CPP Beginners class – Final Project

For your final project in this course, you will create a “Dialog System” using C++.

This final project will be 50% of your final grade.

The Final Project will consist of at least 20 dialogs (2 must be conditional)

This system (module) will be comprised of several components (components can be composed of several classes):

1. Parser
2. Dialog
3. GUI (not implemented by you)
4. **Parser**: The Parser component will be used to consume different types of data and it will output the data to other components as vectors of strings.
   1. The base abstract class is called IParser
   2. The derived class is called JSONParser
   3. JSONParser will be built as a different project in the same solution as a static library. (.lib file)
   4. JSONParser will use a third party library “jsoncpp” and will be used as a wrapper for it.
   5. Examples on how to use jsoncpp can be found at <https://open-source-parsers.github.io/jsoncpp-docs/doxygen/index.html>
   6. You will have to implement

Header files can be found in the Final Project folder

1. **Dialog App:** The Dialog component is the main component.
   1. The actual **DialogBoxBase** class is part of the GUI module.
   2. The **DialogBoxBase** is consisted of many **IDialogElement** objects
   3. **IDialogElement** is an abstract class which you will build your dialogs from
   4. There are 3 abstract classes that derive from **IDialogElement:**
      1. **IDialogCreationElement**
      2. **IDialogRenderingElement**
      3. **IDialogElementButton**
   5. All concrete Dialog element classes derive from the 3 mentioned above.
   6. for example:
      1. **DialogElementTitle** derives from **IDialogRenderingElement** and will tell the dialog box what is the title of the current dialog.
      2. **DialogElementSize** derives from **IDialogCreationElement** and will tell the dialog box the size it should have.
      3. **DialogElementButton** derives from **IDialogElementButton** and sets the text and functionality of the button (currently only determines which dialog will be next).
   7. The headers for the different Dialog Elements are made for you and you need to implement them
   8. You will need to add copy constructor and copy assignment operator to all of the Dialog Element classes
   9. The abstract method **Act()** is used to perform the logics of each dialog element
   10. This Project will be compiled as an EXE and have the main.cpp file
2. **GUI:** will be consumed by your class in order to render the different dialogs (using a lib file and a dll file). You will use API calls to render your dialogs on screen

**GUI API:**

**struct GUI\_ENGINE\_API Color**

**{**

**float r{};**

**float g{};**

**float b{};**

**float a{};**

**};**

**struct GUI\_ENGINE\_API Point**

**{**

**int x{};**

**int y{};**

**};**

**GUI\_ENGINE\_API virtual void SetStartingDialog(int startingId) = 0;**

Determine which dialog will be first (id)

**GUI\_ENGINE\_API virtual void StartDialog(std::string title) = 0;**

Used to set the title

**GUI\_ENGINE\_API virtual void DisplayTextDialog(std::vector<std::string> const & text) = 0;**

Used to set the lines of the dialog.

**GUI\_ENGINE\_API virtual void SetCurrentDialog(int newCurrentDialog) = 0;**

Used to set the current dialog.

**GUI\_ENGINE\_API virtual void SetDialogPosition(Point point) = 0;**

Used to set the position of the dialog window in pixels (0,0 is top left).

**GUI\_ENGINE\_API virtual void SetDialogSize(int width, int height) = 0;**

Used to set the dialog window size in pixels.

**GUI\_ENGINE\_API virtual void SetWindowColor(Color color) = 0;**

Used to set the background color of the window.

**GUI\_ENGINE\_API virtual void SetTextColor(Color color) = 0;**

Used to set the color of the text.

**GUI\_ENGINE\_API virtual void Run() = 0;**

Used to start rendering the gui engine (call this once all dialogs have been added).

**GUI\_ENGINE\_API virtual bool ShutDown() const = 0;**

Used to shut down the gui engine (call this at the end of the main() function.