,
143
    (bitmap.GetWidth() * theApp.zoom), (bitmap.GetHeight() * *
theApp.zoom));
144
145             g.DrawImage(&bitmap, expansionRect);
146         }
147     }
148 }
149
150     if (printing) {
151
152         int i = this->printLine;
153         BOOL finished = FALSE;
154
155         while (!finished && i < this->lines.size()) {
156
157             int lines = std::ceil(float(this->lines[i]->getText
()
.GetLength()) / float((printAreaLength.Width() -
this->lines[i]->getBounds().left * 2) / this-
>textExtent.cx));
158             if (lines == 0) {
159                 lines++;
160             }
161
162             if (this->lines[i]->getBounds().top + (lines +
returnNewLines) * this->defBoxHeight() <
printAreaLength.bottom) {
163
164                 pDc->SetTextColor(RGB(50, 50, 150));
165                 pDc->TextOut(this->sidebar.left + 400 - (this-
>lines[i]->sGetLineNums()).GetLength() * this-
>textExtent.cx, this->lines[i]->getBounds().top +
this->lines[i]->getBounds().Height() / 2 - this-
>textExtent.cy / 2 + (returnNewLines * this-
>defBoxHeight * theApp.zoom), this->lines[i]-
>sGetLineNums(), this->lines[i]->sGetLineNums
()
.GetLength());
166
167                 pDc->SetTextColor(RGB(0, 0, 0));
168                 returnNewLines = this->lines[i]->draw(pDc,
this->textExtent, this->brackets, returnNewLines,
printing, printAreaLength.Width());
169
170                 i++;
171             }
172         else {
173             finished = TRUE;
174         }
175
176     }

```

```
177
178         this->printLine = i;
179
180         if (this->printLine == this->lines.size()) {
181             this->printLine = 0;
182         }
183     }
184     else {
185         for (auto& it : this->lines) {
186
187             pDc->TextOut(this->sidebar.left + (this->lineOffset ↗
188                 * theApp.zoom) - it->sGetLineNums().GetLength() * ↗
189                 this->textExtent.cx - 2, it->getBounds().top + ↗
190                 it->getBounds().Height() / 2 - this- ↗
191                 >textExtent.cy / 2, it->sGetLineNums(), it- ↗
192                 >sGetLineNums().GetLength());
193
194         }
195     }
196 }
197
198     return returnNewLines;
199 }
200
201 // Public getters and checkers
202 int CTextEditorObject::getCaretPos()
203 {
204     return this->caretPos;
205 }
206
207 BOOL CTextEditorObject::pointHighlighted(CPoint point)
208 {
209     // TODO : Check if point is highlighted
210     return FALSE;
211 }
212 BOOL CTextEditorObject::hasHighlight()
213 {
214     return this->hlght;
215 }
216 void CTextEditorObject::hlghtingOff()
217 {
218     this->hlghting = FALSE;
219     this->hlght = FALSE;
220     this->startLine = 0;
221     this->startPos = 0;
222
223     this->recentHlight = FALSE;
224 }
```

```
225     if (this->active) {
226         this->lines[this->activeLine - 1]->setActive(TRUE);
227     }
228
229     for (auto& it : this->lines) {
230         it->setHighlighting(FALSE);
231     }
232 }
233
234 CRect CTextEditorObject::getHighlightClippingRect()
235 {
236     CRect rgn;
237
238     if (this->hlghting) {
239
240         if (this->recentLine == this->activeLine) {
241
242             if (!this->recentHlight && this->hlghting) {
243                 rgn = this->lines[this->activeLine - 1]->getBounds ↩
244                     ();
245             }
246             else if (this->recentPos != this->caretPos) {
247
248                 rgn = this->lines[this->activeLine - 1]->getBounds ↩
249                     ();
250                 rgn.right = rgn.left + max(this->recentPos, this- ↩
251                     >caretPos) * this->textExtent.cx + (2 * ↩
252                     theApp.zoom);
253                 rgn.left = rgn.left + min(this->recentPos, this- ↩
254                     >caretPos) * this->textExtent.cx - (2 * ↩
255                     theApp.zoom);
256             }
257         }
258         return rgn;
259     }
260
261 CRgn* CTextEditorObject::getHighlightExactRgn(int x_offset, int y_offset)
262 {
263     CRgn* rgn = new CRgn();
264     VERIFY(rgn->CreateRectRgn(0, 0, 0, 0));
265
266     for (auto& it : this->lines) {
267
268         if(it->getHStart() != it->getHEnd()) {
269
270             CRgn* rgn1 = new CRgn();
271             int x1 = (it->getHStart() * this->textExtent.cx) + it- ↩
```

```
272             >getBounds().left - (2 * theApp.zoom) - x_offset,
273             y1 = it->getBounds().top - y_offset,
274             x2 = (it->getHEnd() * this->textExtent.cx) + it-
275                 >getBounds().left + (2 * theApp.zoom) - x_offset,
276             y2 = it->getBounds().bottom - y_offset;
277
278             ASSERT(rgn1->CreateRectRgn(x1, y1, x2, y2));
279
280             int nCombineResult = rgn->CombineRgn(rgn, rgn1,
281                                         RGN_OR);
282             ASSERT(nCombineResult != ERROR && nCombineResult !=
283                   NULLREGION);
284         }
285     }
286     return rgn;
287 }
288
289 int CTextEditorObject::getStartLine()
290 {
291     return this->startLine - 1;
292 }
293
294 BOOL CTextEditorObject::isHlghtMultiline()
295 {
296     BOOL first = FALSE;
297
298     for (auto& it : this->lines) {
299         if (it->isHighlighting()) {
300             if (!first) {
301                 first = TRUE;
302             }
303         }
304     }
305     return FALSE;
306 }
307
308 int CTextEditorObject::lineHighlight(int line)
309 {
310     if (this->lines[line]->isHighlighting()) {
311         if (this->lines[line]->isLineHighlighted()) {
312             return 2;
313         }
314     }
315     return 0;
316 }
317
318 CPoint CTextEditorObject::getCaretPoint(CSize caretSize)
319 {
320     CPoint caretPoint;
321     BOOL els = TRUE;
322
323     int offset = 0;
```

```
324
325     if (this->lines[this->activeLine - 1]->isHighlighting() &&
326         (this->lines[this->activeLine - 1]->getLength() == 0)) {
327         offset = 1;
328     }
329     if (this->caretPos >= this->lines[this->activeLine - 1]-
330         >getLength() + offset) {
331         caretPoint.x = (this->lines[this->activeLine - 1]-
332             >getLength() + offset) * (this->textExtent.cx) + (this-
333             >lineOffset * theApp.zoom) + 1;
334     }
335     else {
336         caretPoint.x = (this->caretPos) * this->textExtent.cx +
337             (this->lineOffset * theApp.zoom) + 1;
338     }
339 }
340
341 CSize CTextEditorObject::getTextExtent()
342 {
343     return this->textExtent;
344 }
345
346 void CTextEditorObject::setTextExtent(CSize size)
347 {
348     this->textExtent = size;
349 }
350
351 int CTextEditorObject::getRecentPos()
352 {
353     return this->recentPos;
354 }
355
356 void CTextEditorObject::incrementSublines(int subline, int val)
357 {
358     for (auto& it : this->lines) {
359         it->incrementLine(subline, val);
360     }
361 }
362
363 void CTextEditorObject:: setActive(BOOL active)
364 {
365     this->active = active;
366
367     if (!this->active) {
368
369         this->hoverSubLine = FALSE;
370         this->clickSubLine = FALSE;
371         this->blockLine = 0;
372 }
```

```
373         this->highlightOff();
374
375         this->lines[this->activeLine - 1]->setActive(FALSE);
376     }
377     else {
378         if (this->blockLine != 0) {
379             this->activeLine = this->blockLine;
380         }
381         this->lines[this->activeLine - 1]->setActive(TRUE);
382         this->caretPos = 0;
383     }
384 }
385
386 void CTextEditorObject::move(int x, int y)
387 {
388     CAppObject::move(x, y);
389     for (auto& it : this->lines) {
390         it->move(x, y);
391     }
392     this->sidebar += CPoint(x, y);
393 }
394 void CTextEditorObject::setBounds(CRect bounds)
395 {
396     this->bounds = bounds;
397     CRect newBounds = this->bounds;
398     newBounds.left = newBounds.left + static_cast<int>(this->lineOffset * theApp.zoom); ↗
399     newBounds.bottom = newBounds.top + static_cast<int>(this->defBoxHeight * theApp.zoom); ↗
400
401     for (auto& it : this->lines) {
402
403         it->setBounds(newBounds);
404         newBounds.top += static_cast<int>(this->defBoxHeight * ↗
405             theApp.zoom);
406         newBounds.bottom += static_cast<int>(this->defBoxHeight * ↗
407             theApp.zoom);
408     }
409 }
410 std::tuple<CRect, int> CTextEditorObject::getPrintBounds(int ↗
411     returnNewLines, int printAreaLength)
412 {
413     CRect printBounds = this->lines[0]->getBounds();
414     int thisBoundsNewLines = returnNewLines;
415
416     int printMaxCharLength = (printAreaLength - this->lines[0]->getBounds().left * 2) / this->textExtent.cx; ↗
417
418     for (int i = 0; i < this->lines.size(); i++) {
419         thisBoundsNewLines += std::ceil(double(this->lines[i]->getLength() / printMaxCharLength));
420
421         if (i == this->printLine) {
422             printBounds.top += this->defBoxHeight * theApp.zoom * ↗
423                 thisBoundsNewLines;
```

```
421         }
422     }
423
424     printBounds.right = printBounds.left + printMaxCharLength * ↵
425         this->textExtent.cx; ↵
426     printBounds.bottom = printBounds.top + this->defBoxHeight * ↵
427         theApp.zoom * (thisBoundsNewLines - returnNewLines + this- ↵
428             >lines.size()); ↵
429
430     return std::tuple<CRect, int>(printBounds, thisBoundsNewLines);
431 }
432
433 int CTextEditorObject::getCursorArrow()
434 {
435     return this->cursor_arrow;
436 }
437 void CTextEditorObject::initialise()
438 {
439     this->caretPos = 0;
440     this->lines[this->activeLine - 1]->setActive(FALSE);
441     this->activeLine = 1;
442     this->lines[this->activeLine - 1]->setActive(TRUE);
443 }
444
445 // Public message handlers
446 void CTextEditorObject::OnSize(UINT nType, int cx, int cy)
447 {
448     this->bounds.bottom = this->bounds.top + static_cast<int> ↵
449         (defBoxHeight * theApp.zoom) * (this->lines.size());
450
451     CRect lineBounds;
452     lineBounds.left = static_cast<int>(this->lineOffset * ↵
453         theApp.zoom);
454     lineBounds.right = this->bounds.right;
455     lineBounds.top = this->bounds.top;
456     lineBounds.bottom = lineBounds.top + static_cast<int>(this- ↵
457         >defBoxHeight * theApp.zoom);
458
459     for (auto& it : this->lines) {
460
461         it->setBounds(lineBounds);
462
463         lineBounds.top += static_cast<int>(this->defBoxHeight * ↵
464             theApp.zoom);
465         lineBounds.bottom += static_cast<int>(this->defBoxHeight * ↵
466             theApp.zoom);
467     }
468
469     this->setSidebar();
470 }
471
472 BOOL CTextEditorObject::OnLButtonUp(UINT nFlags, CPoint point)
473 {
474     BOOL redraw = FALSE;
475
476     this->lMouseDown = FALSE;
```

```
469
470     if (this->hlghting) {
471         this->hlghting = FALSE;
472
473         if ((this->startLine == this->activeLine && this->startPos == this->caretPos) ||
474
475             (this->startLine == this->activeLine &&
476             this->startPos >= this->lines[this->activeLine - 1]->getLength() &&
477             this->caretPos >= this->lines[this->activeLine - 1]->getLength())) {
478
479
480         this->hlghtingOff();
481         redraw = TRUE;
482     }
483     else {
484         this->hlght = TRUE;
485     }
486 }
487
488     if (point.x <= this->sidebar.right) {
489         BOOL update = FALSE;
490         int i = 1;
491
492         while (!update && i <= this->lines.size()) {
493
494             if (point.y >= this->lines[i - 1]->getBounds().top &&
495                 point.y < this->lines[i - 1]->getBounds().bottom) {
496
497                 this->clickSubLine = TRUE;
498
499                 this->caretPos = 0;
500                 update = TRUE;
501
502             }
503             i++;
504         }
505
506     }
507     return redraw;
508 }
509 BOOL CTextEditorObject::OnLButtonDown(UINT nFlags, CPoint point)
510 {
511     this->hlghtingOff();
512     BOOL sidebarbttnhold = FALSE;
513     this->lMouseDown = TRUE;
514
515     BOOL update = FALSE;
516     int i = 1;
517
518     // Iterate through all the lines
519     while (!update && i <= this->lines.size()) {
520
521         // The point is in the lines
```

...ce\repos\DesignArk\DesignArk\CTextEditorObject.cpp 12

---

```

522         if (point.y >= this->lines[i - 1]->getBounds().top && ↵
523             point.y < this->lines[i - 1]->getBounds().bottom) {
524             // If we have changed lines
525             if (i != this->activeLine) {
526                 this->lines[this->activeLine - 1]->setActive ↵
527                     (FALSE);
528                 this->activeLine = i;
529                 this->lines[i - 1]->setActive(TRUE);
530             }
531             // If we have selected part of the text
532             if (this->lines[i - 1]->checkPointInBounds(point)) {
533                 if (point.x - static_cast<int>(this->defBoxHeight * theApp.zoom) - (this->lineOffset * theApp.zoom) ↵
534                     >= this->lines[i - 1]->getLength() * this->textExtent.cx) {
535                     this->caretPos = this->lines[i - 1]->getLength ↵
536                         ();
537                 }
538                 else {
539                     this->caretPos = (point.x - (this->lineOffset * theApp.zoom) / this->textExtent.cx);
540                 }
541             }
542             // If we have selected the side of the text editor
543             else {
544                 // If we selected a button
545                 if (point.x < static_cast<int>(this->defBoxHeight * theApp.zoom)) {
546                     this->blockLine = i;
547                     sidebarbttnhold = TRUE;
548                 }
549                 // If we select the bit that isn't the sidebar
550                 else {
551                     this->hlghtingOff();
552                     this->hlght = TRUE;
553                     this->lines[this->activeLine - 1]->highlightLine();
554                 }
555                 this->caretPos = 0;
556             }
557             update = TRUE;
558         }
559     }
560     i++;
561 }
562 if (!update && point.x >= static_cast<int>(this->defBoxHeight * theApp.zoom) + (this->lineOffset * theApp.zoom)) {

```

```
568         this->lines[this->activeLine - 1]->setActive(FALSE);
569         this->activeLine = this->lines.size();
570         this->lines[this->activeLine - 1]->setActive(TRUE);
571         this->caretPos = this->lines[this->activeLine - 1]-
572             >getLength();
573     }
574 
575     this->hlghtStartP = point;
576     this->setBracketsNull();
577 
578     return sidebarbttnhold;
579 }
579 void CTextEditorObject::OnLButtonDblClk(UINT nFlags, CPoint point)
580 {
581     int leftpos = this->caretPos - 1, rightpos = this->caretPos;
582     BOOL leftstop = FALSE, rightstop = FALSE;
583 
584     CString lineText = this->lines[this->activeLine - 1]->getText
585         ();
586 
586     while (!(leftstop && rightstop) && !(leftpos == 0 && rightpos ==
587         lineText.GetLength())) {
588 
589         if (leftpos == 0) {
590             leftstop = TRUE;
591         }
592         else if(lineText.Mid(leftpos, 1) == L" " && !leftstop) {
593             leftpos++;
594             leftstop = TRUE;
595         }
596         else if(!leftstop) {
597             leftpos--;
598         }
599 
600         if ((lineText.Mid(rightpos, 1) == L" " && !rightstop) ||
601             rightpos == lineText.GetLength()) {
602             rightstop = TRUE;
603         }
604         else if(!rightstop) {
605             rightpos++;
606         }
607     }
608 
607     this->lines[this->activeLine - 1]->setHighlighter(leftpos,
609         rightpos);
610     this->hlght = TRUE;
611 
610     this->caretPos = rightpos;
611 }
612 void CTextEditorObject::OnRButtonUp(UINT nFlags, CPoint point)
613 {
614 }
615 void CTextEditorObject::OnRButtonDown(UINT nFlags, CPoint point)
616 {
617 
618     if (!this->hlght) {
```

```
619
620     BOOL update = FALSE;
621     int i = 1;
622
623     while (!update && i <= this->lines.size()) {
624
625         if (this->lines[i - 1]->checkPointInBounds(point)) {
626
627             this->lines[this->activeLine - 1]->setActive
628             (FALSE);
629             this->activeLine = i;
630             this->lines[i - 1]->setActive(TRUE);
631
632             if (point.x - static_cast<int>(this->defBoxHeight *
633                 theApp.zoom) - (this->lineOffset * theApp.zoom)
634                 >= this->lines[i - 1]->getLength() * this-
635                 >textExtent.cx) {
636                 this->caretPos = this->lines[i - 1]->getLength
637                 ();
638             }
639             else {
640                 this->caretPos = (point.x - (this->lineOffset *
641                     theApp.zoom)) / this->textExtent.cx;
642             }
643             update = TRUE;
644
645             i++;
646         }
647
648     }
649
650     //if (!update && point.x >= static_cast<int>(this->defBoxHeight *
651     //    * theApp.zoom)) {
652     //    this->lines[this->activeLine - 1]->setActive(FALSE);
653     //    this->activeLine = this->lines.size();
654     //    this->lines[this->activeLine - 1]->setActive(TRUE);
655     //    this->caretPos = this->lines[this->activeLine - 1]-
656     //        >getLength();
657     //}
658
659     this->setBracketsNull();
660 }
661
662 BOOL CTextEditorObject::OnMouseMove(UINT nFlags, CPoint point)
663 {
664     BOOL mUpdate = FALSE, bUpdate = FALSE;
665     BOOL brk = FALSE;
666
667     int i = 1;
668     BOOL redraw = FALSE;
669
670     while (!mUpdate && !brk && i <= this->lines.size()) {
671
672         if (point.y >= this->lines[i - 1]->getBounds().top &&
673             point.y < this->lines[i - 1]->getBounds().bottom) {
```

```
666         this->mouseLine = i;
667         mUpdate = TRUE;
668
669         if (this->lineNums) {
670
671             if (!this->lines[i - 1]->checkPointInBounds(point)) {
672
673                 if (point.x <= this->sidebar.right) {
674
675                     if (this->blockLine != i) {
676
677                         redraw = TRUE;
678                         this->blockLine = i;
679                     }
680                     bUpdate = TRUE;
681                     this->cursor_arrow = 2;
682                 }
683                 else {
684                     this->cursor_arrow = 1;
685                 }
686             }
687             else {
688                 this->cursor_arrow = 0;
689             }
690         }
691
692         brk = TRUE;
693     }
694     i++;
695 }
696 if (!mUpdate) {
697     this->mouseLine = 0;
698     this->cursor_arrow = 1;
699 }
700 if (!bUpdate) {
701     if (this->blockLine != 0) {
702         redraw = TRUE;
703     }
704     this->blockLine = 0;
705 }
706
707 if (nFlags == MK_LBUTTON) {
708
709     if (this->checkPointInBounds(point)) {
710         this->hlghtEndP = point;
711         this->setHighlighter();
712     }
713     else if (point.y < this->bounds.top) {
714         this->hlghtEndP = CPoint(this->lines[0]->getBounds()
715             (.left + 1, this->lines[0]->getBounds().top +
716             static_cast<int>(this->defBoxHeight *
717             theApp.zoom) / 2);
718     }
719     else {
720         this->hlghtEndP = CPoint(this->lines[this->lines.size() - 1]->getBounds()
721             (.left + 1, this->lines[this->lines.size() - 1]->getBounds().top +
722             static_cast<int>(this->defBoxHeight *
723             theApp.zoom) / 2);
724     }
725 }
```

```

...ce\repos\DesignArk\DesignArk\CTextEditorObject.cpp 16
    - 1]->getLength()*this->textExtent.cx + 1, this-
    >lines[this->lines.size() - 1]->getBounds().top +
    static_cast<int>(this->defBoxHeight * theApp.zoom) /
    2);
718     }
719     redraw = TRUE;
720 }
721 else {
722     this->hlghting = FALSE;
723 }
724
725 return redraw;
726 }
727
728 BOOL CTextEditorObject::OnContextMenu(CWnd* pWnd, CPoint point)
729 {
730     // TODO : Allow restricted choice in context menu when
    selecting point that isn't highlighted
731 #ifndef SHARED_HANDLERS
732     theApp.GetContextMenuManager()->ShowPopupMenu(IDR_POPUP_EDIT,
    point.x, point.y, pWnd, TRUE);
733 #endif
734     return TRUE;
735 }
736
737 int CTextEditorObject::OnRecieveText(CString text, BOOL open)
738 {
739     int itPos; // This is needed as it gets changed in this-
    >brackContains
740     int carriageOccur = 0;
741
742     if ((text == L"(") && this->bracketContains(this->caretPos,
    itPos, 0, 1)) ||
743         (text == L")") && this->bracketContains(this->caretPos,
    itPos, 1, 1)) ||
744         (text == L"]") && this->bracketContains(this->caretPos,
    itPos, 2, 1)) ||
745         (text == L"\\" && this->bracketContains(this->caretPos,
    itPos, 3, 1)) ||
746         (text == L"/" && this->bracketContains(this->caretPos,
    itPos, 4, 1))) {
747
748     this->brackets.erase(this->brackets.begin() + itPos);
749     this->brackets.shrink_to_fit();
750
751     this->caretPos++;
752 }
753
754 else {
755     int carriage = text.Find(L"\n");
756     CString recursiveText;
757     if (carriage != -1) {
758
759         recursiveText = text.Mid(carriage + 1);
760         text = text.Mid(0, carriage);
761     }

```

```
762         this->lines[this->activeLine - 1]->concatenateString(text, ↵
    763             this->caretPos);
    764
    765     if (text == L"(") {
    766
        this->lines[this->activeLine - 1]->concatenateString ↵
            (L")", this->caretPos + 1);
        this->moveBrackets(2, this->caretPos);
    768
    769     this->brackets.push_back({ this->caretPos, this-      ↵
        >caretPos + 1, 0 });
    770 }
    771 else if (text == L"{") {
    772
        this->lines[this->activeLine - 1]->concatenateString ↵
            (L"}", this->caretPos + 1);
        this->moveBrackets(2, this->caretPos);
    775
    776     this->brackets.push_back({ this->caretPos, this-      ↵
        >caretPos + 1, 1 });
    777 }
    778 else if (text == L"[") {
    779
        this->lines[this->activeLine - 1]->concatenateString ↵
            (L"]", this->caretPos + 1);
        this->moveBrackets(2, this->caretPos);
    782
    783     this->brackets.push_back({ this->caretPos, this-      ↵
        >caretPos + 1, 2 });
    784 }
    785 else if (text == L"\\"") {
    786
        this->lines[this->activeLine - 1]->concatenateString ↵
            (L"\\"", this->caretPos + 1);
        this->moveBrackets(2, this->caretPos);
    789
    790     this->brackets.push_back({ this->caretPos, this-      ↵
        >caretPos + 1, 3 });
    791 }
    792 else if (text == L"'''") {
    793
        this->lines[this->activeLine - 1]->concatenateString ↵
            (L"''", this->caretPos + 1);
        this->moveBrackets(2, this->caretPos);
    796
    797     this->brackets.push_back({ this->caretPos, this-      ↵
        >caretPos + 1, 4 });
    798 }
    799 else {
        this->moveBrackets(text.GetLength(), this->caretPos);
    }
    802     this->caretPos += text.GetLength();
    803
    804     if (carridge != -1) {
    805
        this->OnRecieveReturn(open);
```

```
807             carriageOccur++;
808             carriageOccur += this->OnRecieveText(recursiveText);
809         }
810     }
811     return carriageOccur;
812 }
813 void CTextEditorObject::OnRecieveBackspace()
814 {
815     if (!this->hlght && this->activeLine > 0 && this->caretPos != 0) { // Remove a character
816
817         CString activeChar = this->lines[this->activeLine - 1]->getText().Mid(this->caretPos - 1, 1);
818         int itPos;
819
820         if (((activeChar == L"(" || activeChar == L")" || activeChar == L"[ " || activeChar == L"]" || activeChar == L"\"") && this->bracketContains(this->caretPos - 1, itPos, 5, 0))) {
821             this->lines[this->activeLine - 1]->backspace(this->brackets[itPos][1]);
822             this->moveBrackets(-1, this->brackets[itPos][1]);
823
824             this->brackets.erase(this->brackets.begin() + itPos);
825             this->brackets.shrink_to_fit();
826         }
827
828     else {
829
830         CString spaces = L"";
831
832         if ((this->caretPos % theApp.indentSize) == 0) {
833             spaces = L"    ";
834
835             if (this->lines[this->activeLine - 1]->getText().Mid(this->caretPos - theApp.indentSize, theApp.indentSize) == spaces) {
836
837                 for (int j = 0; j < theApp.indentSize-1; j++) {
838
839                     this->lines[this->activeLine - 1]->backspace(this->caretPos);
840                     this->moveBrackets(-1, this->caretPos);
841                     this->caretPos--;
842                 }
843             }
844         }
845     else {
846         for (int i = 0; i < (this->caretPos % theApp.indentSize); i++) {
847             spaces += L" ";
848         }
849
850         if (this->lines[this->activeLine - 1]->getText().Mid(this->caretPos - (this->caretPos % theApp.indentSize), (this->caretPos %
```

```
...ce\repos\DesignArk\DesignArk\CTextEditorObject.cpp 19
                                theApp.indentSize)) == spaces) {
851
852             int it = (this->caretPos % theApp.indentSize) - 1;
853
854             for (int j = 0; j < it; j++) {
855
856                 this->lines[this->activeLine - 1]->backspace(this->caretPos);
857                 this->moveBrackets(-1, this->caretPos);
858                 this->caretPos--;
859             }
860         }
861     }
862 }
863 this->lines[this->activeLine - 1]->backspace(this->caretPos);
864 this->moveBrackets(-1, this->caretPos);
865 this->caretPos--;
866 }
867
868 else if (!this->hlghting && !this->hlght && this->activeLine > 1 && this->caretPos == 0) { // Remove a line
869
870     this->bounds.bottom -= static_cast<int>(this->defBoxHeight * theApp.zoom);
871
872     this->caretPos = this->lines[this->activeLine - 2]->getLength();
873     this->lines[this->activeLine - 2]->concatenateString(this->lines[this->activeLine - 1]->getText(), this->caretPos);
874
875     for (int i = this->lines.size() - 1; i >= this->activeLine; i--) {
876         this->lines[i]->move(0, -static_cast<int>(this->defBoxHeight * theApp.zoom));
877         this->lines[i]->setID(this->lines[i - 1]->getID());
878         this->lines[i]->incrementLine(this->lines[i]->iGetLineNums().size() - 1, -1);
879     }
880
881     this->lines.erase(this->lines.begin() + this->activeLine - 1);
882     this->activeLine--;
883     this->lines[this->activeLine - 1]->setActive(TRUE);
884
885     this->setSidebar();
886 }
887
888 else if (this->hlght) { // Remove a highlight
889
890     BOOL first = FALSE;
891     int i = 0;
892     BOOL end = FALSE;
893
894     while (!end && i < this->lines.size()) { // Iterate through
```

```
    all the lines
895
896        if (this->lines[i]->isHighlighting()) { // If the line ↵
897            is highlighting
898                if (!first) { // If this is the first line that we ↵
899                    have found
900                        this->activeLine = i + 1; // This will become ↵
901                        our active line
902                            this->caretPos = this->lines[i]->getHStart ↵
903 () ; // Set caret at the end of the line
904                            this->lines[i]->backspace(this->caretPos); // ↵
905 Remove all that we have highlighted on this line
906                            this->lines[i]->setHighlighting(FALSE); // Tell ↵
907 it we are no longer highlighting
908
909                first = TRUE; // Let the loop know that we have ↵
910 found the first highlighted line
911 }
912 else if(!this->lines[i]->isLineHighlighted() || i ↵
913 == this->lines.size() - 1) { // If the entire line ↵
914 is not highlighted
915
916     this->lines[i]->backspace(this->caretPos);
917     this->lines[this->activeLine - 1]->
918 >concatenateString(this->lines[i]->getText(), this->caretPos);
919
920         for (int j = i; j >= this->activeLine; j--) {
921
922             this->lines.erase(this->lines.begin() + j);
923
924                 for (int k = this->lines.size() - 1; k >= this->activeLine; k--) {
925
926                     this->lines[k]->move(0, -static_cast<int>(
927 (this->defBoxHeight * theApp.zoom) * (i - this->activeLine + 1));
928                     this->lines[k]->setID(this->lines[k]->getID());
929                     this->lines[k]->incrementLine(this->lines[k]->iGetLineNums().size() - 1, -(i - this->activeLine + 1));
930
931             this->bounds.bottom -= static_cast<int>(this->defBoxHeight * theApp.zoom) * (i - this->activeLine + 1);
932
933                 end = TRUE;
934             }
935         }
936     else if (first) {
```

```
931             end = TRUE;
932         }
933         i++;
934     }
935
936     this->highlightingOff();
937     this->setSidebar();
938 }
939
940 void CTextEditorObject::OnReceiveTab()
941 {
942     int itPos;
943     if (this->bracketContains(this->caretPos, itPos, 5, 1)) {
944
945         this->brackets.erase(this->brackets.begin() + itPos);
946         this->brackets.shrink_to_fit();
947
948         this->caretPos++;
949     }
950     else {
951         CString indent = L"";
952         for (int i = 0; i < theApp.indentSize; i++) {
953             indent.Append(L" ");
954         }
955         this->OnReceiveText(indent);
956     }
957 }
958 void CTextEditorObject::OnReceiveReturn(BOOL open)
959 {
960     if (this->maxLines > this->lines.size() || this->maxLines == 0) {
961         // If there is space for another line
962
963         this->bounds.bottom += static_cast<int>(this->defBoxHeight *
964             theApp.zoom);
965         CString newLineText = L"";
966         int numTab = 0;
967
968         if (!open) {
969             BOOL cln = TRUE, checkEnd = TRUE, checkStart = TRUE;
970             int cmtPos = -1, clnPos = -1;
971
972             int frontIt = 0;
973
974             for (int i = this->lines[this->activeLine - 1]->getText().GetLength() - 1; i >= 0; i--) {
975
976                 if ((this->lines[this->activeLine - 1]->getText().Mid(i, 1) != L":" && this->lines[this->activeLine - 1]->getText().Mid(i, 1) != L" "/* && */
977                     i == 0 // This may need put back if bugs are found */ && checkEnd) {
978                     cln = FALSE;
979                 }
980                 else if (this->lines[this->activeLine - 1]->getText().Mid(i, 1) == L":" && cln) {
981                     clnPos = i;
```

```
979                     checkEnd = FALSE;
980                 }
981
982                 if ((this->lines[this->activeLine - 1]->getText      ↵
983                   () .Mid(frontIt, 1) != L" " || frontIt == this-      ↵
984                   >lines[this->activeLine - 1]->getText().GetLength    ↵
985                   () - 1) && checkStart) {
986
987                     numTab = (frontIt - (frontIt %           ↵
988                     theApp.indentSize)) / theApp.indentSize;
989                     checkStart = FALSE;
990                 }
991                 if (this->lines[this->activeLine - 1]->getText      ↵
992                   () .Mid(frontIt, theApp.commentType.GetLength()) ==    ↵
993                   theApp.commentType) {
994                     cmtPos = frontIt;
995                 }
996
997                 frontIt++;
998             }
999
1000             if (frontIt == 0) {
1001                 cln = FALSE;
1002             }
1003
1004             if (cln && (cmtPos >= clnPos || cmtPos == -1)) {
1005                 numTab++;
1006             }
1007
1008             CString indent = L"";
1009             for (int i = 0; i < theApp.indentSize; i++) {
1010                 indent.Append(L" ");
1011             }
1012
1013             for (int i = 0; i < numTab; i++) {
1014                 newLineText.Append(indent);
1015             }
1016         }
1017         newLineText.Append(this->lines[this->activeLine - 1]-      ↵
1018             >getText().Mid(this->caretPos));
1019
1020         auto itpos = this->lines.begin() + this->activeLine;
1021
1022         auto newit = this->lines.insert(
1023             itpos,
1024             new CTextLineObject(
1025                 this->lines[this->activeLine - 1]->getBounds(),
1026                 L"0",
1027                 this->lines[this->activeLine - 1]->iGetLineNums(),
1028                 newLineText,
1029                 TRUE)); // Add a new line
1030
1031
1032         this->lines[this->activeLine - 1]->setText(this->lines      ↵
1033             [this->activeLine - 1]->getText().Mid(0, this-      ↵
1034             >caretPos));
```

```
1026         this->lines[this->activeLine - 1]->setActive(FALSE); // Set ↵
1027             old line not active
1028
1029         this->caretPos = 0 + numTab * theApp.indentSize;
1030         this->activeLine++; // Change position for editing
1031
1032         CString id;
1033
1034         for (int i = this->activeLine-1; i < this->lines.size(); i+ ↵
1035             +) {
1036
1037             id.Format(L"%d", i);
1038             this->lines[i]->move(0, static_cast<int>(this-      ↵
1039                 >defBoxHeight * theApp.zoom));
1040             this->lines[i]->setID(id);
1041             this->lines[i]->incrementLine(this->lines[i]-      ↵
1042                 >iGetLineNums().size()-1, 1);
1043         } // Move all lines and reset all ids
1044     }
1045
1046 void CTextEditorObject::OnKeyDown(UINT nChar, UINT nRepCnt, UINT      ↵
1047     nFlags)
1048 {
1049     switch (nChar) {
1050
1051     case VK_LEFT:
1052         if (this->caretPos > 0) {
1053             this->caretPos--;
1054
1055             int itPos;
1056
1057             if (this->bracketContains(this->caretPos, itPos, 5, 0)) ↵
1058                 {
1059                     this->setBracketsNull();
1060                 }
1061             else if (this->caretPos == 0 && this->activeLine > 1) {
1062
1063                 this->lines[this->activeLine - 1]->setActive(FALSE);
1064                 this->activeLine--;
1065                 this->lines[this->activeLine - 1]->setActive(TRUE);
1066
1067                 this->setBracketsNull();
1068
1069                 this->caretPos = this->lines[this->activeLine - 1]-      ↵
1070                     >getLength();
1071             }
1072             break;
1073
1074     case VK_UP:
1075         if (this->activeLine > 1) {
```

```

...ce\repos\DesignArk\DesignArk\CTextEditorObject.cpp
1075         this->lines[this->activeLine - 1]->setActive(FALSE);
1076         this->activeLine--;
1077         this->lines[this->activeLine - 1]->setActive(TRUE);
1078
1079         this->setBracketsNull();
1080     }
1081     break;
1082
1083 case VK_RIGHT:
1084     if (this->caretPos < this->lines[this->activeLine - 1]-
1085         >getLength()) {
1086
1087         int itPos;
1088
1089         if (this->bracketContains(this->caretPos, itPos, 5, 1)) {
1090
1091             this->brackets.erase(this->brackets.begin() +
1092             itPos);
1093             this->brackets.shrink_to_fit();
1094
1095             this->caretPos++;
1096         }
1097         else if (this->caretPos == this->lines[this->activeLine - 1]-
1098             >getLength() && this->activeLine != this->lines.size -
1099             ()) {
1100
1101             this->lines[this->activeLine - 1]->setActive(FALSE);
1102             this->activeLine++;
1103             this->lines[this->activeLine - 1]->setActive(TRUE);
1104
1105             this->caretPos = 0;
1106         }
1107         break;
1108
1109 case VK_DOWN:
1110     if (this->activeLine < this->lines.size()) {
1111
1112         this->lines[this->activeLine - 1]->setActive(FALSE);
1113         this->activeLine++;
1114         this->lines[this->activeLine - 1]->setActive(TRUE);
1115
1116     break;
1117
1118     }
1119 }
1120
1121 // Public Queries
1122 std::vector <int> CTextEditorObject::iGetLineNum(int a)
1123 {
1124     if (a != 0) {

```

```
1126         return this->lines[a-1]->iGetLineNums();
1127     }
1128
1129     else if (this->clickSubLine) {
1130         this->clickSubLine = FALSE;
1131         return this->lines[this->activeLine-1]->iGetLineNums();
1132     }
1133     return std::vector <int> { 0 };
1134 }
1135 CString CTextEditorObject::sGetLineNum(int a)
1136 {
1137     if (a != 0) {
1138         return this->lines[a - 1]->sGetLineNums();
1139     }
1140
1141     else if (this->clickSubLine) {
1142         this->clickSubLine = FALSE;
1143         return this->lines[this->activeLine - 1]->sGetLineNums();
1144     }
1145     return L"0";
1146 }
1147
1148 int CTextEditorObject::getActiveLine()
1149 {
1150     return this->activeLine-1;
1151 }
1152 int CTextEditorObject::getBlockLine()
1153 {
1154     return this->blockLine - 1;
1155 }
1156 int CTextEditorObject::getPrintLine()
1157 {
1158     return this->printLine;
1159 }
1160 CString CTextEditorObject::getLineText(int line)
1161 {
1162     if (line == 0) {
1163         return this->lines[this->activeLine - 1]->getText();
1164     }
1165     return this->lines[line - 1]->getText();
1166 }
1167 CRect CTextEditorObject::getLineBounds(int line)
1168 {
1169     return this->lines[line]->getBounds();
1170 }
1171 CString CTextEditorObject::getHighlightedText()
1172 {
1173     if (this->hlght || this->hlghting) {
1174
1175         int i = 0;
1176         BOOL done = FALSE;
1177
1178         CString selectedText = L"";
1179
1180         while (!done && i < this->lines.size()) {
1181
```

```
1182         if (this->lines[i]->isHighlighting()) {
1183
1184             int j = i;
1185             BOOL it = TRUE;
1186
1187             while (it && this->lines[j]->isHighlighting()) {
1188
1189                 selectedText.Append(this->lines[j]-
1190 >getHighlightedText());
1191                 selectedText.Append(L"\n");
1192
1193                 if (j == this->lines.size() - 1) {
1194                     it = FALSE;
1195                 }
1196                 else {
1197                     j++;
1198                 }
1199
1200             selectedText = selectedText.Mid(0,
1201                 selectedText.GetLength() - 1);
1202             done = TRUE;
1203         }
1204     }
1205
1206     return selectedText;
1207 }
1208 else {
1209     return CString();
1210 }
1211 }
1212 int CTextEditorObject::getLineTextWidth()
1213 {
1214     int max = this->lines[0]->getText().GetLength();
1215
1216     for (int i = 1; i < this->lines.size(); i++) {
1217         if (max < this->lines[i]->getText().GetLength()) {
1218             max = this->lines[i]->getText().GetLength();
1219         }
1220     }
1221
1222     max *= this->textExtent.cx;
1223     max += 2;
1224
1225     return max;
1226 }
1227
1228 int CTextEditorObject::getNumLines()
1229 {
1230     return this->lines.size();
1231 }
1232
1233 int CTextEditorObject::getBoxHeight(BOOL default)
1234 {
1235     if (default) {
```

```

1236         return this->defBoxHeight;
1237     }
1238     return static_cast<int>(this->defBoxHeight * theApp.zoom);
1239 }
1240
1241 void CTextEditorObject::setSidebar()
1242 {
1243     if (this->lineNums) {
1244         this->sidebar.top = this->bounds.top;
1245         this->sidebar.left = this->bounds.left;
1246         this->sidebar.bottom = this->bounds.top + this->lines.size ↵
1247             () * static_cast<int>(this->defBoxHeight * theApp.zoom);
1248         this->sidebar.right = this->bounds.left + static_cast<int> ↵
1249             (this->defBoxHeight * theApp.zoom);
1250     }
1251 }
1252 void CTextEditorObject::setHighlighter()
1253 {
1254     // Find which lines the hlght points lie in
1255     int i = 1;
1256     BOOL startP = FALSE, endP = FALSE;
1257     int endLine = 0;
1258     int endPos = 0;
1259
1260     if (this->recentLine != this->activeLine) {
1261         this->recentLine = this->activeLine;
1262     }
1263     if (this->recentPos != this->caretPos) {
1264         this->recentPos = this->caretPos;
1265     }
1266     if (this->recentHlght != this->hlghting) {
1267         this->recentHlght = this->hlghting;
1268     }
1269
1270     while (!(startP && endP) && i <= this->lines.size()) {
1271
1272         if (!startP && this->hlghtStartP.y >= this->lines[i - 1]- ↵
1273             >getBounds().top && this->hlghtStartP.y < this->lines[i - 1]- ↵
1274             >getBounds().bottom && !this->hlghting) {
1275
1276             this->startLine = i;
1277             if (this->hlghtStartP.x > (this->lineOffset * ↵
1278                 theApp.zoom)) {
1279                 this->startPos = (this->hlghtStartP.x - (this- ↵
1280                     >lineOffset * theApp.zoom)) / this->textExtent.cx;
1281             }
1282             else {
1283                 this->startPos = 0;
1284             }
1285
1286             this->hlghting = TRUE;
1287             startP = TRUE;
1288         }
1289
1290         if (!endP && this->hlghtEndP.y >= this->lines[i - 1]- ↵
1291             >getBounds().top && this->hlghtEndP.y < this->lines[i - 1]- ↵

```

```
1] ->getBounds().bottom) {  
1285  
1286             endLine = i;  
1287             if (this->hlghtEndP.x > (this->lineOffset *  
1288                 theApp.zoom)) {  
1289                 endPos = (this->hlghtEndP.x - (this->lineOffset *  
1290                     theApp.zoom)) / this->textExtent.cx;  
1291             }  
1292             else {  
1293                 endPos = 0;  
1294             }  
1295             endP = TRUE;  
1296         }  
1297         i++;  
1298     }  
1299     if (!endP) {  
1300         endLine = this->lines.size();  
1301         endPos = this->lines[endLine - 1]->getLength();  
1302     }  
1303  
1304     // Unhighlight all lines up to start line and after end line  
1305     for (int i = 0; i < this->startLine - 1; i++) {  
1306         this->lines[i]->setHighlighting(FALSE);  
1307     }  
1308  
1309     for (int i = endLine; i < this->lines.size(); i++) {  
1310         this->lines[i]->setHighlighting(FALSE);  
1311     }  
1312  
1313     // If we need to highlight over multiple lines  
1314     if (this->startLine != endLine) {  
1315  
1316         // Reset the caret on the highlighter  
1317         this->lines[this->activeLine - 1]->setActive(FALSE);  
1318         this->activeLine = endLine;  
1319  
1320         // If we need to highlight in reverse  
1321         if (this->startLine > endLine) {  
1322  
1323             if (endPos >= this->lines[endLine - 1]->getLength() +  
1324                 1) {  
1325                 endPos = this->lines[endLine - 1]->getLength();  
1326             }  
1327  
1328             this->lines[endLine - 1]->setHighlighter(endPos, this->  
1329                 lines[endLine - 1]->getLength() + 1);  
1330             this->lines[endLine - 1]->setHighlighting(TRUE);  
1331  
1332             // To make sure the highlight doesn't go past the  
1333                 length of the last line  
1334             if (this->startPos > this->lines[this->startLine - 1]->  
1335                 getLength()) {  
1336                 if (this->lines[this->startLine - 1]->getLength()  
1337                     == 0) {  
1338
```

```
1333             this->startPos = 2;
1334         }
1335     else {
1336         this->startPos = this->lines[this->startLine - 1]->getLength() + 1;
1337     }
1338 }
1339
1340 this->lines[this->startLine - 1]->setHighlighter(0, this->startPos);
1341 this->lines[this->startLine - 1]->setHighlighting(true);
1342
1343 // Highlight all lines inbetween end and start line
1344 for (int i = endLine + 1; i < this->startLine; i++) {
1345
1346     this->lines[i - 1]->highlightLine();
1347     this->lines[i - 1]->setHighlighting(true);
1348 }
1349 }
1350 else {
1351
1352     this->lines[this->startLine - 1]->setHighlighter(this->startPos, this->lines[this->startLine - 1]->getLength() + 1);
1353     this->lines[this->startLine - 1]->setHighlighting(true);
1354
1355 // To make sure the highlight doesn't go past the length of the last line
1356 if (endPos > this->lines[this->activeLine - 1]->getLength()) {
1357     if (this->lines[this->activeLine - 1]->getLength() == 0) {
1358         endPos = 1;
1359     }
1360     else {
1361         endPos = this->lines[this->activeLine - 1]->getLength();
1362     }
1363 }
1364
1365 this->lines[endLine - 1]->setHighlighter(0, endPos);
1366 this->lines[endLine - 1]->setHighlighting(true);
1367
1368 // Highlight all lines inbetween start and end line
1369 for (int i = startLine + 1; i < endLine; i++) {
1370
1371     this->lines[i - 1]->highlightLine();
1372     this->lines[i - 1]->setHighlighting(true);
1373 }
1374 }
1375 this->caretPos = endPos;
1376 }
1377
1378 // If it is just a one line highlight
```

```
1379     else {
1380
1381         if (this->startPos > this->lines[this->startLine - 1]-
1382             >getLength() && endPos > this->lines[this->startLine -
1383             1]->getLength()) {
1384             this->hlghting = FALSE;
1385         }
1386
1387         else {
1388             if (this->startPos > this->lines[this->startLine - 1]-
1389                 >getLength()) {
1390                 this->startPos = this->lines[this->startLine - 1]-
1391                     >getLength();
1392             }
1393             if (endPos > this->lines[this->startLine - 1]-
1394                 >getLength()) {
1395                 endPos = this->lines[this->startLine - 1]-
1396                     >getLength();
1397                 this->caretPos = endPos + 1;
1398             }
1399             else {
1400                 this->caretPos = endPos;
1401             }
1402
1403             this->lines[this->startLine - 1]->setHighlighter(this-
1404                 >startPos, endPos);
1405             this->lines[this->startLine - 1]->setHighlighting
1406                 (TRUE);
1407
1408             this->activeLine = this->startLine;
1409         }
1410     }
1411
1412     void CTextEditorObject::setBracketsNull()
1413     {
1414         for (auto& it : this->brackets) {
1415             it = std::vector<int>{};
1416         }
1417     }
1418
1419     BOOL CTextEditorObject::bracketContains(int value, int& itPos, int
1420         type, int side)
1421     {
1422         int i = 0;
1423
1424         for (auto& it : this->brackets) {
1425
1426             if (it[side] == value) {
1427                 if (it[2] == type || type == 5) {
1428                     itPos = i;
1429                     return TRUE;
1430                 }
1431             }
1432             i++;
1433         }
1434     }
```

```
...ce\repos\DesignArk\DesignArk\CTextEditorObject.cpp
1426     return FALSE;
1427 }
1428 void CTextEditorObject::moveBrackets(int val, int index)
1429 {
1430     for (auto& bracket : this->brackets) {
1431         if (index <= bracket[0]) {
1432             bracket[0] += val;
1433             bracket[1] += val;
1434         }
1435         else if (index <= bracket[1]) {
1436             bracket[1] += val;
1437         }
1438     }
1439 }
1440 }
1441
1442
```

```
1 #pragma once
2
3 #include "CAppObject.h"
4
5 class CTextLineObject
6     : public CAppObject
7 {
8 public:
9     // Public constructors
10    CTextLineObject(CRect bounds, CString ID, std::vector<int>      ↪
11                    lineNums, CString text = L"", BOOL active = FALSE);
12    ~CTextLineObject();
13
14    //DECLARE_SERIAL(CTextLineObject)
15
16    // Public commands
17    int draw(CDC* pDC, CSize textExtent,
18              std::vector<std::vector<int>> brackets, int returnNewLines,      ↪
19              BOOL printing, int printAreaLength = -1);      ↪
20
21    // Public getters & setters
22    CString getText();
23    int getLength();
24    void setText(CString text);
25    void concatenateString(CString text, int position);
26    void backspace(int position);
27
28    BOOL isHighlighting();
29    void setHighlighting(BOOL val);
30    void setHighlighter(int pos1, int pos2);
31    void highlightLine();
32    BOOL isLineHighlighted();
33    int getHStart();
34    int getHEnd();
35    CString getHighlightedText();
36
37    int getNumSubLines();
38    std::vector<int> iGetLineNums();
39    CString sGetLineNums();
40    void addSublines(std::vector<int> subs);
41    void incrementLine(int subline, int val);
42
43 private:
44
45     // Private Functions
46     BOOL onlyNums(CString str);
47     BOOL brkPosContains(std::vector<std::vector<int>> vector, int      ↪
48                         value, int side = 1);
49
50     // Private resources
51     CString text;
52     Gdiplus::Rect highlighter;
```

```
53     BOOL highlight;
54     int hStart;
55     int hEnd;
56     BOOL lineHighlight;
57
58     int numSubLines;
59     std::vector<int> lineNums;
60
61     BOOL smartColour;
62 };
63
64
```

```
1 #include "pch.h"
2 #include "CTextLineObject.h"
3
4 // Public constructors
5 CTextLineObject::CTextLineObject(CRect bounds, CString ID,
6     std::vector<int> lineNums, CString text, BOOL active)    ↵
7     : CAppObject(bounds, ID, active)
8 {
9     this->text = text;
10
11    this->highlighter.Y = bounds.top;
12    this->highlighter.Height = bounds.Height();
13    this->highlight = FALSE;
14    hStart = 0;
15    hEnd = 0;
16
17    this->numSubLines = lineNums.size();
18    this->lineNums = lineNums;
19
20    this->smartColour = TRUE;
21 }
22 CTextLineObject::~CTextLineObject()
23 {
24
25 // Public commands
26 int CTextLineObject::draw(CDC* pDC, CSize textExtent,      ↵
27     std::vector<vector<int>> brackets, int returnNewLines, BOOL      ↵
28     printing, int printAreaLength)      ↵
29 {
30     // Local variables
31     BOOL autoBrkt = brackets.size() > 0; // If we have any auto      ↵
32         brackets (used when not printing)
33     int lengthPrint = 0; // Stores the current length of the print      ↵
34         on the line that is currently being drawn to (only used when      ↵
35         printing)
36
37     // Gdiplus::Bitmap newLineArrow(L"res/newLine.png"); // Taking      ↵
38         this out as it does not work and is not needed
39     int printMaxCharLength = (printAreaLength - this->bounds.left *      ↵
40         2)/textExtent.cx;
41
42     // If we are using SmartColour
43     if (this->smartColour) {
44
45         // Local variables
46         CString word = L""; // Stores the word that is being found      ↵
47             and is being colour checked
48         CString character = L""; // Stores any characters that      ↵
49             should separate words
50         int pos = 0; // Stores the position that is being drawn at
51         BOOL comment = FALSE; // Stores whether the rest of the line      ↵
52             is a comment
53         int i = 0; // iterator for the loop, letter position that is      ↵
54             being looked at
55 }
```

```

45         // Iterate through characters in the line
46         while (i < this->text.GetLength() + 1 && !comment) {
47
48             // Checks to see if there is a comment
49             if (this->text.Mid(i, theApp.commentType.GetLength()) != ↪
50                 theApp.commentType) {
51
52                 // Character of the current position of the loop
53                 character = this->text.Mid(i, 1);
54                 BOOL hit = FALSE;
55
56                 // Checks to see if the character is a 'break'      ↪
57                 // character - ie, if the character should seperate    ↪
58                 // words - or if we are at the end of the loop
59                 if (character == L" " ||
60                     character == L"!" ||
61                     character == L"£" ||
62                     character == L"$" ||
63                     character == L "%" ||
64                     character == L"^" ||
65                     character == L"&" ||
66                     character == L"*" ||
67                     character == L "(" ||
68                     character == L ")" ||
69                     character == L "+" ||
70                     character == L "-" ||
71                     character == L "/" ||
72                     character == L ":" ||
73                     character == L ";" ||
74                     character == L "=" ||
75                     character == L "<" ||
76                     character == L "," ||
77                     character == L "_" ||
78                     character == L "-" ||
79                     character == L ">" ||
80                     character == L"." ||
81                     character == L "?" ||
82                     character == L "/" ||
83                     character == L "@" ||
84                     character == L "{" ||
85                     character == L "[" ||
86                     character == L "]" ||
87                     i == this->text.GetLength()) {
88
89                 BOOL newColour = FALSE; // Stores if we have      ↪
90                 // changed the colour, and if it needs to be changed    ↪
91                 // back
92
93                 // Checks if the word is a number
94                 if (this->onlyNums(word)) {
95                     // If so, sets the word to the number colour
96                     pDC->SetTextColor(theApp.numberColour);
97                     newColour = TRUE;

```

```

96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
}
else { // If the word is not a number colour
    // Binary search through all of the words to ↵
    find if the current word matches one in the list
    int it = 0, n =
        theApp.smartColour_String.size();
while (!newColour && it < n) {
    // Binary Search
    int it2 = 0, n2 =
        theApp.smartColour_String[it].size();
    int min = 0, max = (n2)-1;
while (!newColour && it2 < n2) {
    if (max < min) {
        newColour == TRUE;
    }
    int guess = floor((min + max) / 2);
    CString tempWord = word;
    CString tempAppWord =
        theApp.smartColour_String[it][guess];
    tempWord.MakeLower();
    tempAppWord.MakeLower();

    // If this word on the list matches ↵
    our searching word
    if (tempWord == tempAppWord) {
        newColour = TRUE;
        pDc->SetTextColor
        (theApp.smartColour_Colour[it]); // Set the word's ↵
        colour
    }
    else if (theApp.smartColour_String ↵
        [it][guess] < word) {
        min = guess + 1;
    }
    else {
        max = guess - 1;
    }
    it2++;
}
it++;
}
}

// Drawing the word
// Drawing if we have a printer
if (printing) {
}

```

```

...urce\repos\DesignArk\DesignArk\CTextLineObject.cpp 4
144                                     // Add the length if the word to the length ↵
145                                     of our print
146                                     lengthPrint += word.GetLength();
147
148                                     while (lengthPrint > printMaxCharLength) ↵
149                                     { // As a word could technically have a length ↵
150                                     greater than 50, we must use a while to check for ↵
151                                     lines
152                                     CString tempWord = L"";
153                                     if ((i - word.GetLength()) % ↵
154                                     printMaxCharLength != 0) {
155                                         tempWord = word.Mid(0, ↵
156                                         printMaxCharLength - ((i - word.GetLength()) % ↵
157                                         printMaxCharLength)); // Create a temporary word to ↵
                                         store the start of the word that will go on the ↵
                                         end of the current line
158                                         word = word.Mid(printMaxCharLength - ↵
                                         ((i - word.GetLength()) % printMaxCharLength)); // ↵
                                         Change the word to be whatever we didn't print and ↵
                                         continue the loop
159                                         }
160                                         else {
161                                             pos++;
162                                         }
163
164                                     pDC->TextOut(this->bounds.left + 2 + ↵
165                                     (textExtent.cx * pos), this->bounds.top + this- ↵
166                                     >bounds.Height() / 2 - textExtent.cy / 2 + ↵
167                                     returnNewLines * this->bounds.Height(), tempWord, ↵
168                                     tempWord.GetLength()); // Print what we can to fill ↵
                                         the line
169
170                                     lengthPrint -= printMaxCharLength; // ↵
171                                     Since we are creating a new line, subtract the old ↵
                                         60 characters from this
172
173                                     /*pos += tempWord.GetLength() + 2;
174
175                                     Gdiplus::Graphics g(pDC->GetSafeHdc());
176                                     Gdiplus::Rect expansionRect(this- ↵
177                                     >bounds.left + 2 + (textExtent.cx * pos), this- ↵
178                                     >bounds.top + this->bounds.Height() / 2 + ↵
179                                     returnNewLines * this->bounds.Height(),
180                                         textExtent.cx, textExtent.cx);
181
182                                     g.DrawImage(&newLineArrow,
183                                     expansionRect);*/
184
185                                     returnNewLines++;
186                                     pos = 0;
187                                     }
188                                     // Print the 'scraps' of the word at the end ↵
189                                     on the next line
190                                     pDC->TextOut(this->bounds.left + 2 + ↵
191                                     (textExtent.cx * pos), this->bounds.top + this- ↵

```

```

...urce\repos\DesignArk\DesignArk\CTextLineObject.cpp 5
    >bounds.Height() / 2 - textExtent.cy / 2 +
    returnNewLines * this->bounds.Height(), word,
    word.GetLength());
175
176         // Reset the position to draw from
177         pos += word.GetLength();
178     }
179     // Not using printer
180     else {
181         pDC->TextOut(this->bounds.left + 2 +
    (textExtent.cx * pos), this->bounds.top + this-
    >bounds.Height() / 2 - textExtent.cy / 2, word,
    word.GetLength());
182
183         pos = i + 1;
184
185         /*if (character != L"" && character !=      ↵
    L'') {
186             pos++;
187             }*/
188         }
189
190         word = L""; // Reset the word since we are no      ↵
longer using it
191
192         // Now we print the character that we used to      ↵
find the seperation
193
194         BOOL grey = FALSE; // Stores if we used the      ↵
autobracket
195
196         // If there is a auto bracket for the character      ↵
(never used when printing)
197         if (this->active && autoBrkt && (character ==      ↵
L")" || character == L"}" || character == L"]") &&      ↵
this->brkPosContains(brackets, i) && !printing) {
198
199             grey = TRUE;
200             newColour = TRUE;
201
202             pDC->SetTextColor(RGB(128, 128, 128)); //      ↵
Set colour to grey for the auto bracket
203
204             pDC->TextOut(this->bounds.left + 3 +
    (textExtent.cx * (i + word.GetLength()))), this-
    >bounds.top + this->bounds.Height() / 2 -
    textExtent.cy / 2, character, character.GetLength
    ()); // Draw the auto bracket
205         }
206
207         // If, at any point, changed the colour
208         if (newColour) {
209             pDC->SetTextColor(RGB(0, 0, 0)); // Revert      ↵
colour back to black (original colour)
210         }
211

```

```

...urce\repos\DesignArk\DesignArk\CTextLineObject.cpp 6
212                                     // If the character was not an auto bracket
213                                     if(!grey && character != L"\\" && character != L"\\") {
214                                         lengthPrint += character.GetLength();
215
216                                         if (printing) {
217
218                                             if (lengthPrint > printMaxCharLength) {
219
220
221                                             pDc->TextOut(this->bounds.left + 2,    ↵
222                                             this->bounds.top + this->bounds.Height() / 2 -   ↵
223                                             textExtent.cy / 2, character, character.GetLength   ↵
224                                             ());
225
226                                         /*pos += character.GetLength() + 2;
227
228                                         Gdiplus::Graphics g(pDc->GetSafeHdc   ↵
229                                         ());
230                                         Gdiplus::Rect expansionRect(this-    ↵
231                                         >bounds.left + 2 + (textExtent.cx * pos), this-    ↵
232                                         >bounds.top + this->bounds.Height() / 2 +   ↵
233                                         returnNewLines * this->bounds.Height(),   ↵
234                                         textExtent.cx, textExtent.cy);
235
236                                         g.DrawImage(&newLineArrow,           ↵
237                                         expansionRect);*/
238
239                                         lengthPrint -=
240                                         printMaxCharLength; // Since we are creating a new ↵
241                                         line, subtract the old 60 characters from this ↵
242                                         returnNewLines++;
243                                         pos = 0;
244
245                                         }
246                                         else {
247
248                                             pDc->TextOut(this->bounds.left + 2 + ↵
249                                             (textExtent.cx * pos), this->bounds.top + this-    ↵
250                                             >bounds.Height() / 2 - textExtent.cy / 2 +   ↵
251                                             returnNewLines * this->bounds.Height(), character,   ↵
252                                             character.GetLength());
253                                         }
254
255                                         pos++;
256                                         }
257                                         else {
258
259                                             pDc->TextOut(this->bounds.left + 2 + ↵
260                                             (textExtent.cx * (i + word.GetLength()))), this-    ↵
261                                             >bounds.top + this->bounds.Height() / 2 -   ↵
262                                             textExtent.cy / 2, character, character.GetLength   ↵
263                                             ());
264                                         }
265                                         }
266                                         }

```

```
249         }
250
251         else if (character == L"\\" || character == L"\") {
252
253             pDc->SetTextColor(COLORREF(RGB(255, 128, 0)));
254
255             int j = i + 1;
256             BOOL found = FALSE;
257
258             while (!found && j < this->text.GetLength()) {
259
260                 if (this->text.Mid(j, 1) == character) {
261
262                     word = this->text.Mid(i, j - i + 1);
263
264                     if (printing) {
265
266                         // Add the length if the word to the
267                         // length of our print
268                         lengthPrint += word.GetLength();
269
270                         while (lengthPrint >
271                             printMaxCharLength) { // As a word could
272                             technically have a length greater than 50, we must
273                             use a while to check for lines
274
275                             CString tempWord = L"";
276                             if ((i - word.GetLength()) %
277                                 printMaxCharLength != 0) {
278
279                             tempWord = word.Mid(0,
280                                     printMaxCharLength - ((i - word.GetLength()) %
281                                         printMaxCharLength)); // Create a temporary word to
282                                         store the start of the word that will go on the
283                                         end of the current line
284
285                             word = word.Mid
286                             (printMaxCharLength - ((i - word.GetLength()) %
287                                         printMaxCharLength)); // Change the word to be
288                                         whatever we didn't print and continue the loop
289
290                         }
291                         else {
292
293                             pos++;
294                         }
295
296                         pDc->TextOut(this->bounds.left +
297                             2 + (textExtent.cx * pos), this->bounds.top +
298                             this->bounds.Height() / 2 - textExtent.cy / 2 +
299                             returnNewLines * this->bounds.Height(), tempWord,
300                             tempWord.GetLength()); // Print what we can to fill
301                             the line
302
303                         lengthPrint -=
304                             printMaxCharLength; // Since we are creating a new
305                             line, subtract the old 60 characters from this
306
307                         /*pos += tempWord.GetLength() + */
308                         2;
309
310             }
```

```
285
286             Gdiplus::Graphics g(pDc-
287             >GetSafeHdc());
288             Gdiplus::Rect expansionRect
289             (this->bounds.left + 2 + (textExtent.cx * pos),
290             this->bounds.top + this->bounds.Height() / 2 +
291             returnNewLines * this->bounds.Height(),
292             textExtent.cx,
293             textExtent.cx);
294
295             g.DrawImage(&newLineArrow,
296             expansionRect);*/
297
298             returnNewLines++;
299             pos = 0;
300         }
301     }
302
303     // Print the 'scraps' of the word at the
304     // end on the next line
305     pDc->TextOut(this->bounds.left + 2 +
306     (textExtent.cx * pos), this->bounds.top + this-
307     >bounds.Height() / 2 - textExtent.cy / 2 +
308     returnNewLines * this->bounds.Height(), word,
309     word.GetLength());
310
311     pos += word.GetLength();
312     found = TRUE;
313
314     j++;
315
316     if (!found) {
317         word = this->text.Mid(i);
318         pDc->TextOut(this->bounds.left + 2 +
319         (textExtent.cx * pos), this->bounds.top + this-
320         >bounds.Height() / 2 - textExtent.cy / 2 +
321         returnNewLines * this->bounds.Height(), word,
322         word.GetLength());
323     }
324
325     pDc->SetTextColor(RGB(0, 0, 0)); // Revert
326     colour back to black (original colour)
327     i += word.GetLength() - 1;
328
329     word = L""; // Reset the word
330
331     pos = i + 1;
332
333     // If character is not a break away character, or we
334     // are not at the end of the loop
335     else {
```

```

...urce\repos\DesignArk\DesignArk\CTextLineObject.cpp 9
324             // Append the character to the word we are to ↵
325             look at
326             word += character;
327         }
328     }
329
330     // This means that there is a comment
331     else {
332
333         // Set colour of the comment
334         pDc->SetTextColor(theApp.commentColour);
335
336         // Add the rest of the line to the word for drawing
337         word.Append(this->text.Mid(i));
338
339         // If we are printing...
340         if (printing) {
341
342             // Add the length of the word to the length of ↵
343             // our print
344             lengthPrint += word.GetLength();
345
346             while (lengthPrint > printMaxCharLength) { // As ↵
347                 a word could technically have a length greater ↵
348                 than 50, we must use a while to check for lines
349
350                 CString tempWord = L"";
351                 if ((i - word.GetLength()) % ↵
352                     printMaxCharLength != 0) {
353
354                     tempWord = word.Mid(0, ↵
355                     printMaxCharLength - pos); // Create a temporary ↵
356                     word to store the start of the word that will go on ↵
357                     the end of the current line
358                     word = word.Mid(printMaxCharLength - ↵
359                     pos); // Change the word to be whatever we didn't ↵
360                     print and continue the loop
361
362                     }
363                 else {
364                     pos++;
365                 }
366
367                 pDc->TextOut(this->bounds.left + 2 + ↵
368                 (textExtent.cx * pos), this->bounds.top + this- ↵
369                 >bounds.Height() / 2 - textExtent.cy / 2 + ↵
370                 returnNewLines * this->bounds.Height(), tempWord, ↵
371                 tempWord.GetLength()); // Print what we can to fill ↵
372                 the line
373
374                 lengthPrint -= printMaxCharLength; // Since ↵
375                 we are creating a new line, subtract the old 60 ↵
376                 characters from this
377
378                 /*pos += tempWord.GetLength() + 2;
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
779
780
781
782
783
784
785
786
787
788
789
789
790
791
792
793
794
795
796
797
798
799
799
800
801
802
803
804
805
806
807
808
809
809
810
811
812
813
814
815
816
817
818
819
819
820
821
822
823
824
825
826
827
828
829
829
830
831
832
833
834
835
836
837
838
839
839
840
841
842
843
844
845
846
847
848
849
849
850
851
852
853
854
855
856
857
858
859
859
860
861
862
863
864
865
866
867
868
869
869
870
871
872
873
874
875
876
877
878
879
879
880
881
882
883
884
885
886
887
888
889
889
890
891
892
893
894
895
896
897
898
899
899
900

```

...urce\repos\DesignArk\DesignArk\CTextLineObject.cpp

---

```

363                     Gdiplus::Graphics g(pDc->GetSafeHdc());
364                     Gdiplus::Rect expansionRect(this-
365 >bounds.left + 2 + (textExtent.cx * pos), this-
366 >bounds.top + this->bounds.Height() / 2 +
367 returnNewLines * this->bounds.Height(),
368                     textExtent.cx, textExtent.cy);

369                     g.DrawImage(&newLineArrow, expansionRect); */

370                     returnNewLines++;
371                     pos = 0;
372                 }
373                 // Print the 'scraps' of the word at the end on the next line
374                 pDc->TextOut(this->bounds.left + 2 +
375 (textExtent.cx * pos), this->bounds.top + this-
376 >bounds.Height() / 2 - textExtent.cy / 2 +
377 returnNewLines * this->bounds.Height(), word,
378 word.GetLength());

379             }
380             // Not printing...
381             else {
382                 pDc->TextOut(this->bounds.left + 2 +
383 (textExtent.cx * pos), this->bounds.top + this-
384 >bounds.Height() / 2 - textExtent.cy / 2, word,
385 word.GetLength());
386             }
387             i++;
388         }
389     }
390     // If we are not using SmartColour
391     else {
392
393         if (printing) {
394             // Add the length if the word to the length of our print
395             CString word = this->text;
396             lengthPrint += word.GetLength();
397
398             while (lengthPrint > printMaxCharLength) { // As a word could technically have a length greater than 60, we must use a while to check for lines
399
400                 CString tempWord = word.Mid(0,
401 printMaxCharLength); // Create a temporary word to store the start of the word that will go on the end of the current line
402                 pDc->TextOut(this->bounds.left + 2, this->bounds.top +
403 + this->bounds.Height() / 2 - textExtent.cy / 2 +
```

```

...urce\repos\DesignArk\DesignArk\CTextLineObject.cpp 11
        returnNewLines * this->bounds.Height(), tempWord, ↵
        tempWord.GetLength()); // Print what we can to fill ↵
        the line
402
403     /*int pos = tempWord.GetLength() + 2;
404
405     Gdiplus::Graphics g(pDc->GetSafeHdc());
406     Gdiplus::Rect expansionRect(this->bounds.left + 2 + ↵
407         (textExtent.cx * pos), this->bounds.top + this- ↵
408         >bounds.Height() / 2 + returnNewLines * this- ↵
409         >bounds.Height(),
410         textExtent.cx, textExtent.cy);
411
412     g.DrawImage(&newLineArrow, expansionRect);*/
413
414     word = word.Mid(printMaxCharLength); // Change the ↵
415     word to be whatever we didn't print and continue ↵
416     the loop
417     lengthPrint -= printMaxCharLength; // Since we are ↵
418     creating a new line, subtract the old 60 characters ↵
419     from this
420     returnNewLines++;
421 }
422 // Print the 'scraps' of the word at the end on the next ↵
423 // line
424 pDc->TextOut(this->bounds.left + 2, this->bounds.top + ↵
425     this->bounds.Height() / 2 - textExtent.cy / 2 + ↵
426     returnNewLines * this->bounds.Height(), word, ↵
427     word.GetLength());
428 }
429 else {
430     pDc->TextOut(this->bounds.left + 2, this->bounds.top + ↵
431         this->bounds.Height() / 2 - textExtent.cy / 2, this- ↵
432         >text, this->text.GetLength());
433 }
434
435 if (this->active && !this->highlight && !printing) {
436
437     pDc->SelectStockObject(HOLLOW_BRUSH);
438     pDc->Rectangle(bounds);
439 }
440
441 if (this->highlight && !printing) {
442
443     this->highlighter.X = hStart * textExtent.cx + this- ↵
444         >bounds.left + 2;
445     this->highlighter.Width = (hEnd * textExtent.cx + this- ↵
446         >bounds.left + 2) - this->highlighter.X;
447
448     Gdiplus::Graphics g(pDc->GetSafeHdc());
449
450     Gdiplus::SolidBrush solidBrush(Gdiplus::Color(100, GetRValue ↵
451         (theApp.highlightColour), GetGValue ↵
452         (theApp.highlightColour), GetBValue ↵
453         (theApp.highlightColour)));

```

```

...urce\repos\DesignArk\DesignArk\CTextLineObject.cpp
437         g.FillRectangle(&solidBrush, this->highlighter);
438     }
439     return returnNewLines;
440 }
441
442 // Public getters & setters
443 CString CTextLineObject::getText()
444 {
445     return this->text;
446 }
447 int CTextLineObject::getLength()
448 {
449     return this->text.GetLength();
450 }
451 void CTextLineObject::setText(CString text)
452 {
453     this->text = text;
454 }
455 void CTextLineObject::concatenateString(CString text, int position)
456 {
457     if (!GetKeyState(VK_INSERT)) {
458         this->text = this->text.Mid(0, position) + text + this->text.Mid(position, this->text.GetLength());
459     }
460     else {
461         this->text = this->text.Mid(0, position) + text + this->text.Mid(position + 1, this->text.GetLength());
462     }
463 }
464 void CTextLineObject::backspace(int position)
465 {
466     if (!this->highlight) {
467         this->text = this->text.Mid(0, position - 1) + this->text.Mid(position, this->text.GetLength());
468     }
469     else {
470         this->text = this->text.Mid(0, min(this->hStart, this->hEnd)) + this->text.Mid(max(this->hStart, this->hEnd), this->text.GetLength());
471         this->setHighlighting(FALSE);
472     }
473 }
474
475 BOOL CTextLineObject::isHighlighting()
476 {
477     return this->highlight;
478 }
479 void CTextLineObject::setHighlighting(BOOL val)
480 {
481     if (!val) {
482         this->highlighter.X = 0;
483         this->highlighter.Width = 0;
484         this->lineHighlight = FALSE;
485     }
486     this->highlight = val;
487 }

```

---

...urce\repos\DesignArk\DesignArk\CTextLineObject.cpp

```

488 void CTextLineObject::setHighlighter(int pos1, int pos2)
489 {
490     if (pos1 > this->text.GetLength()) {
491         pos1 = this->text.GetLength() + 1;
492     }
493     if (pos2 > this->text.GetLength()) {
494         pos2 = this->text.GetLength() + 1;
495     }
496
497     if ((pos1 == 0 && pos2 == this->text.GetLength() + 1) || (pos2 == 0 && pos1 == this->text.GetLength() + 1)) {
498         this->lineHighlight = TRUE;
499     }
500
501     this->hStart = min(pos1, pos2);
502     this->hEnd = max(pos1, pos2);
503
504     if (pos1 != pos2) {
505         this->highlight = TRUE;
506     }
507 }
508 void CTextLineObject::highlightLine()
509 {
510     hStart = 0;
511     hEnd = this->text.GetLength() + 1;
512     this->highlight = TRUE;
513     this->lineHighlight = TRUE;
514 }
515
516 BOOL CTextLineObject::isLineHighlighted()
517 {
518     return this->lineHighlight;
519 }
520
521 int CTextLineObject::getHStart()
522 {
523     return this->hStart;
524 }
525
526 int CTextLineObject::getHEnd()
527 {
528     return this->hEnd;
529 }
530
531 CString CTextLineObject::getHighlightedText()
532 {
533     if (this->highlight) {
534         return this->text.Mid(this->hStart, this->hEnd - this->hStart);
535     }
536     return CString();
537 }
538
539 int CTextLineObject::getNumSubLines()
540 {
541     return this->numSubLines;

```

```

...urce\repos\DesignArk\DesignArk\CTextLineObject.cpp
542 }
543 std::vector<int> CTextLineObject::iGetLineNums()
544 {
545     return this->lineNums;
546 }
547 CString CTextLineObject::sGetLineNums()
548 {
549     CString num;
550     for (auto& it : this->lineNums) {
551
552         CString concat;
553         concat.Format(L"%d", it);
554         num.Append(concat);
555         num.Append(L".");
556     }
557     return num.Mid(0, num.GetLength() - 1);
558 }
559 void CTextLineObject::addSublines(std::vector<int> subs)
560 {
561     this->numSubLines += subs.size();
562     for (auto& it : subs) {
563         this->lineNums.push_back(it);
564     }
565 }
566
567 void CTextLineObject::incrementLine(int subline, int val)
568 {
569     if (subline > this->lineNums.size()) {
570         throw("ERROR::subline::OUT OF RANGE");
571     }
572     this->lineNums[subline] += val;
573 }
574
575 void CTextLineObject::setBounds(CRect bounds)
576 {
577     CAppObject::setBounds(bounds);
578
579     this->highlighter.X = bounds.left;
580     this->highlighter.Y = bounds.top;
581     this->highlighter.Height = bounds.Height();
582 }
583
584 void CTextLineObject::move(int x, int y)
585 {
586     CAppObject::move(x, y);
587     this->highlighter.Offset(x, y);
588 }
589
590 BOOL CTextLineObject::onlyNums(CString str)
591 {
592     for (int i = 0; i < str.GetLength(); i++) {
593
594         if (str.Mid(i, 1) != L"0" &&
595             str.Mid(i, 1) != L"1" &&
596             str.Mid(i, 1) != L"2" &&
597             str.Mid(i, 1) != L"3" &&

```

```
598         str.Mid(i, 1) != L"4" &&
599         str.Mid(i, 1) != L"5" &&
600         str.Mid(i, 1) != L"6" &&
601         str.Mid(i, 1) != L"7" &&
602         str.Mid(i, 1) != L"8" &&
603         str.Mid(i, 1) != L"9" ) {
604     return FALSE;
605 }
606 }
607
608     return TRUE;
609 }
610 BOOL CTextLineObject::brkPosContains(std::vector<std::vector<int>> ▷
611     vector, int value, int side)
612 {
613     for (int i = 0; i < vector.size(); i++) {
614         if (vector[i][side] == value) {
615             return TRUE;
616         }
617     }
618
619     return FALSE;
620 }
621
```

```
1 #pragma once
2
3 #include "CSmartColourStatements.h"
4 #include "CSmartColourTypes.h"
5 #include "CSmartColourBuiltin.h"
6 #include "CSmartColourOther.h"
7
8 // CSsmartColour
9
10 class CSmartColour : public CPropertySheet
11 {
12
13 public:
14     CSmartColour(std::vector<std::vector<CString>> page_strings,
15                  std::vector<COLORREF> page_colour,
16                  CString commentType, COLORREF commentColour,
17                  COLORREF numberColour, COLORREF highlightColour,
18                  CString fonts, CString indents,
19                  LPCTSTR pszCaption = L"",
20                  CWnd* pParentWnd =
21                  nullptr, UINT iSelectPage = 0);
22
23     virtual ~CSmartColour();
24
25 protected:
26     CSmartColourTypes types;
27     CSmartColourStatements statements;
28     CSmartColourBuiltin builtin;
29     CSmartColourOther other;
30
31     DECLARE_DYNAMIC(CSmartColour)
32     DECLARE_MESSAGE_MAP()
33 };
```

```
1 // CSmartColour.cpp : implementation file
2 //
3
4 #include "pch.h"
5 #include "DesignArk.h"
6
7 #include "CSmartColour.h"
8
9 // CSmartColour
10
11 IMPLEMENT_DYNAMIC(CSmartColour, CPropertySheet)
12
13 CSmartColour::CSmartColour(std::vector<std::vector<CString>>
14                             page_strings, std::vector<COLORREF> page_colour,
15                             CString commentType, COLORREF
16                             commentColour, COLORREF numberColour, COLORREF
17                             highlightColour, CString fonts, CString indents,
18                             LPCTSTR pszCaption, CWnd* pParentWnd,
19                             UINT iSelectPage)
20 :CPropertySheet(pszCaption, pParentWnd, iSelectPage),
21 statements(page_strings[0], page_colour[0], page_strings[1],
22             page_colour[1]),
23 types(page_strings[2], page_colour[2]),
24 builtin(page_strings[3], page_colour[3], page_strings[4],
25         page_colour[4]),
26 other(commentType, commentColour, numberColour, highlightColour,
27       fonts, indents)
28 {
29     this->AddPage(&this->statements);
30     this->AddPage(&this->types);
31     this->AddPage(&this->builtin);
32     this->AddPage(&this->other);
33 }
34
35
36
37 // CSmartColour message handlers
38
```

```
1 #pragma once
2 #include "afxdialogex.h"
3
4
5 // CSmartColourOps dialog
6
7 class CSmartColourStatements : public CMFCPropertyPage
8 {
9
10 public:
11     CSmartColourStatements(std::vector<CString> syntax_string,
12                           COLORREF syntax_colour, std::vector<CString> statement_string,
13                           COLORREF statement_colour, CWnd* pParent = nullptr); // standard constructor
14     virtual ~CSmartColourStatements();
15
16 // Dialog Data
17 #ifdef AFX DESIGN TIME
18     enum { IDD = IDD_SMARTCOLOUR_OPS };
19#endif
20
21 protected:
22     virtual void DoDataExchange(CDataExchange* pDX); // DDX/DDV
23         support
24     virtual BOOL OnInitDialog();
25     virtual BOOL OnKillActive();
26
27     DECLARE_MESSAGE_MAP()
28     DECLARE_DYNAMIC(CSmartColourStatements)
29
30 public:
31     // Controls
32     CVSLListBox m_Listbox_Statements_syntax_control;
33     CMFCCColorButton m_Combo_Statements_syntax_control;
34
35     CVSLListBox m_Listbox_Statements_statements_control;
36     CMFCCColorButton m_Combo_Statements_statements_control;
37
38     // Variables
39     std::vector<CString> m.Strings_Statements_syntax_var;
40     COLORREF m.Colour_Statements_syntax_var;
41
42     std::vector<CString> m.Strings_Statements_statements_var;
43     COLORREF m.Colour_Statements_statements_var;
44
45 }
```

```
1 // CSmartColourOps.cpp : implementation file
2 //
3
4 #include "pch.h"
5 #include "DesignArk.h"
6 #include "afxdialogex.h"
7 #include "CSmartColourStatements.h"
8
9
10 // CSmartColourOps dialog
11
12 IMPLEMENT_DYNAMIC(CSmartColourStatements, CMFCPropertyPage)
13
14 CSmartColourStatements::CSmartColourStatements(std::vector<CString>
15     syntax_string, COLORREF syntax_colour, std::vector<CString>
16     statement_string, COLORREF statement_colour, CWnd* pParent/* =
17     nullptr*/)
18     : CMFCPropertyPage(IDD_SMARTCOLOUR_OPS /*, pParent*/
19     , m_Listbox_Statements_syntax_control()
20     , m_Listbox_Statements_statements_control())
21 {
22     this->m.Strings.Statements.syntax_var = syntax_string;
23     this->m.Colour.Statements.syntax_var = syntax_colour;
24 }
25
26 CSmartColourStatements::~CSmartColourStatements()
27 {
28 }
29
30 void CSmartColourStatements::DoDataExchange(CDataExchange* pDX)
31 {
32     CMFCPropertyPage::DoDataExchange(pDX);
33     DDX_Control(pDX, IDC_MFCVSLISTBOX_OPS,
34         m_Listbox_Statements_statements_control);
35     DDX_Control(pDX, IDC_COMBO_COLOUR_OPS,
36         m_Combo_Statements_statements_control);
37     DDX_Control(pDX, IDC_MFCVSLISTBOX_OPS2,
38         m_Listbox_Statements_syntax_control);
39     DDX_Control(pDX, IDC_COMBO_COLOUR_OPS2,
40         m_Combo_Statements_syntax_control);
41 }
42
43
44     for (auto& it : this->m.Strings.Statements.syntax_var) {
45         this->m.Listbox_Statements_syntax_control.AddItem(it);
46     }
47
48     this->m.Combo_Statements_syntax_control.SetColor(this-
49         >m.Colour.Statements.syntax_var);
```

```
49
50     for (auto& it : this->m_Strings_Statements_statements_var) {
51         this->m_Listbox_Statements_statements_control.AddItem(it);
52     }
53
54     this->m_Combos_Statements_statements_control.SetColor(this-
55         >m_Colour_Statements_statements_var);                    ↵
56
57     return 0;
58 }
59 BOOL CSmartColourStatements::OnKillActive()
60 {
61     // Syntax
62     this->m_Strings_Statements_syntax_var.resize(this-
63         >m_Listbox_Statements_syntax_control.GetCount());          ↵
64
65     for (int i = 0; i < this-
66         >m_Listbox_Statements_syntax_control.GetCount(); i++) {    ↵
67         this->m_Strings_Statements_syntax_var[i] = this-
68             >m_Listbox_Statements_syntax_control.GetItemText(i);    ↵
69     }
70
71     // Insertion Sort
72     for (int i = 1; i < this->m_Strings_Statements_syntax_var.size()    ↵
73         (); i++) {
74         CString currentValue = this->m_Strings_Statements_syntax_var[    ↵
75             i];
76         int pos = i;
77
78         while (pos > 0 && this->m_Strings_Statements_syntax_var[pos - 1]    ↵
79             > currentValue) {
80             this->m_Strings_Statements_syntax_var[pos] = this-
81                 >m_Combos_Statements_syntax_control.GetColor();          ↵
82
83             // Statements
84             this->m_Strings_Statements_statements_var.resize(this-
85                 >m_Listbox_Statements_statements_control.GetCount());    ↵
86
87             for (int i = 0; i < this-
88                 >m_Listbox_Statements_statements_control.GetCount(); i++) {    ↵
89                 this->m_Strings_Statements_statements_var[i] = this-
90                     >m_Listbox_Statements_statements_control.GetItemText(i);    ↵
91             }
92
93             // Insertion Sort
94             for (int i = 1; i < this-
95                 >m_Strings_Statements_statements_var.size(); i++) {
96                 CString currentValue = this-
```

```
92         >m_Strings_Statements_statements_var[i];
93         int pos = i;
94
95         while (pos > 0 && this->m_Strings_Statements_statements_var [pos - 1] > currentValue) {
96             this->m_Strings_Statements_statements_var[pos] = this-
97                 >m_Strings_Statements_statements_var[pos - 1];
98             pos--;
99         }
100
101        this->m_Colour_Statements_statements_var = this-
102            >m_Combo_Statements_statements_control.GetColor();
103
104    }
105
106
107
108 BEGIN_MESSAGE_MAP(CSmartColourStatements, CMFCPropertyPage)
109 END_MESSAGE_MAP()
110
111
112 // CSmartColourOps message handlers
113
```

```
1 #pragma once
2 #include "afxdialogex.h"
3
4
5 // CSmartColourTypes dialog
6
7 class CSmartColourTypes : public CMFCPropertyPage
8 {
9
10 public:
11     CSmartColourTypes(std::vector<CString> types_string, COLORREF
12                         colour, CWnd* pParent = nullptr);    // standard constructor
13     virtual ~CSmartColourTypes();
14
15 // Dialog Data
16 #ifdef AFX DESIGN TIME
17     enum { IDD = IDD_SMARTCOLOUR_TYPES };
18 #endif#
19
20 protected:
21     virtual void DoDataExchange(CDataExchange* pDX);      // DDX/DDV
22     virtual support
23     virtual BOOL OnInitDialog();
24     virtual BOOL OnKillActive();
25
26     DECLARE_DYNAMIC(CSmartColourTypes)
27     DECLARE_MESSAGE_MAP()
28
29 public:
30     // Controls
31     CVSListBox m_Listbox_Types_control;
32     CMFCColorButton m_Combo_Types_colour_control;
33
34     // Variables
35     COLORREF m_Combo_Types_colour_var;
36     std::vector<CString> m_listbox_types_strings;
37 }
```

```
1 // CSmartColourTypes.cpp : implementation file
2 //
3
4 #include "pch.h"
5 #include "DesignArk.h"
6 #include "afxdialogex.h"
7 #include "CSmartColourTypes.h"
8
9
10 // CSmartColourTypes dialog
11
12 IMPLEMENT_DYNAMIC(CSmartColourTypes, CMFCPropertyPage)
13
14 CSmartColourTypes::CSmartColourTypes(std::vector<CString>
15     types_string, COLORREF colour, CWnd* pParent /*=nullptr*/)
16     : CMFCPropertyPage(IDD_SMARTCOLOUR_TYPES, pParent)
17     , m_Listbox_Types_control()
18 {
19     this->m_listbox_types_strings = types_string;
20     this->m_Combo_Types_colour_var = colour;
21 }
22 CSmartColourTypes::~CSmartColourTypes()
23 {
24 }
25 void CSmartColourTypes::DoDataExchange(CDataExchange* pDX)
26 {
27     CMFCPropertyPage::DoDataExchange(pDX);
28
29     DDX_Control(pDX, IDC_MFCVSLISTBOX_TYPES,
30         m_Listbox_Types_control);
31     DDX_Control(pDX, IDC_COMBO_COLOUR_TYPES,
32         m_Combo_Types_colour_control);
33 }
34
35 BOOL CSmartColourTypes::OnInitDialog()
36 {
37     CDialog::OnInitDialog();
38
39     for (auto& it : this->m_listbox_types_strings) {
40         this->m_Listbox_Types_control.AddItem(it);
41     }
42     this->m_Combo_Types_colour_control.SetColor(this-
43         >m_Combo_Types_colour_var);
44
45     return 0;
46 }
47
48 BOOL CSmartColourTypes::OnKillActive()
49 {
50     this->m_listbox_types_strings.resize(this-
51         >m_Listbox_Types_control.GetCount());
52
53     for (int i = 0; i < this->m_Listbox_Types_control.GetCount(); i++)
54     {
55         this->m_listbox_types_strings[i] = this-
```

```
...ce\repos\DesignArk\DesignArk\CSmartColourTypes.cpp 2
    >m_Listbox_Types_control.GetItemText(i);
51 }
52
53 // Insertion Sort
54 for (int i = 1; i < this->m_listbox_types_strings.size(); i++) {
55
56     CString currentValue = this->m_listbox_types_strings[i];
57     int pos = i;
58
59     while (pos > 0 && this->m_listbox_types_strings[pos - 1] > currentValue) {
60         this->m_listbox_types_strings[pos] = this->m_listbox_types_strings[pos - 1];  ↵
61         pos--;
62     }
63     this->m_listbox_types_strings[pos] = currentValue;
64 }
65
66 this->m_Combos_Colour_var = this->m_Combos_Colour_control.GetColor();  ↵
67
68 return CMFCPropertyPage::OnKillActive();
69 }
70
71
72 BEGIN_MESSAGE_MAP(CSmartColourTypes, CMFCPropertyPage)
73 END_MESSAGE_MAP()
74
```

```
1 #pragma once
2 #include "afxdialogex.h"
3
4
5 // CSmartColourBuiltIn dialog
6
7 class CSmartColourBuiltIn : public CMFCPropertyPage
8 {
9
10 public:
11     CSmartColourBuiltIn(std::vector<CString> classes_string, COLORREF classes_colour,
12                         std::vector<CString> functions_string, COLORREF functions_colour, CWnd* pParent = nullptr); // standard constructor
13     virtual ~CSmartColourBuiltIn();
14
15 // Dialog Data
16 #ifdef AFX DESIGN TIME
17     enum { IDD = IDD_SMARTCOLOUR_BUILTIN };
18#endif
19
20 protected:
21     virtual void DoDataExchange(CDataExchange* pDX); // DDX/DDV
22     support
23     virtual BOOL OnInitDialog();
24     virtual BOOL OnKillActive();
25
26     DECLARE_DYNAMIC(CSmartColourBuiltIn)
27     DECLARE_MESSAGE_MAP()
28
29 public:
30
31     // Controls
32     CVSListBox m_Listbox_Builtin_Classes_control;
33     CMFCColorButton m_Combo_Builtin_Classes_colour_control;
34
35     // Variables
36     std::vector<CString> m_Listbox_Builtin_Classes_var;
37     COLORREF m_Combo_Builtin_Classes_colour_var;
38
39     std::vector<CString> m_Listbox_Builtin_Functions_var;
40     COLORREF m_Combo_Builtin_Functions_colour_var;
41 };
42
```

```
1 // CSmartColourBuiltIn.cpp : implementation file
2 //
3
4 #include "pch.h"
5 #include "DesignArk.h"
6 #include "afxdialogex.h"
7 #include "CSmartColourBuiltIn.h"
8
9
10 // CSmartColourBuiltIn dialog
11
12 IMPLEMENT_DYNAMIC(CSmartColourBuiltIn, CMFCPropertyPage)
13
14 CSmartColourBuiltIn::CSmartColourBuiltIn(std::vector<CString>
15     classes_string, COLORREF classes_colour, std::vector<CString>
16     functions_string, COLORREF functions_colour, CWnd* pParent /
17     *=nullptr*)
18     : CMFCPropertyPage(IDD_SMARTCOLOUR_BUILTIN/*, pParent*/)
19 {
20     this->m_Listbox_Builtin_Classes_var = classes_string;
21     this->m_Combo_Builtin_Classes_colour_var = classes_colour;
22 }
23
24 CSmartColourBuiltIn::~CSmartColourBuiltIn()
25 {
26 }
27
28 void CSmartColourBuiltIn::DoDataExchange(CDataExchange* pDX)
29 {
30     CMFCPropertyPage::DoDataExchange(pDX);
31     DDX_Control(pDX, IDC_MFCVSLISTBOX_BUILTIN_CLASSES,
32         m_Listbox_Builtin_Classes_control);
33     DDX_Control(pDX, IDC_COMBO_COLOUR_BUILTIN_CLASSES,
34         m_Combo_Builtin_Classes_colour_control);
35     DDX_Control(pDX, IDC_MFCVSLISTBOX_BUILTIN_FUNCTIONS,
36         m_Listbox_Builtin_Functions_control);
37     DDX_Control(pDX, IDC_COMBO_COLOUR_BUILTIN_FUNCTIONS,
38         m_Combo_Builtin_Functions_colour_control);
39 }
40
41 BOOL CSmartColourBuiltIn::OnInitDialog()
42 {
43     CDialog::OnInitDialog();
44
45     for (auto& it : this->m_Listbox_Builtin_Classes_var) {
46         this->m_Listbox_Builtin_Classes_control.AddItem(it);
47     }
48     this->m_Combo_Builtin_Classes_colour_control.SetColor(this-
49         >m_Combo_Builtin_Classes_colour_var);
50
51     for (auto& it : this->m_Listbox_Builtin_Functions_var) {
52         this->m_Listbox_Builtin_Functions_control.AddItem(it);
53     }
54 }
```

```

...\\repos\\DesignArk\\DesignArk\\CSmartColourBuiltin.cpp 2
49     this->m_Combo_Builtin_Functions_colour_control.SetColor(this-
50         >m_Combo_Builtin_Functions_colour_var);
51
52     return 0;
53 }
54 BOOL CSmartColourBuiltin::OnKillActive()
55 {
56     // Classes
57     this->m_Listbox_Builtin_Classes_var.resize(this-
58         >m_Listbox_Builtin_Classes_control.GetCount());
59
60     for (int i = 0; i < this-
61         >m_Listbox_Builtin_Classes_control.GetCount(); i++) {
62         this->m_Listbox_Builtin_Classes_var[i] = this-
63             >m_Listbox_Builtin_Classes_control.GetItemText(i);
64     }
65
66     // Insertion Sort
67     for (int i = 1; i < this->m_Listbox_Builtin_Classes_var.size(); i++)
68     {
69         CString currentValue = this->m_Listbox_Builtin_Classes_var
70             [i];
71         int pos = i;
72
73         while (pos > 0 && this->m_Listbox_Builtin_Classes_var[pos -
74             1] > currentValue) {
75             this->m_Listbox_Builtin_Classes_var[pos] = this-
76                 >m_Listbox_Builtin_Classes_var[pos - 1];
77             pos--;
78         }
79         this->m_Listbox_Builtin_Classes_var[pos] = currentValue;
80     }
81
82     this->m_Combo_Builtin_Classes_colour_var = this-
83         >m_Combo_Builtin_Classes_colour_control.GetColor();
84
85     //Functions
86     this->m_Listbox_Builtin_Functions_var.resize(this-
87         >m_Listbox_Builtin_Functions_control.GetCount());
88
89     for (int i = 0; i < this-
90         >m_Listbox_Builtin_Functions_control.GetCount(); i++) {
91         this->m_Listbox_Builtin_Functions_var[i] = this-
92             >m_Listbox_Builtin_Functions_control.GetItemText(i);
93     }
94
95     // Insertion Sort
96     for (int i = 1; i < this->m_Listbox_Builtin_Functions_var.size
97         (); i++) {
98         CString currentValue = this->m_Listbox_Builtin_Functions_var
99             [i];
100        int pos = i;
101
102        while (pos > 0 && this->m_Listbox_Builtin_Functions_var[pos -
103            1] > currentValue) {

```

...\\repos\\DesignArk\\DesignArk\\CSmartColourBuiltIn.cpp 3

---

```
90         this->m_Listbox_Builtin_Functions_var[pos] = this-
91             >m_Listbox_Builtin_Functions_var[pos - 1];
92         pos--;
93         this->m_Listbox_Builtin_Functions_var[pos] = currentValue;
94     }
95
96     this->m_Combo_Builtin_Functions_colour_var = this-
97         >m_Combo_Builtin_Functions_colour_control.GetColor();
98
99     return CMFCPropertyPage::OnKillActive();
100 }
101
102
103 BEGIN_MESSAGE_MAP(CSmartColourBuiltIn, CMFCPropertyPage)
104 END_MESSAGE_MAP()
105
106
107 // CSmartColourBuiltIn message handlers
108
```

```
1 #pragma once
2 #include "afxdialogex.h"
3
4
5 // CSmartColourOther dialog
6
7 class CSmartColourOther : public CMFCPropertyPage
8 {
9
10 public:
11     CSmartColourOther(CString comments_type, COLORREF
12                         comments_colour, COLORREF numbers_colour, COLORREF
13                         highlight_colour, CString fonts, CString indentSize, CWnd*
14                         pParent = nullptr); // standard constructor
15     virtual ~CSmartColourOther();
16
17 // Dialog Data
18 #ifdef AFX DESIGN TIME
19     enum { IDD = IDD_SMARTCOLOUR_OTHER };
20 #endif
21
22 protected:
23     virtual void DoDataExchange(CDataExchange* pDX); // DDX/DDV
24     support
25     virtual BOOL OnInitDialog();
26     virtual BOOL OnKillActive();
27
28     DECLARE_DYNAMIC(CSmartColourOther)
29     DECLARE_MESSAGE_MAP()
30
31 public:
32
33     // Controls
34     CComboBox m_pCombo_comments_type_control;
35     CMFCColorButton m_pCombo_comments_colour_control;
36     CMFCColorButton m_pCombo_numbers_colour_control;
37     CMFCColorButton m_pCombo_highlight_colour_control;
38     CComboBox m_pCombo_fonts_control;
39     CEdit m_pCombo_indent_control;
40
41     // Variables
42     CString m_pCombo_comments_type_var;
43     COLORREF m_pCombo_comments_colour_var;
44     COLORREF m_pCombo_numbers_colour_var;
45     COLORREF m_pCombo_highlight_colour_var;
46     CString m_pCombo_fonts_var;
47     CString m_pCombo_indent_var;
48 };
49
```

```
1 // CSmartColourOther.cpp : implementation file
2 //
3
4 #include "pch.h"
5 #include "DesignArk.h"
6 #include "afxdialogex.h"
7 #include "CSmartColourOther.h"
8
9
10 // CSmartColourOther dialog
11
12 IMPLEMENT_DYNAMIC(CSmartColourOther, CMFCPropertyPage)
13
14 CSmartColourOther::CSmartColourOther(CString comments_type, COLORREF comments_colour, COLORREF numbers_colour, COLORREF highlight_colour, CString fonts, CString indentSize, CWnd* pParent /*=nullptr*/)
15     : CMFCPropertyPage(IDD_SMARTCOLOUR_OTHER/*, pParent*/)
16 {
17     this->m_pCombo_comments_type_var = comments_type;
18     this->m_pCombo_comments_colour_var = comments_colour;
19     this->m_pCombo_numbers_colour_var = numbers_colour;
20     this->m_pCombo_highlight_colour_var = highlight_colour;
21     this->m_pCombo_fonts_var = fonts;
22     this->m_pCombo_indent_var = indentSize;
23 }
24
25 CSmartColourOther::~CSmartColourOther()
26 {
27 }
28
29 void CSmartColourOther::DoDataExchange(CDataExchange* pDX)
30 {
31     CMFCPropertyPage::DoDataExchange(pDX);
32     DDX_Control(pDX, IDC_SMARTCOLOUR_OTHERS_COMMENT_COMBO,
33                 m_pCombo_comments_type_control);
33     DDX_Control(pDX, IDC_SMARTCOLOUR_OTHER_COMMENTS_COLOUR,
34                 m_pCombo_comments_colour_control);
34     DDX_Control(pDX, IDC_SMARTCOLOUR_OTHERS_NUMBERS_COLOUR,
35                 m_pCombo_numbers_colour_control);
35     DDX_Control(pDX, IDC_SMARTCOLOUR_OTHERS_NUMBERS_COLOUR2,
36                 m_pCombo_highlight_colour_control);
36     DDX_Control(pDX, IDC_SMARTCOLOUR_OTHERS_FONT,
37                 m_pCombo_fonts_control);
37     DDX_Control(pDX, IDC_SMARTCOLOR_OTHER_INDENT,
38                 m_pCombo_indent_control);
38 }
39 BOOL CSmartColourOther::OnInitDialog()
40 {
41     CDialog::OnInitDialog();
42
43     this->m_pCombo_comments_type_control.SetCurSel(this-
44             >m_pCombo_comments_type_control.FindString(-1, this-
45             >m_pCombo_comments_type_var));
44     this->m_pCombo_comments_colour_control.SetColor(this-
45             >m_pCombo_comments_colour_var);
```

```
45
46     this->m_pCombo_numbers_colour_control.SetColor(this-
47         >m_pCombo_numbers_colour_var);
48     this->m_pCombo_highlight_colour_control.SetColor(this-
49         >m_pCombo_highlight_colour_var);
50
51     this->m_pCombo_fonts_control.SetCurSel(this-
52         >m_pCombo_fonts_control.FindString(-1, this-
53             >m_pCombo_fonts_var));
54     this->m_pCombo_indent_control.SetWindowTextW(this-
55         >m_pCombo_indent_var);
56
57     return 0;
58 }
59
60 BOOL CSmartColourOther::OnKillActive()
61 {
62     this->m_pCombo_comments_type_control.GetLBText(this-
63         >m_pCombo_comments_type_control.GetCurSel(), this-
64         >m_pCombo_comments_type_var);
65     this->m_pCombo_comments_colour_var = this-
66         >m_pCombo_comments_colour_control.GetColor();
67
68     this->m_pCombo_numbers_colour_var = this-
69         >m_pCombo_numbers_colour_control.GetColor();
70     this->m_pCombo_highlight_colour_var = this-
71         >m_pCombo_highlight_colour_control.GetColor();
72
73     this->m_pCombo_fonts_control.GetLBText(this-
74         >m_pCombo_fonts_control.GetCurSel(), this->m_pCombo_fonts_var);
75     this->m_pCombo_indent_control.GetWindowTextW(this-
76         >m_pCombo_indent_var);
77
78     return CMFCPropertyPage::OnKillActive();
79 }
80
81
82 BEGIN_MESSAGE_MAP(CSmartColourOther, CMFCPropertyPage)
83 END_MESSAGE_MAP()
```

```
1 #pragma once
2 #include "afxdialogex.h"
3
4
5 // CAboutDlg dialog
6
7 class CAboutDlg : public CDlgEx
8 {
9
10 public:
11     CAboutDlg(CWnd* pParent = nullptr);    // standard constructor
12     virtual ~CAboutDlg();
13
14 // Dialog Data
15 #ifdef AFX_DESIGN_TIME
16     enum { IDD = IDD_APP_ABOUT };
17#endif
18
19 protected:
20     virtual void DoDataExchange(CDataExchange* pDX);    // DDX/DDV ↗
21             support
22     DECLARE_DYNAMIC(CAboutDlg)
23     DECLARE_MESSAGE_MAP()
24 };
25
```

```
1 // CAboutDlg.cpp : implementation file
2 //
3
4 #include "pch.h"
5 #include "DesignArk.h"
6 #include "afxdialogex.h"
7 #include "CAboutDlg.h"
8
9
10 // CAboutDlg dialog
11
12 IMPLEMENT_DYNAMIC(CAboutDlg, CDialogEx)
13
14 CAboutDlg::CAboutDlg(CWnd* pParent /*=nullptr*/)
15     : CDialogEx(IDD_APP_ABOUT, pParent)
16 {
17
18 }
19
20 CAboutDlg::~CAboutDlg()
21 {
22 }
23
24 void CAboutDlg::DoDataExchange(CDataExchange* pDX)
25 {
26     CDialogEx::DoDataExchange(pDX);
27 }
28
29
30 BEGIN_MESSAGE_MAP(CAboutDlg, CDialogEx)
31 END_MESSAGE_MAP()
32
33
34 // CAboutDlg message handlers
35
```

```
1  <!DOCTYPE html>
2  <html>
3  <head>
4
5  <title>Design Ark</title>
6  <link rel="icon" href="img/logo.png">
7
8  <style>
9      div.feature-block{
10
11      padding:5px;
12      margin:10px;
13
14      text-align:center;
15
16  }
17  .feature{
18      text-align:center;
19      display:inline-block;
20
21      padding:10px;
22      margin:10px;
23  }
24  .feature p{
25      font-size:15px;
26      font-family:'Consolas';
27  }
28  .feature h3{
29      font-size:20px;
30      font-family:'Consolas';
31  }
32  .intro{
33      font-size:20px;
34      font-family:'Consolas';
35  }
36 </style>
37
38 </head>
39 <body>
40
41     <?php
42         include 'banner.html';
43         include 'nav.html';
44     ?>
45
46     <p align="center" class='intro'>
47         DesignArk: Pseudo is an app which allows <br>
48         the user to write pseudocode quickly and effectivley. <br>
49     </p>
50
51     <div class="feature-block">
52
53         <div class="feature">
54              ↗
55             <h3>SmartColour</h3>
```

```

C:\xampp\htdocs\DesignArk\index.php
2
56      <p><i>SmartColour allows you to add and change keywords, <br>
57          and their colours, in your pseudo code depending on <br>
58          your preferred writing style</i></p>
59
60      <div class="feature">
61          
62              height="360">
63          <h3>Subline Implementation</h3>
64          <p><i>Subline implementation allows you to add and edit <br>
65              sublines seamlessly <br>
66              in order to write your pseudo code effectivley</i></p>
67      </div>
68
69      <div class="feature">
70          
71              height="360">
72          <h3>Easy Printing and Exporting</h3>
73          <p><i>DesignArk: Pseudo can effectivley print your pseudo <br>
74              code files <br>
75              and export them to PDF format</i></p>
76      </div>
77
78      <?php
79          include 'footer.html';
80      ?>
81
82      <script>
83          // Page dependant script
84          document.getElementById('home').style="background-color: rgb(175, 73, 73);";
85
86          document.getElementById('footer-link-first').innerHTML='Downloads';
87          document.getElementById('footer-link-second').innerHTML='User-Docs';
88          document.getElementById('footer-link-first').href="downloads.php";
89          document.getElementById('footer-link-second').href="userdocs.php";
90      </script>
91
92  </body>
93  </html>
94

```

```
1 <!DOCTYPE html>
2 <html>
3 <head>
4
5 <title>Design Ark</title>
6 <link rel="icon" href="img/logo.png">
7
8 <style>
9
10    h2{
11        font-family:'Consolas';
12        font-size:30px;
13    }
14
15    div.container{
16
17        text-align:center
18    }
19
20    div.downloadTray{
21
22        background-color:aliceblue;
23
24        border: 1px;
25        border-style: solid;
26        border-color: black;
27        border-radius:10px;
28
29        margin: 25px;
30        padding:0px;
31        padding-bottom:10px;
32
33        display:inline-block;
34    }
35
36    div.downloadTray .title{
37
38        margin:0px;
39        font-family:'Consolas';
40    }
41
42    div.downloadTray .record{
43
44        margin:0px;
45        padding:0px;
46    }
47
48    div.downloadTray .record ul{
49
50        list-style-type: none;
51
52        margin: 0;
53        padding: 3px;
54
55    }
56
```

```
57     div.downloadTray .record li{
58         margin-left:30px;
59         margin-right:30px;
60         padding:0;
61         display:inline-block;
62
63         font-family:'Consolas';
64     }
65
66 </style>
67
68 </head>
69 <body>
70
71     <?php
72         include 'banner.html';
73         include 'nav.html';
74     ?>
75
76     <h2 align="center">Downloadable Files</h2>
77
78     <div class="container">
79
80         <div class="downloadTray">
81             <div class="title">
82                 <h3 align="center">Design Ark Downloads</h3>
83             </div>
84             <div class="record">
85                 <ul>
86                     <li style="margin-right:5px"></li>
87                     <li style="margin-left:5px">DesignArk: Pseudo</li>
88                     <li style="margin-left:5px">Windows (x64)</li>
89                     <li>v1</li>
90                     <li><a href="downloads/DesignArk.exe" download>Download</a></li>
91                 </ul>
92             </div>
93         </div>
94
95         <div class="downloadTray">
96             <div class="title">
97                 <h3 align="center">User Documentation</h3>
98             </div>
99             <div class="record">
100                <ul>
101                    <li style="margin-right:5px"></li>
102                    <li style="margin-left:5px">DesignArk: Pseudo</li>
103                    <li>v1</li>
104                    <li><a href="downloads/userdocs.pdf" download>Download</a></li>
105                </ul>
106            </div>
```

```
107      </div>
108
109    </div>
110
111    <?php
112      include 'footer.html';
113    ?>
114
115
116    <script>
117      // Page dependant script
118      document.getElementById('downloads').style="background-
119        color: rgb(175, 73, 73);";
120
121      document.getElementById('footer-link-
122        first').innerHTML='Home';
123      document.getElementById('footer-link-
124        second').innerHTML='User-Docs';
125      document.getElementById('footer-link-first').href ".";
126      document.getElementById('footer-link-
127        second').href="userdocs.php";
128
129    </body>
130  </html>
131
```

```
1  <!DOCTYPE html>
2  <html>
3  <head>
4
5  <title>Design Ark</title>
6  <link rel="icon" href="img/logo.png">
7
8  <style>
9
10 .viewer{
11     width:75%;
12     height: 900px;
13 }
14 div .title{
15     font-family:'Consolas';
16     font-size:30px;
17 }
18 div .link{
19     font-family:'Consolas';
20 }
21
22 </style>
23
24 </head>
25 <body>
26
27     <?php
28         include 'banner.html';
29         include 'nav.html';
30     ?>
31
32     <div>
33         <h2 align="center" class="title">User Documentation for      ↗
34             Design Ark: Pseudo</h2>
35         <p align="center" class="link"><a href="downloads/
36             userdocs.pdf" target="_blank">View in Fullscreen</a></p>
37     </div>
38
39     <p align="center">
40         <iframe class="viewer" src="downloads/userdocs.pdf"></iframe>
41     </p>
42
43     <?php
44         include 'footer.html';
45     ?>
46
47     <script>
48         // Page dependant script
49         document.getElementById('userdocs').style="background-color: ↗
50             rgb(175, 73, 73);";
51
52         document.getElementById('footer-link-
53             first').innerHTML='Home';
54         document.getElementById('footer-link-
55             second').innerHTML='Downloads';
56
```

```
52     document.getElementById('footer-link-first').href ".";
53     document.getElementById('footer-link-
      second').href = "downloads.php";
54 </script>
55
56 </body>
57 </html>
58
```



```
1 <head>
2 <style>
3
4 html body {
5     margin:0;
6     padding:0;
7     background-color:antiquewhite
8 }
9
10 .main-header{
11
12     background-color: teal;
13     text-align: center;
14
15     height: 90px;
16     width: 100% - 2px;
17
18     border: 1px;
19     border-style: solid;
20     border-color: black;
21
22     padding-top: 5px;
23     padding-bottom: 5px;
24 }
25
26 </style>
27 </head>
28
29 <header class="main-header" role="banner">
30     <a href="..">
31         
32     </a>
33 </header>
34
35
```

```
1 <style>
2
3 div.navbar{
4     background-color: brown;
5
6     border:1px;
7     border-top: 0px;
8     border-style: solid;
9     border-color: black;
10
11    width:100% - 2px;
12 }
13
14
15 div.navbar ul {
16
17     text-align: center;
18
19     background-color: brown;
20     list-style-type: none;
21
22     margin: 0;
23     padding: 0;
24
25     overflow: hidden;
26
27     width:100%;
28 }
29
30
31 div.navbar li {
32     display:inline-block;
33 }
34
35 div.navbar li a {
36
37     color: black;
38
39     display: block;
40     padding: 14px 16px;
41
42     text-decoration: none;
43
44     font-family:'Consolas';
45 }
46
47 div.navbar li a:hover {
48     background-color: rgb(187, 63, 63);
49 }
50 div.navbar li a:active{
51     background-color: rgb(189, 89, 89);
52 }
53
54 </style>
55
56 <div class="navbar">
```

```
57      <ul>
58          <li id="home"><a href=".">Home</a></li>
59          <li id="downloads"><a href="downloads.php">Downloads</a></li>
60          <li id="userdocs"><a href="userdocs.php">User-Docs</a></li>
61      </ul>
62  </div>
```

```
1 <style>
2
3
4     .footer{
5
6         background-color: brown;
7
8         border:1px;
9         border-style: solid;
10        border-color: black;
11
12        bottom:0;
13        width:100% - 32px;
14
15        padding: 15px;
16        padding-top:5px;
17
18        position:static;
19    }
20
21    .footer p{
22
23        font-family:'Consolas';
24    }
25
26    .footer .topButton{
27        float:right;
28        color:black;
29        cursor:pointer;
30
31        font-family:'Consolas';
32    }
33
34    .footer .footer-links{
35
36        text-align: center;
37        list-style-type: none;
38
39        margin: 0;
40        padding: 0;
41
42    }
43
44    .footer .footer-links li {
45        display:inline-block;
46    }
47
48    .footer .footer-links li a {
49
50        color: black;
51
52        padding-left: 14px;
53        padding-right: 14px;
54
55        font-family:'Consolas';
56    }
```

```
57
58 </style>
59
60 <script>
61     function backToTop() {
62         document.body.scrollTop = 0;
63         document.documentElement.scrollTop = 0;
64     }
65 </script>
66
67 <footer id="footer" class="footer">
68
69     <p>
70         Contact us: <br>
71         Email: <a href="mailto:admin@designark.co.uk" style="color: black;">admin@designark.co.uk</a> <br>
72         Phone No: +1 234 567890<br>
73     </p>
74
75     <a id="topButton" onclick="backToTop()" class="topButton"><u>Back to Top</u></a>
76
77     <ul class="footer-links">
78         <li><a href="downloads.php" id="footer-link-first">Downloads</a></li>
79         <li><a href="userdocs.php" id="footer-link-second">User-Docs</a></li>
80     </ul>
81
82
83 </footer>
```

## Final Product

## Application

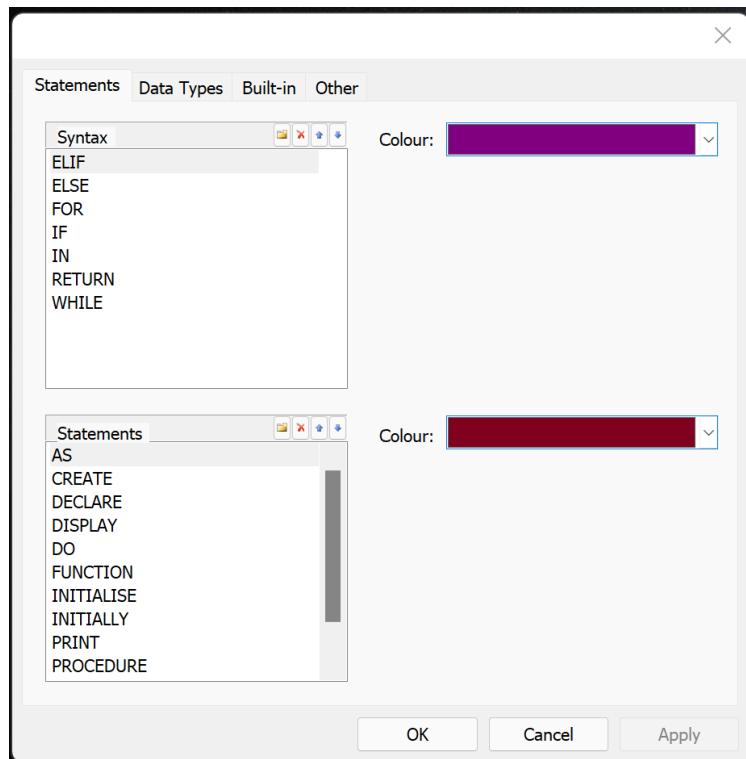
File Edit View Window Help

InsertionSort X

```
1 // To meet the SQA Advanced Higher Computing Science criteria,
2 // the program implements a insertion sort in all
3 // CSmartColour's property pages when closing them
4
5 // The following pseudocode is a design for CSmartColourTypes' sort
6 // It will show how it works in practise for the rest of the property sheets
7
8 FOR i IN RANGE(1, m_Listbox_Types_control.SizeOf()):
9
10    CREATE variables
11
12
13 WHILE pos > 0 AND m_Listbox_Types_control[pos-1] > currentValue:
14     SET m_Listbox_Types_control[pos] TO m_Listbox_Types_control[pos - 1]
15     pos -= 1
16
17 m_Listbox_Types_control = currentValue
18
19

11.1 DECLARE CString currentValue INITIALLY AS
11.2 DECLARE INT pos INITIALLY AS i
```

Ready CAP NUM



## Website

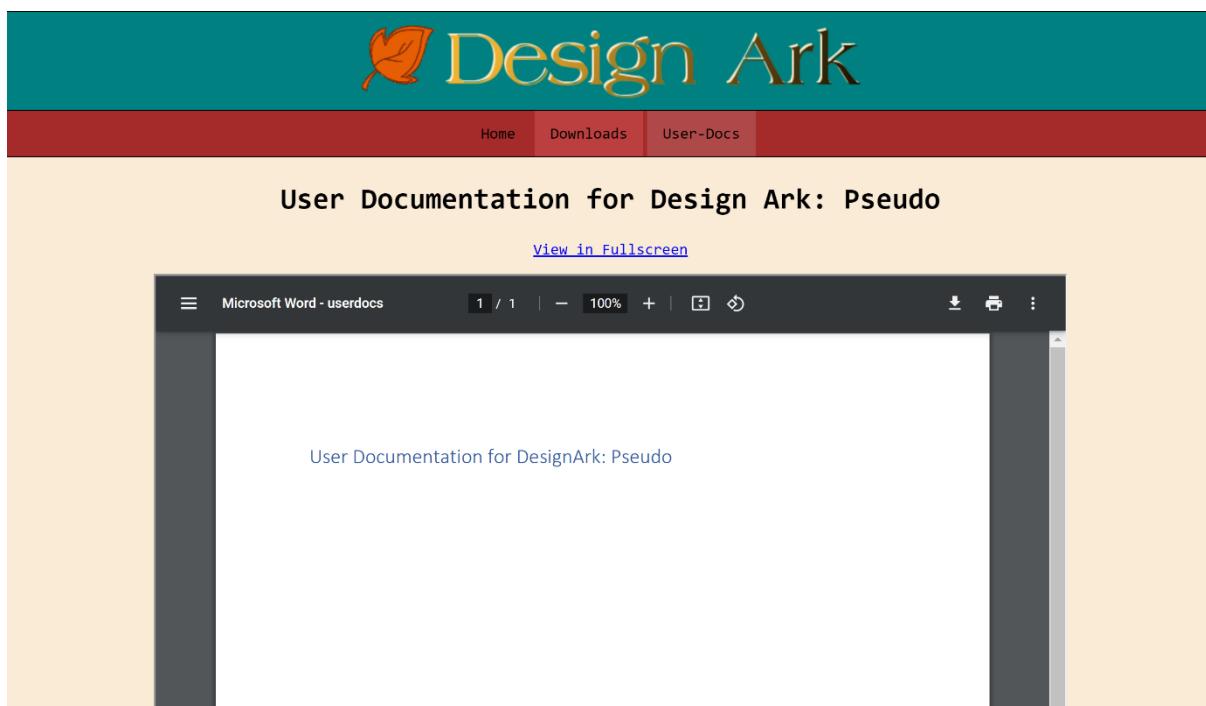
*Home page:*

The screenshot shows the Design Ark Home page. At the top is a teal header with the logo 'Design Ark' and a leaf icon. Below it is a red navigation bar with 'Home', 'Downloads', and 'User-Docs'. The main content area has a light beige background. It features three sections: 'SmartColour' (a screenshot of a window showing a color palette and code editor), 'Subline Implementation' (a screenshot of a window showing a code editor with sublines), and 'Easy Printing and Exporting' (a screenshot of a PDF file titled 'Findthemax.txt' containing pseudocode). Below these sections is a red footer with contact information: 'Contact us: Email: jamesmcorkindale@gmail.com Phone No: 07482 561740'. It also includes links for 'Downloads' and 'User-Docs'.

*Downloads page:*

The screenshot shows the Design Ark Downloads page. At the top is a teal header with the logo 'Design Ark' and a leaf icon. Below it is a red navigation bar with 'Home', 'Downloads' (which is highlighted in red), and 'User-Docs'. The main content area has a light beige background and features two boxes: 'Design Ark Downloads' (listing 'DesignArk: Pseudo Windows (x64) v1 Download') and 'User Documentation' (listing 'DesignArk: Pseudo v1 Download'). Below these boxes is a red footer with contact information: 'Contact us: Email: admin@designpark.co.uk Phone No: +1 234 567890'. It also includes links for 'Home' and 'User-Docs'.

User docs page:



## New Skills/Knowledge Evaluation

While implementing this project, I have learned many new skills:

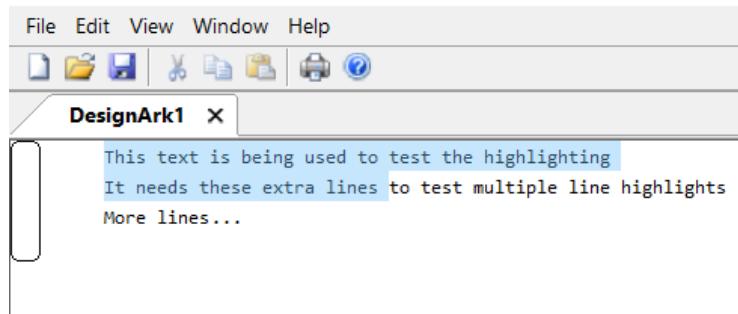
- Most notably, I have learned how to effectively use the Microsoft Foundation Classes (MFC) framework, which is the basis of the application that I have made. I have learnt most of the basic skills and architecture needed to create a Windows app, including:
  - The Document/View architecture
  - The MFC Print architecture
  - Using scroll views
  - Making modal and modeless dialog boxes
  - Message maps and message handling
  - Using the CRgn class to make more efficient drawing and rendering
  - Using a caret
  - Changing cursor appearances
  - Using the save and open dialog boxes
- I have also gained basic knowledge in using the GDI+ library to have more complex rendering. This was used to make a transparent rectangle for the highlighting and make dynamically sizing bitmap images.
- I have learned the very basics of the Win32 API, however, very minimal amounts of this API were used in the implementation
- I have learned how to use the Windows Clipboard API, to copy/paste text to and from the OS' clipboard

## Log of On-Going Tests

### Highlighting Bug

An error occurs when highlighting a new editor object for the first time. The error causes the highlight to highlight from the start of the editor (line 0, caretPos 0) to the spot the user is currently on.

#### *Highlighting Glitch:*



To debug this error and to see what's happening, we should set break points at two places. One when we draw the editor and one when we create the highlight.

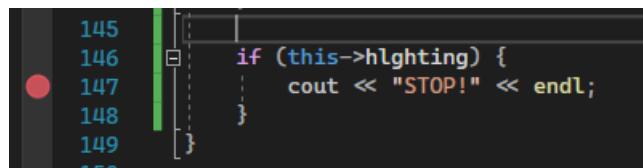
#### *Breakpoints:*



```
142     it->draw(pDc, this->text)
143 }
144 }
145 cout << "STOP!" << endl;
146 }
147
1029
1030     this->caretPos = endPos;
1031 }
1032
1033 cout << "STOP!" << endl;
1034 }
1035 +void CTextEditorObject::setBrackets
```

When we attempt to debug this however, there is a problem. Because we stop the program (and close the app) every time the breakpoint hits, whenever we continue the program, we reopen the app => redraw the app => setting off the breakpoint again, and we are stuck in a loop we can't break, so it will be impossible to test this. We can fix this by adding a condition on the breakpoint in the draw function; that we only set it off if we are highlighting.

#### *New Breakpoint:*



```
145     if (this->highlighting) {
146         cout << "STOP!" << endl;
147     }
148 }
149 }
```

Now, when we debug this and see are variable values, we can see that hlightEndP is set to (0, 0) for the first frame of highlighting. This would then make sense that the highlighting is drawing from there then. Now, we just need to find out where we are setting it to (0, 0), or if we are not setting it in the first place and it is just using its default value.

We set hlightEndP only in the draw function of the object, so we should start by looking there to see if there is anything that doesn't look right.

Where we set hlightEndP:

```
if (nFlags == MK_LBUTTON) {  
    if (this->checkPointInBounds(point)) {  
        this->setHighlighter();  
        this->hlightEndP = point;  
    }  
    else if (point.y < this->bounds.top) {  
        this->hlightEndP = CPoint(this->lines[0]->getBounds().left + 1, this->bounds.top);  
    }  
    else {  
        this->hlightEndP = CPoint(this->lines[this->lines.size() - 1]->getBounds().right - 1, this->bounds.bottom);  
    }  
    redraw = TRUE;  
}  
else {  
    this->highlighting = FALSE;  
}
```

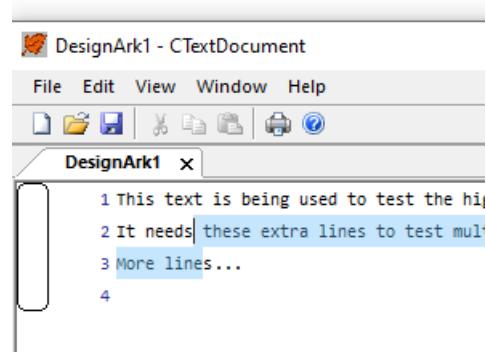
It's not obvious at first, but we are setting the hlightEndP AFTER we create the highlighter. So, it would make sense that for the first frame, the class sets the highlighter using the default value of the CPoint class (0, 0). So, if we swap the setHighlighter() call and the setting of hlightEndP, that will probably fix the error.

Fixed code:

```
if (nFlags == MK_LBUTTON) {  
    if (this->checkPointInBounds(point)) {  
        this->hlightEndP = point;  
        this->setHighlighter();  
    }  
    else if (point.y < this->bounds.top) {
```

Now when we run that, we can see it works.

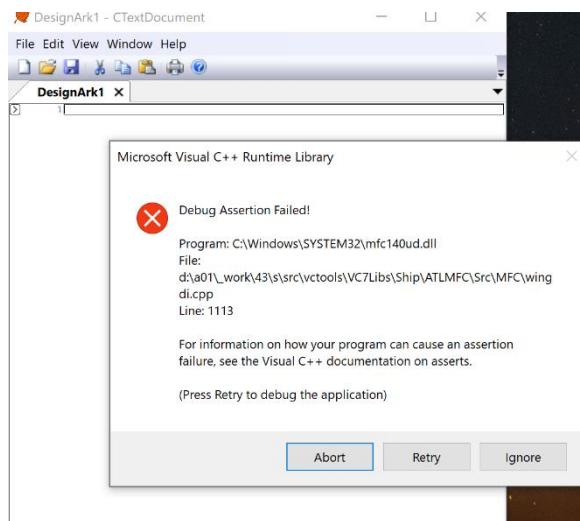
Fixed app:



## Error When Pressing the Sidebar Button After Highlighting

There is a bug that occurs when the user attempts to sidebar button after they have highlighted.

*Initial Failure:*



*Occurring From:*

A screenshot of a debugger interface showing a portion of C++ code. The code is as follows:

```
if (type == 0) {
    VERIFY(this->rgn->CreateRectRgn(x1, y1, x2, y2));
}
else {
    CRgn* rgn1 = new CRgn();
    VERIFY(rgn1->CreateRectRgn(x1, y1, x2, y2));
    this->rgn->CombineRgn(this->rgn, rgn1, RGN_OR);
}

break;
```

A red circle with a white 'X' marks the line number 672, indicating a breakpoint. A tooltip "Exception Thrown" points to the line, and another tooltip "DesignArk.exe has triggered a breakpoint." is visible. There are also "Copy Details" and "Start Live Share session..." options.

From this, we can see that the problem is because we are trying to initialise the rgn object despite already having initialised it. We have had this problem before, and the solution was to put in the two different clauses (initialise rgn or combine a CRgn with rgn).

When we trace this thread, we can see that the problem occurs in the previous function (OnLButtonDown), when we decide what value of 'type' we will use.

*Not selecting the type to use correctly:*

A screenshot of a debugger showing the assembly code for the function. The code is:

```
671     if (sidebarbtn) {
672         if (this->recentLine == this->recentBlockLine) {
673             this->selectClippingRgn(ADD_RGN_SIDEBAR_BTN, 0);
674         }
675         else {
676             this->selectClippingRgn(ADD_RGN_SIDEBAR_BTN, 1);
677         }
678     }
679 }
```

The line 675 is highlighted with a red arrow pointing to it, indicating the current instruction being executed.

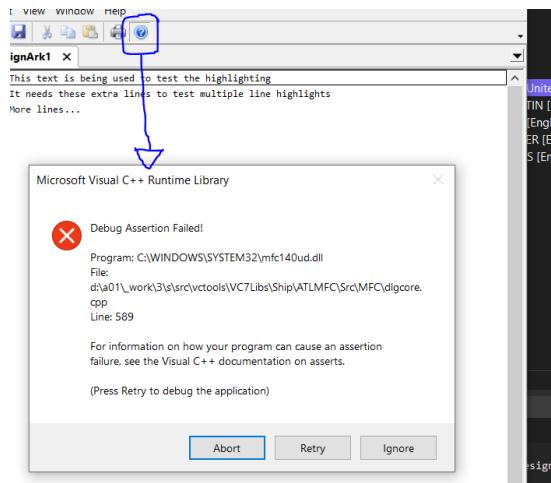
We can see we are selecting the '0' type at line 675, when we should be selecting the '1' type in line 678.

*More here...*

## About Dialog Assertion Failure

This failure arises from clicking the 'About' button and sending the ID\_APP\_ABOUT message to the program. This failure was not always the case as it was working when it was first implemented, so obviously something has been changed since then by accident.

### Initial Failure:



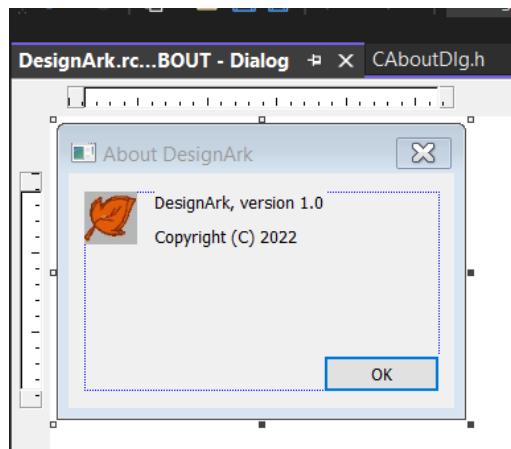
### Occurring From:

```
27 // Public Commands
28 void CDesignArkApp::OnSmartColour() { ... }
29
30 //void CDesignArkApp::OnAppAbout() // This bit is not working
31 {
32     CAboutDlg aboutDlg;
33     aboutDlg.DoModal(); // This line is triggering a debug assertion failure
34 }
35
36 void CDesignArkApp::OnOpenSite() { ... }
37 void CDesignArkApp::OnUserdocs() { ... }
38
39 // Protected Implementation
```

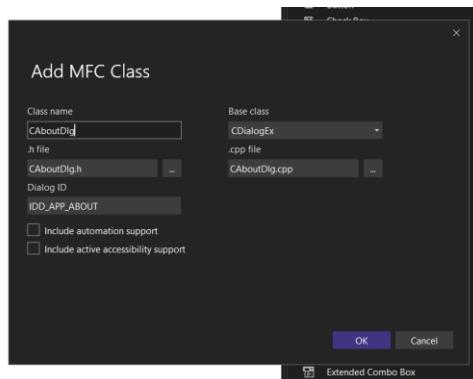
Upon a quick inspection, a problem cannot be easily found. So, at this point, there are two options. 1: To search through the code line by line and possibly find a problem, or 2: Just delete all the code and dialog box and remake everything. The second option will most likely be quicker, and easier, so we will go for that.

After deleting everything, we need to make a new dialog box, using the in-built designer in Visual Studio 2022, we will then make the class and then add the code.

### New Dialog Box:



**New Class:**



**New Code:**

```
#pragma once
#include "afxdialogex.h"

// CAboutDlg dialog

class CAboutDlg : public CDlgEx
{
    DECLARE_DYNAMIC(CAboutDlg)

public:
    CAboutDlg(CWnd* pParent = nullptr); // standard constructor
    virtual ~CAboutDlg();

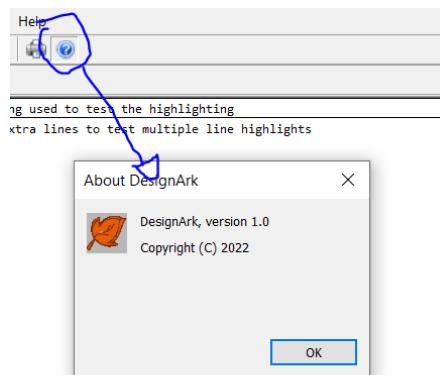
// Dialog Data
#ifdef AFX_DESIGN_TIME
    enum { IDD = IDD_APP_ABOUT };
#endif

protected:
    virtual void DoDataExchange(CDataExchange* pDX); // DDX/DDV support
    DECLARE_MESSAGE_MAP()
};


```

We have now implemented a new CAboutDlg (however, it is still identical to the old one), so we need to test it again to see if it works.

**About Dialog Test:**

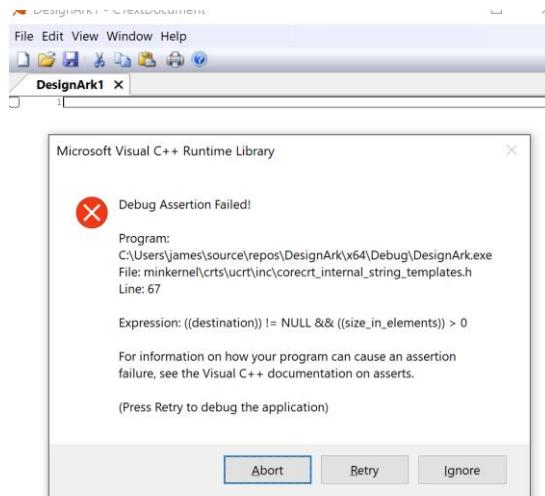


It is now working as intended.

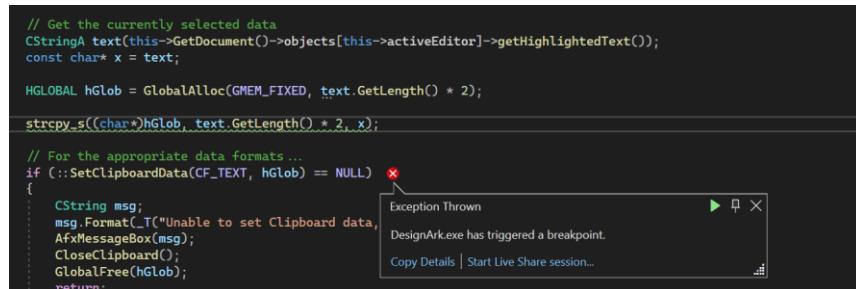
## Copying Nothing Bug

While implementing the copying to the clipboard, a bug was noticed that causes an assertion failure when the user attempted to copy (CTRL + C or using a copy button in toolbar/menu) and had not of highlighted anything.

*Initial Failure (After attempting to copy):*



*Occurring From:*



From this, we can see that the error occurs due to the variable 'x' being void. To fix this, we can simply put a check in the OnCopy function to see if we are highlighting.

*Highlight Check:*

```
if (this->GetDocument()->objects[this->activeEditor]->hasHighlight()) {  
    if (!OpenClipboard())  
    {  
        AfxMessageBox(_T("Cannot open the Clipboard"));  
        return;  
    }  
}
```

Now, after testing this again, it works and does not create an error.

Test Case ID	Functional Requirement	Test Case Objective	Test Case Description	Expected Result
		Create a new file from the menu and tool bar. Open a text file with some basic psuedo code, that includes sublines. 'InsertionSort.txt' from the Design section of this project will be used.	Creates a basic file (with sublines) and save it, making sure that it saves with the appropriate line spacing, line numbers and text.	Opens a new file that is either blank if it was created new. If opened 'InsertionSort.txt', it should be filled with the text of 'InsertionSort.txt' with the sublines used.
1 Creating a New File/Opening a File		Ensure that new blank files are created and that text files can be opened (with CEditorObjects if there are sublines).	Saves a txt file of the document that was saved. It has the line numbers on each line with the appropriate spacing.	Saves a 'txt' file of the document that was saved. It has the line numbers on each line with the appropriate spacing.
2 Save Files				
		Ensure that the app can connect to a printer and print the correctly formatted file. Ensure that the file can be exported as a PDF, in the same format as the printed format.	Print the 'InsertionSort.txt' file to my home printer and two different school printers, to ensure that the printing is consistent even with different printers. While printing each time, export to PDF at the same time and check if the PDF and print match.	Prints the pages on each printer exactly as it was highlighted. There should be correct line spacing as well. It should also be visible on the OS clipboard.
3 Print file and export as PDF				
4 Edit OS print settings		Ensure that the app can edit the OS print settings when printing, such as print colour, double sided, page size etc..	Print the 'InsertionSort.txt' file to a printer twice, each time with different colour settings, to ensure that the print settings work and are consistent.	Prints the pages, using the correct colour settings. Each time printing with the same settings on different printers should have identical prints.
5 Exiting the application		Ensure that the app can close without any errors like memory leaks or assertion function, top right 'X' button on window and via the task bar.	Close the application several times and test if it works. Test close from menu and via the task bar.	Application closes, is removed from the task bar and from the OS task manager.
6 Copy text to the clipboard		Ensure that the user will be able to copy highlighted text to the clipboard from the menu, the toolbar and commands (CTRL + C).	Using 'InsertionSort.txt', copy one of the editors and attempt to paste it into a text editor.	Highlighted text will be copied and will be pasted into Microsoft word, exactly as it was highlighted. There should be correct line spacing as well. It should also be visible on the OS clipboard.

		Application adds the text into the current editor, from the active line in the current caret position. It creates a new line when it receives a carriage character (\n).
Paste text from the clipboard into the application 7	editor, including a carriage character to add new lines.	Create a small word file in a text editor (including some keywords) and copy it, attempt to paste it into the application.
Opening customise window dialog 8	Ensure that the customise dialog box can open without any faults or errors.	Open the window in several different running instances of the application and check for errors. No need to test the actual functionality of the dialog box as this is being handled by the OS and the MFC framework.
9 Enabling/disabling the tool bar	Ensure that the tool bar can be enabled and disabled.	Use this button several times in several different instances of the application and check for failures.
10 Enabling/disabling the status bar	Ensure that the status bar can be enabled and disabled.	Use this button several times in several different instances of the application and check for failures.
11 Changing the windows theme	Ensure the user can change the windows theme to a selection of in-built themes.	Use this function several times in several different instances of the application, with different themes, and check for failures.
12 Close any tabs	Ensure that any tabs can be closed from the main window, or from a dialog box that specifically manages active tabs.	Use this button and function several times in several different instances of the application and check for failures.
13 Open a new window from the app	Ensure that new instances of the app can be made from the menu bar.	Use this function several times in several different instances of the application and check for failures.
14 Use the SmartColour dialog box	Add/remove several words to/from each list box and check that they are being implemented by the application. Change several colours and check that they are also being implemented. Test each font to make sure that all are able to be used. Make sure all changes happen as soon as the dialog is closed.	SmartColour dialog can be opened without any faults or errors.  Keywords can be added and removed. Colours can be changed. Fonts and indentation size can be changed. When the dialog is closed, the changes are automatically updated.

		Use this function several times in several different instances of the application and check for failures. Also test it with different default browsers (Microsoft Edge and Google Chrome).	Website opens in the default browser.
15 Open the website from the app	Ensure that the website can be opened from the menu bar in the default browser.		
16 Open/close the about dialog box	Ensure that the about dialog box opens and closes without error.	Open/close the about dialog several times in several different instances of the application and check for failures.	About dialog box is shown/closed.
17 Add/remove text from lines	Ensure that the user can add/remove text to/from the editor.	Write a basic file and make sure there are no problems when adding or removing text.	Text is deleted and removed from the line. Text is added and rendered on to the line. Text is edited at caret position.
		Write a basic file and make sure there are no problems when adding or removing lines and sublines. Check that when adding empty or contains the text that was sublines, if it already exists, the application to the right of the caret. This line is selected. A new subline is created in that when removing lines (via highlight or backspace) that if its subline exists, don't remove it.	Lines and/or a subline is/are deleted and removed from editor/file. A line or subline is added and rendered on to the line. A new line is created in the line underneath the old line, it is selected. A new subline is created in that when removing lines (via highlight or backspace) that if its subline exists, don't remove it.
		Ensure that the user can add/remove line and sublines to/from the editor.	
		Ensure that when lines are added/removed, that sublines are incremented appropriately.	
	Add/Remove lines and subline 18 blocks		

Subline button arrow is rendered when hovering and selecting it. The highlighter is in the appropriate position as it should be expected to be.. It is also transparent and the colour that is selected in the SmartColour. The sidebar rectangle reaches from the top to bottom of each editor, each editor has its own sidebar. Text and numbers coloured depending on the SmartColour colours. Line numbers render as the one colour. Caret renders where the editor, line and caretPos is selected. Cursor image depends where in the screen the cursor has moved to. It should change depending on where it is.

Check that each GUI object is being rendered, do this by running the application and looking at each object. (Subline button arrow, highlighter, selected line rectangle, sidebar, text and line numbers, caret, cursor).

Ensure that all GUI objects render efficiently and appropriately.

## 19 Run GUI objects and interface

## Persona Testing

1. Catherine is a teenager studying Higher Computing Science in school. She needs to write pseudo code for her latest coding project and is going to use Design Ark as it is already on her school computers. She has a fair degree of skill using computers, she is fluent in using Windows applications and has had lots of experience in using them.

In general, Catherine found the program easy to use. She enjoys using the subline features which she hasn't seen in any other apps. However, she has discovered a couple bugs which makes the program not run correctly. Most of the time she must restart the application and loses her work. She says that this might put her off from using the application in the future if it is not fixed soon.

2. Lucy is a senior software engineer. She has worked with computers for her entire career and is well experienced in all sorts of apps and websites. She needs to upgrade the software used on the computers in her office and has chosen to download Design Ark for her employees.

Lucy found the experience of downloading the application on all the computers easy and quick. She enjoys the fact that the storage that the app takes up on the computers is minimal (a few megabytes). However, when downloading the app, a few of the computer's anti-virus software flagged the download.

When using the app, she likes the simplicity and elegance of the interface. She finds it easy to use and learn. Her favourite part about the app, is the ability to export as PDF, and print the files in a slick format, with easy readability. She also likes the fact that new keywords can be added to be coloured as she tends to write pseudocode in a more unique style than most other developers.

3. Sunny is a graphics designer who has recently started coding in her free time and wants to learn about pseudocode. She has little to no experience using any technology other than the apps she uses for graphic design.

Despite Sunny's lack of computer fluency, she enjoys using the website to download the app and to view the documentation. She finds it extremely simple and informative about the application's capabilities. However, she is mildly put off by the basicness of the website, as a graphic designer, she has come to expect a high level of design in websites.

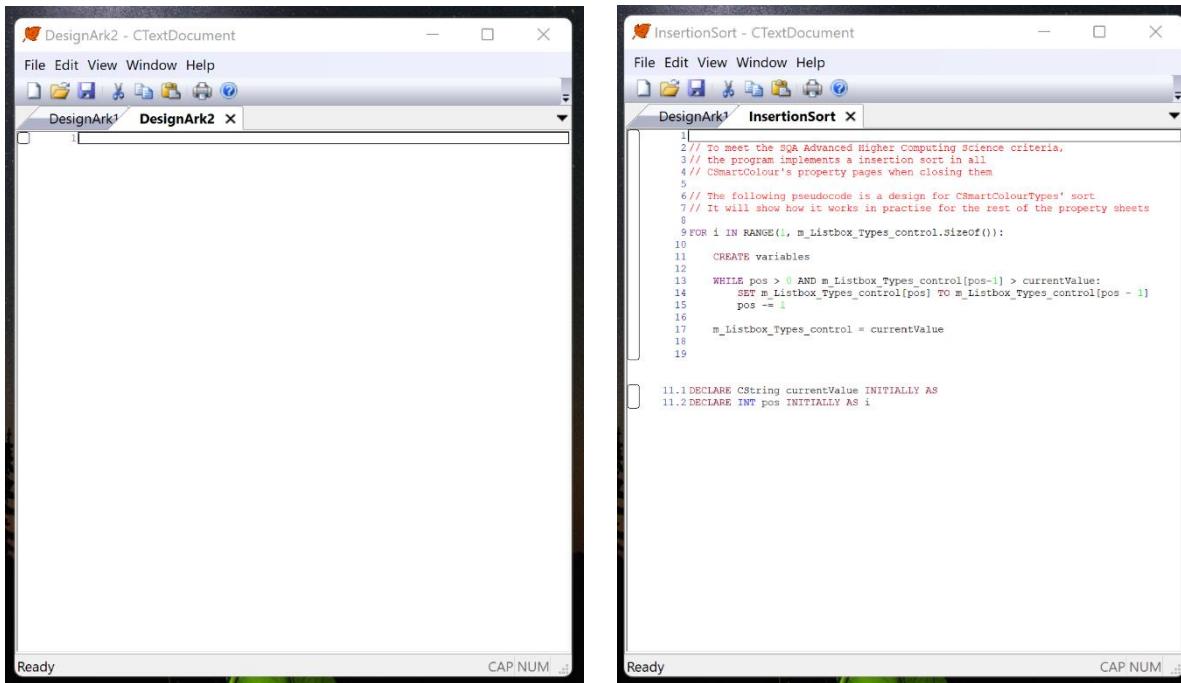
Sunny enjoys the experience of learning about pseudocode while using the application to write. The SmartColour feature excites her as it adds a bit more of customisation to an otherwise simple app. She likes using it to change the highlight colours and fonts. However, she does admit to this feature distracting her from her learning.

## Testing Requirements (Actual Results of Texting)

## Test Case ID 1

*Opened new:*

Opened 'InsertionSort.txt':



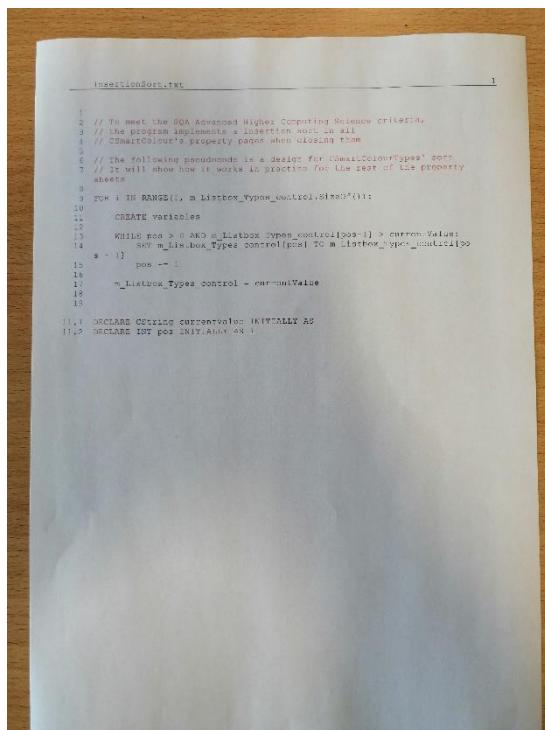
## Test Case ID 2

*Saved in a .txt file:*

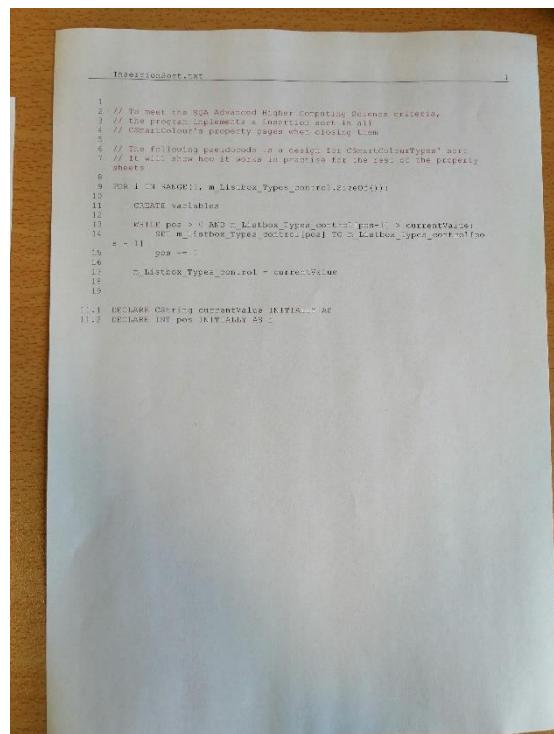
```
1 // To meet the SQA Advanced Higher Computing Science criteria,
2 // the program implements a insertion sort in all
3 // CSmartColour's property pages when closing them
4
5
6 // The following pseudocode is a design for CSmartColourTypes' sort
7 // It will show how it works in practise for the rest of the property sheets
8
9 FOR i IN RANGE(1, m_Listbox_Types_control.SizeOf()):
10
11     CREATE variables
12
13     WHILE pos > 0 AND m_Listbox_Types_control[pos-1] > currentValue:
14         SET m_Listbox_Types_control[pos] TO m_Listbox_Types_control[pos - 1]
15         pos -= 1
16
17     m_Listbox_Types_control = currentValue
18
19
20.1 DECLARE CString currentValue INITIALLY AS m_Listbox_Types_control[i]
20.2 DECLARE INT pos INITIALLY AS i
```

## Test Case ID 3

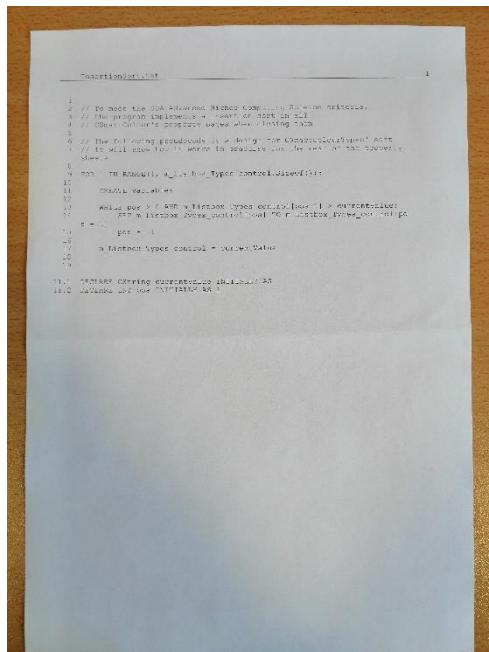
*Home print:*



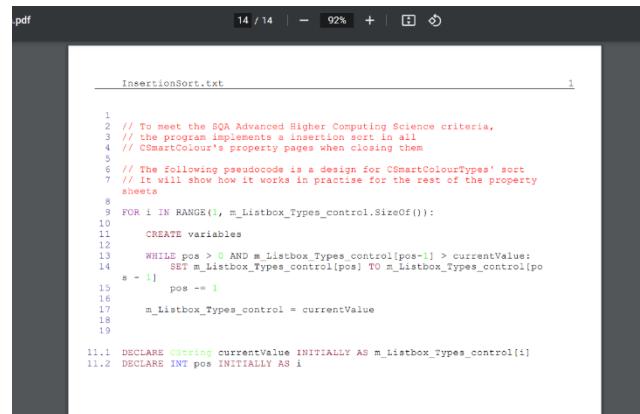
*School print 1:*



*School print 2:*

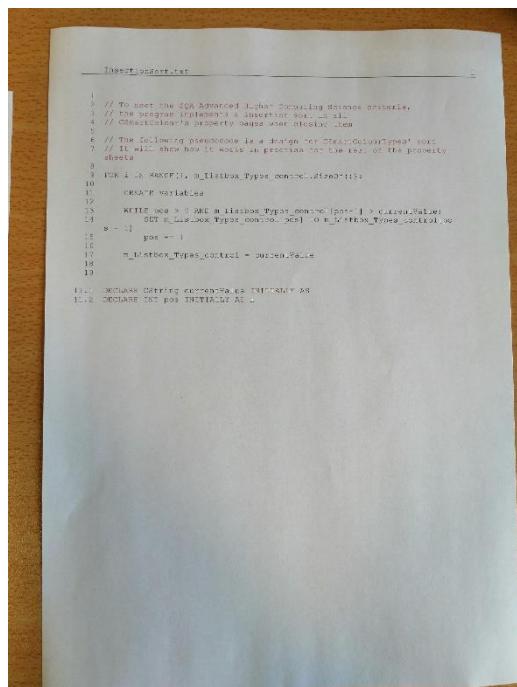


*Exported PDF:*



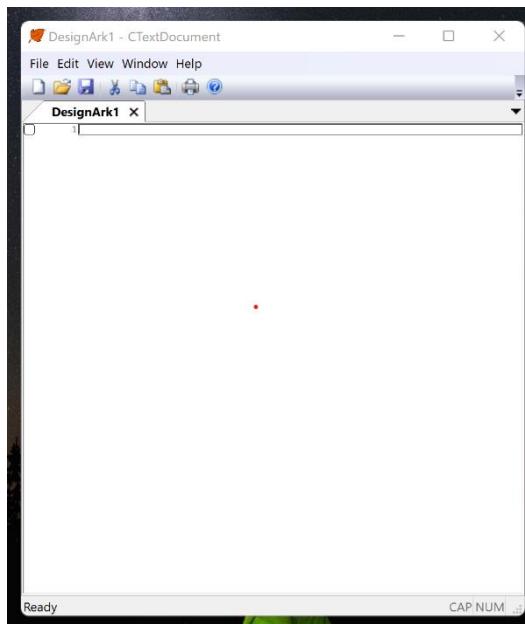
## Test Case ID 4

*Print with colour:*



Test Case ID 5

*Application Open:*



*App on taskbar:*



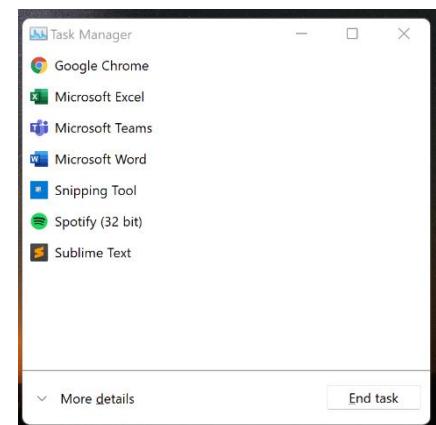
*App no longer on taskbar:*



*App in OS Task Manager:*



*App no longer in OS Task Manager:*



## Test Case ID 6

### *Highlighted and copied text:*

```
1 // To meet the SQA Advanced Higher Computing Science criteria,
2 // the program implements a insertion sort in all
3 // CSmartColour's property pages when closing them
4
5 // The following pseudocode is a design for CSmartColourTypes' sort
6 // It will show how it works in practise for the rest of the property sheets
7
8 FOR i IN RANGE(1, m_Listbox_Types_control.SizeOf()):
9
10    CREATE variables
11
12    WHILE pos > 0 AND m_Listbox_Types_control[pos-1] > currentValue:
13        SET m_Listbox_Types_control[pos] TO m_Listbox_Types_control[pos - 1]
14        pos -= 1
15
16    m_Listbox_Types_control = currentValue
17
18
19
20.1 DECLARE CString currentValue INITIALLY AS
20.2 DECLARE INT pos INITIALLY AS i
```

*After copying, pasted text:*

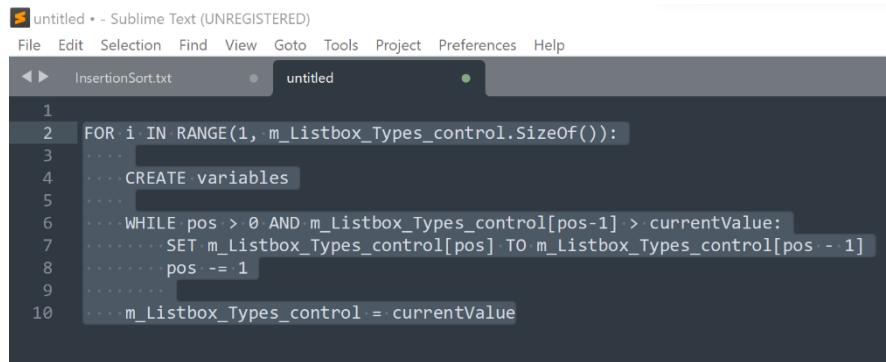
```
FOR i IN RANGE(1, m_Listbox_Types_control.SizeOf()):  
    CREATE variables  
  
    WHILE pos > 0 AND m_Listbox_Types_control[pos-1] > currentValue:  
        SET m_Listbox_Types_control[pos] TO m_Listbox_Types_control[pos - 1]  
        pos -= 1  
  
    m_Listbox_Types_control = currentValue
```

*Text in clipboard:*



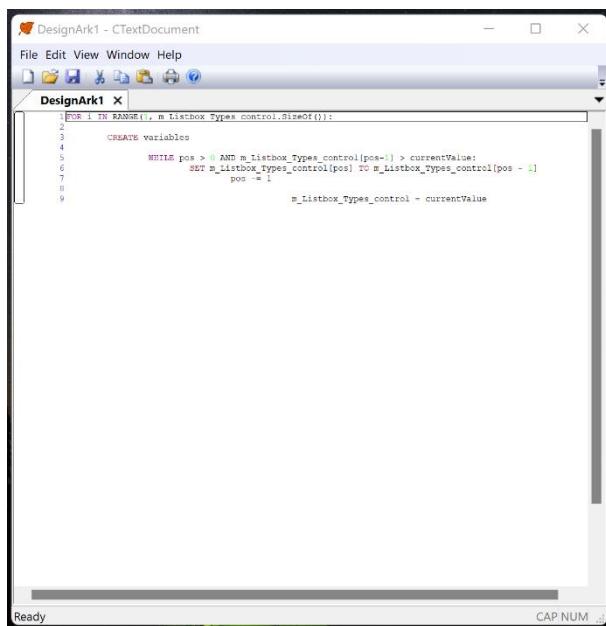
## Test Case ID 7

*Highlighted and copied text:*



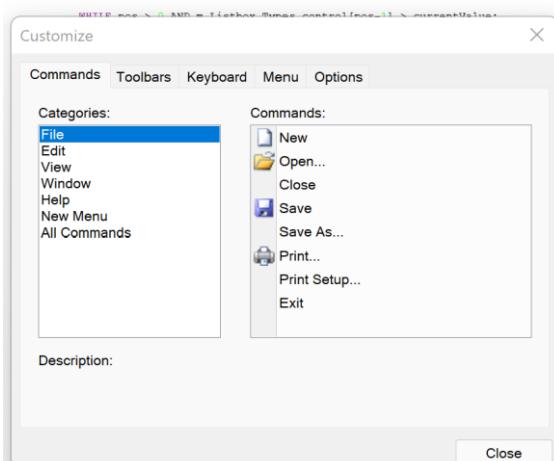
```
untitled • - Sublime Text (UNREGISTERED)
File Edit Selection Find View Goto Tools Project Preferences Help
◀ ▶ InsertionSort.txt • untitled •
1
2 FOR i IN RANGE(1, m_Listbox_Types_control.SizeOf()):
3 ...
4 ... CREATE variables
5 ...
6 ... WHILE pos > 0 AND m_Listbox_Types_control[pos-1] > currentValue:
7 ...     SET m_Listbox_Types_control[pos] TO m_Listbox_Types_control[pos - 1]
8 ...     pos -= 1
9 ...
10 m_Listbox_Types_control = currentValue
```

*Text pasted into the application:*



## Test Case ID 8

*Customise window dialog box successfully opening:*



## Test Case ID 9

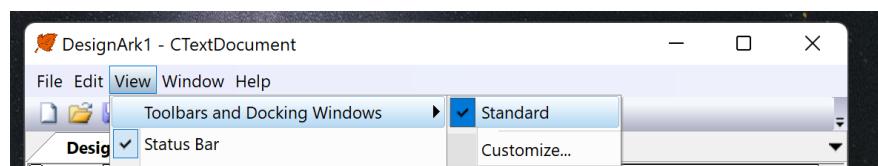
*Toolbar enabled:*



*Toolbar disabled:*



*Button to enable/disable this feature:*



## Test Case ID 10

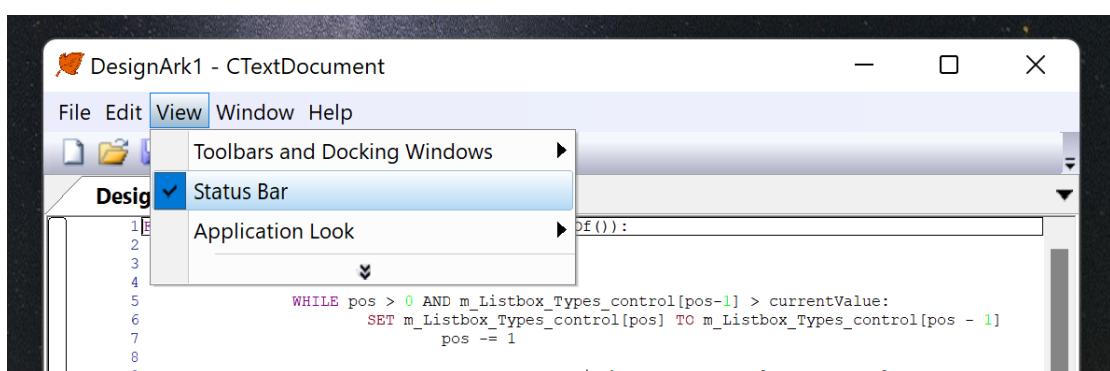
*Status bar enabled:*



*Status bar disabled:*



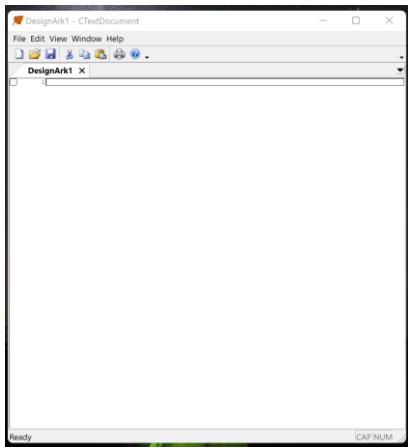
*Button to enable/disable this feature:*



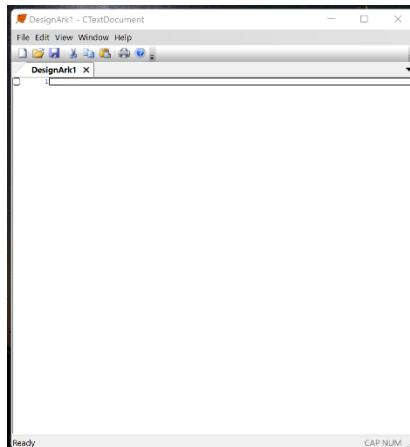
## Test Case ID 11

*Some different windows themes:*

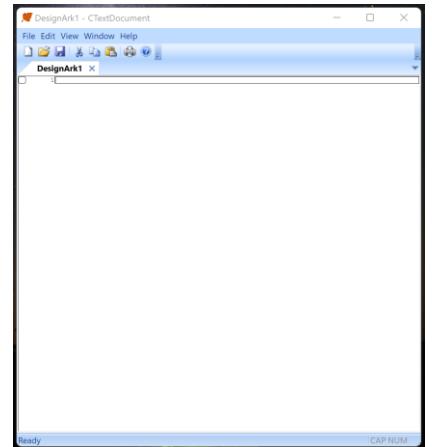
*Windows 2000:*



*Visual Studio 2005:*



*Office 2007:*



## Test Case ID 12

*Tab closing with two+ tabs already open. Before and after:*

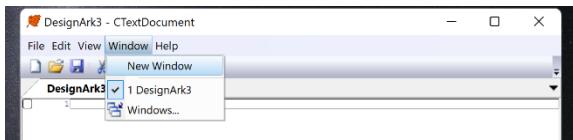


*Tab closing with only one tab already open. Before and after:*



## Test Case ID 13

*New window button:*



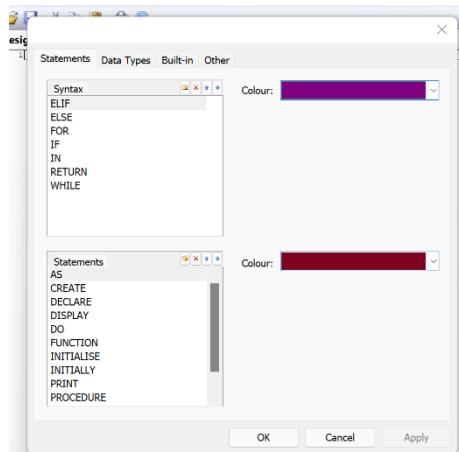
*New window:*



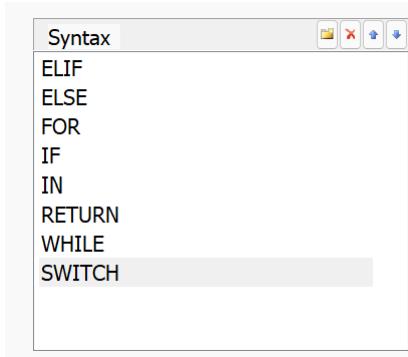
*New window shown on the taskbar:*

Test Case ID 14

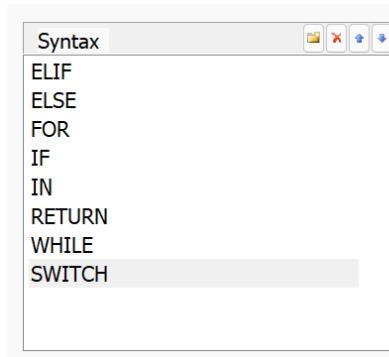
*SmartColour window opening successfully:*



*New keyword can be added:*

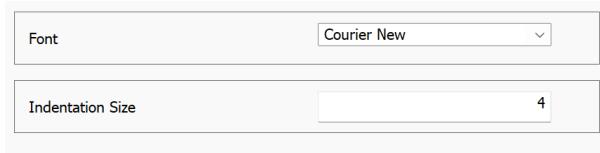


*Colour can be changed:*

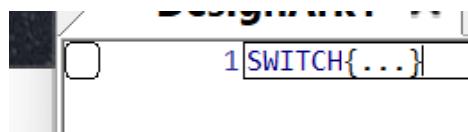


Colour:

*Fonts and indentation can be changed. Before and after:*

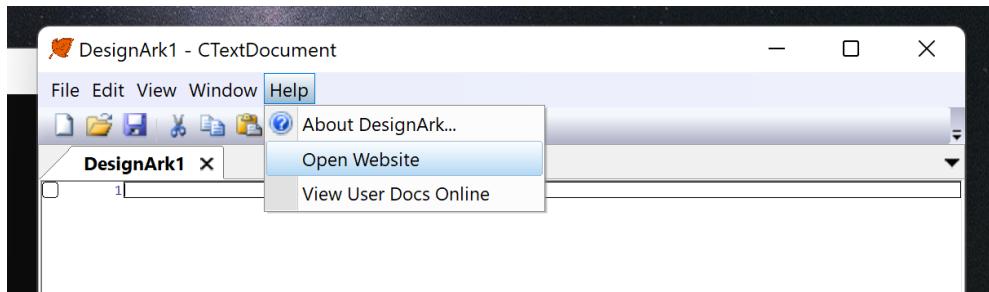


*Changes are updated when SmartColour dialog is close:*



## Test Case ID 15

*Buttons to open the website:*

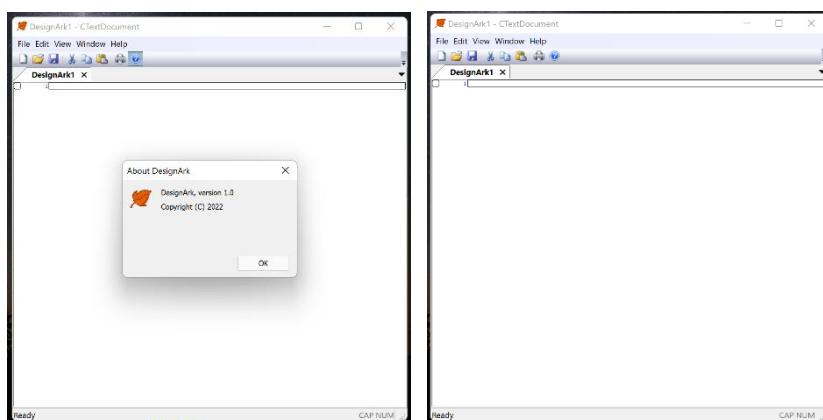


*Opened website (via buttons):*

A screenshot of the Design Ark website. The header features the "Design Ark" logo. Below the header, there are three main sections: "SmartColour" (showing a screenshot of the software interface with colored keywords), "Subline Implementation" (showing a screenshot of the software interface with multiple lines of code), and "Easy Printing and Exporting" (showing a screenshot of the software interface with a print preview). At the bottom of the page, there is contact information: "Contact us: Email: jamesmccorkindale@gmail.com Phone No: 07482 561740" and a "Back to Top" link.

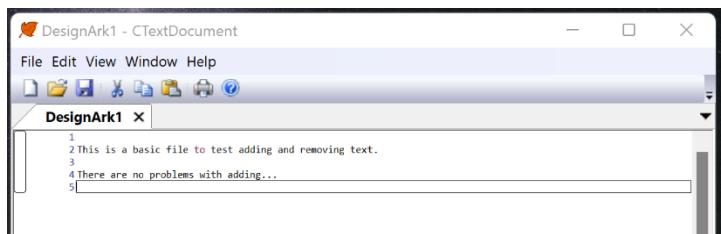
## Test Case ID 16

*About dialog. Opens and closes:*

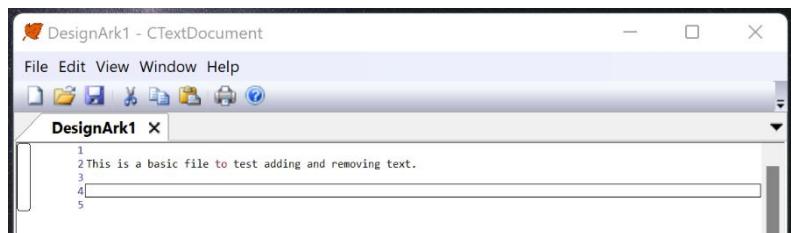


## Test Case ID 17

*Adding text:*

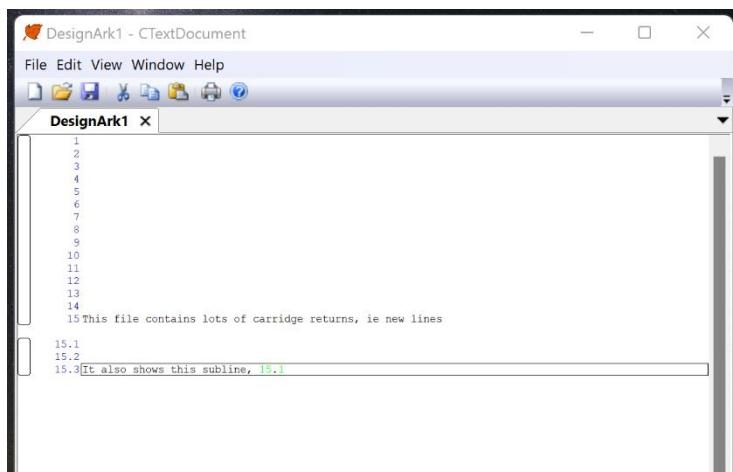


*Removing text:*

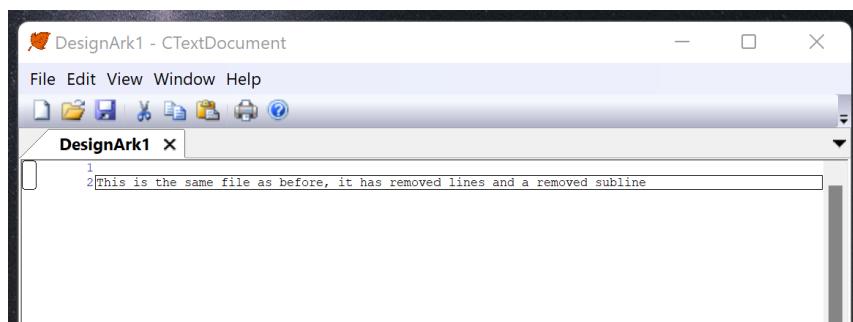


## Test Case ID 18

*New lines and subline:*



*Removed lines and deleted sublines (using the same file as before):*

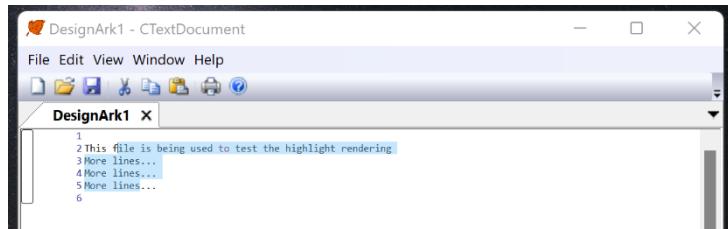


Test Case ID 19

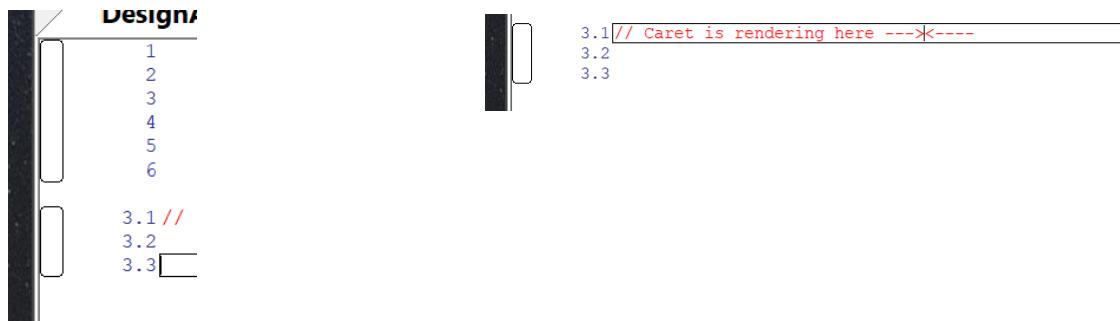
*Subline arrow button is not rendering properly:*



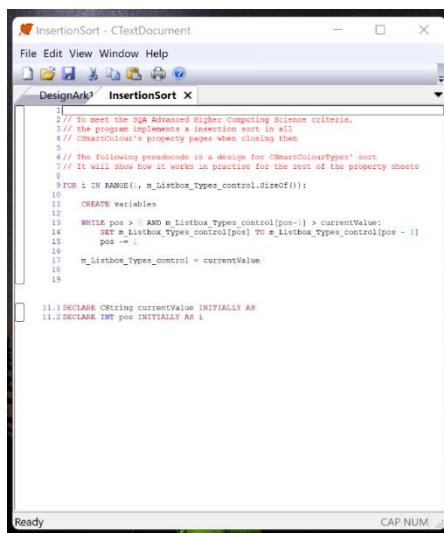
*Highlight rendering:*



*Sidebar rectangle and line numbers:      Caret renders:*



*Coloured text:*



*Cursor type changes at the appropriate place:*

*\*Cursor can not be snipped with MS Snipping Tool, but it is working.*

## Test Results

### Test Case ID 1

Application can successfully open a .txt file in the editor, that contains sublines as well as successfully creating a new, blank file.

### Test Case ID 2

Application can successfully save a document written using the editor, with the line numbers and appropriate spacing.

### Test Case ID 3

Application consistently prints the same file across different printers. It successfully exports to PDF, which matches the print pages.

### Test Case ID 4

Application can effectively change the print settings across different printers.

### Test Case ID 5

Application can close without any errors occurring.

### Test Case ID 6

Application successfully copies the highlighted text to the clipboard. From there, it is able to be pasted into another text editor.

### Test Case ID 7

Application cannot yet successfully show it can paste into itself. When pasting, it will auto-indent the text, even if the indents are already present.

### Test Case ID 8

Application can successfully open the customise window dialog box, to enable the user to customise their app.

### Test Case ID 9

Application can successfully enable and disable the toolbar

### Test Case ID 10

Application can successfully enable and disable the status bar

### Test Case ID 11

Application can successfully change its themes to match the user's selection

### Test Case ID 12

Application can successfully close tabs. However, no warning occurs if the document has not been saved. This would possibly be something to add in the future.

### Test Case ID 13

Application can successfully instantiate a new window.

#### Test Case ID 14

Application can open and close the SmartColour dialog successfully. When it closes it automatically updates the view to show the changes. Keywords, colours, fonts, and indentation sizes can all be successfully changed without issue. However, there is no validation on the indentation size input, so it is possible the user could input an extremely high value, which would give them trouble when writing.

#### Test Case ID 15

Application can successfully open the website from the menu buttons.

#### Test Case ID 16

Application can successfully open and close the about dialog box.

#### Test Case ID 17

Application can successfully add and remove text in the selected editor line.

#### Test Case ID 18

Application can successfully add and remove lines and sublines. It can remove lines via highlight and backspace, and there is validation when removing them; they can't have a subline. When lines are incremented or decremented (usually via lines above or below, being added or removed), their sublines are also incremented or decremented successfully.

#### Test Case ID 19

In terms of the GUI, the highlighter, sidebar rectangle, line numbers, caret, coloured text, and cursor all successfully render. The sidebar arrow button, that is used to create a new subline does not render correctly, when the application is not debugging. This is due to it needing access to the bitmap images, which has not been implemented.

# Evaluation

## Evaluation Report

### Fitness For Purpose

The program is fit for purpose. It meets all the end-user requirements, and the functional requirements have all been tested and work, bar a few minor bugs. The program allows the user to write pseudocode in a simple and easy to understand environment.

The user can create a new file for editing, either by opening a new app, or clicking one of the 'New' buttons in the app. The app has basic text editor functionalities like copy, paste, cut, and can highlight text. There is a slight bug when pasting text, which causes the lines to gain indentation, however, this can easily be fixed and further versions.

It allows the user to edit sublines in the same view. They work as if they're their own text editor. The sublines can be added simply by clicking a button on the sidebar, next to the parent line. Sublines can be removed, when they are empty, by clicking the backspace button. Sublines will be incremented/decremented when their parent line is also incremented/decremented. Parent lines will not be able to be removed, via highlight, and a return carriage cannot be added inside the line's text (i.e., a new line that separates the text).

The app allows the user to zoom in and out of the view, to make the text and other object larger or smaller, depending on the user's preferences and needs.

It allows the user to add new keywords to the SmartColour feature, add edit any keyword colours. The SmartColour feature also allows highlight colour, font, indentation size and comment colour customisation.

The application has printing capabilities, it can effectively print a well formatted version of a file, using the SmartColour colours. It also can export a PDF format of the document as well, which is identical to the print version.

The application allows the user to save their pseudocode files in a well formatted .txt file. It contains the line numbers and text, with the appropriate spacing. The application can also open .txt files and format them into the editor. It is currently best to only open files that have been created by the application previously, as the application will put any files that don't contain the sublines' numbers straight into the first, main editor.

One other bug to note, is the fact that the sidebar button bitmap doesn't render correctly when the app is not debugging. This doesn't affect the fitness for purpose as the application is still completely usable, but it may be harder to understand what the button does for first time users.

## Maintainability

The code is extremely maintainable. It contains meaningful variable and class names all the way through the code. This makes it very readable and understandable for anyone else that wishes to edit my project.

There is internal commentary throughout the code. It is used in every algorithm which is particularly hard to read and understand at first glance.

The code is created in modules which will allow very easy addition of new features, for example, a possible future version could contain a graph creator to start integrating all software design onto one platform. The modules also allow for easy bug fixes as every section of the code is limited to a very specific area, like a class or method.

## Robustness

The program is very robust. It allows all text to be written on the editors and new lines can be easily added and removed. The one problem might occur with the indentation size integer input, which does not have any validation and can cause unwanted indentation sizes if the user were to accidentally, or deliberately enter a large number.