 **Weight: 15%**

Data Structures

Project 1 – Menu System

Table of Contents

[Objective 1](#_Toc286820742)

[Resources 1](#_Toc286820743)

[Architecture 2](#_Toc286820744)

[menuItem 2](#_Toc286820745)

[menu 2](#_Toc286820746)

[menuManager 2](#_Toc286820747)

[Requirements 3](#_Toc286820748)

[Milestones 3](#_Toc286820749)

[Grading Breakdown 3](#_Toc286820750)

[Standard Deductions 3](#_Toc286820751)

[Submission 4](#_Toc286820752)

# Objective

You will be responsible for implementing a working menu management system.

# Resources

Contained in the *Project1Resources* folder on the server you will find a collection of .mnu files which you will use to load the various options/submenus available to the player. The main.mnu file will always be the first menu loaded and presented to the user. Each menu file contains the name of the menu, the number of options, and the options themselves.

A menu file will look something like this:

DS Test Menu

6

MENU "Start" "play.mnu"

MENU "Options" "opt.mnu"

MENU "Help" "help.mnu"

OPTION "Data Structures Project 1B"

OPTION "Full Sail"

OPTION "Game Design and Development"

The first line of the file contains the title of the menu. In this example “DS Test Menu” is the title.

The second line of the file contains the number of items that will be loaded from the file for this menu. In this example, we will load 6 items from the file.

The remaining lines in the file will contain the menu items, one per line.

The first tag will be either the keyword OPTION or MENU.

The second tag will be the name of that item(the string we will write to the screen)

The third tag will only be found on MENU items. It will contain the name of the submenu file to load next.

# Architecture

The system will have multiple layers. The menu manager class will manage a number of menus, each menu will manage a number of selections, and each selection on a menu is itself a C++ structure.

## menuItem

Our most basic structure is the menuItem. Each menuItem stores a name, a Boolean to represent whether it is an OPTION or MENU, and finally a submenu path (only stored if the is\_submenu bool is true):

struct menuItem

{

string name;

bool is\_subMenu;

string subMenuPath;

};

## menu

When we group a number of these menuItems together we get a single menu. Each menu has a title(string), a number of menuItems(DynArray<menuItem>) and it will need some way to keep track of which menuItem is currently highlighted. Our menu class might look something like this:

class menu

{

string title;

DynArray<menuItem> choices;

unsigned int currSelection;

};

## menuManager

The final part of the system is a class that will manage any number of active menus. This should work using stack behavior.

Class menuManager

{

SLList<menu> activeMenus;

};

# Requirements

* The Up and Down arrow keys will be used to move the cursor, enter will be used to make a selection
* An exit option should be added at the bottom of every menu.
* The cursor should start at the top of each submenu
* Selecting exit from a submenu takes you back to the previous menu.
  + The cursor position should be ‘remembered’ on the previous menu
* Selecting exit from the main menu should exit the program.
* You must parse the double quotes out of the strings you read from the file
* The cursor should exhibit wraparound behavior (hitting the down arrow when the cursor is on the bottom option should move the cursor to the top option. Hitting the up arrow when the cursor is on the top option should move the cursor to the bottom option)
* You must display the title of each menu.
* Main.mnu will be the only file you should have hardcoded. All other filenames will be loaded from this file and the submenu files dynamically. I will test this project with another set of files to ensure this behavior is dynamic.
* You are required to use a SLList to keep track of the previous menus.
* You are required to ensure that the cursor moves at a reasonable speed that allows the user to actually select their desired option.
  + – 100 points if the behavior is not testable in a predictable manner because the cursor moves too fast.

# Milestones

Day 2 – Load main.mnu, parse the text, and display the main menu to the screen

Day 3 – Real-Time Keyboard input. Selection highlighting / changing.

Day 4 – Submenu entry &exit. Remembering what was highlighted on previous menus.

Day 5 – Turn-In.

# Grading Breakdown

Requirement Point value

|  |  |
| --- | --- |
| Main.mnu loaded and displayed | 20 |
| Submenus loaded dynamically and displayed | 40 |
| Exit added to each menu | 5 |
| Exit on submenu returns to previous menu, exit on main menu exits program | 20 |
| Cursor position on previous menus ‘remembered’ | 15 |

# Standard Deductions

|  |  |
| --- | --- |
| Compiler errors | 100 |
| Crash | 100 |
| Memory Leak | 50 each |
| Header protection | 20 each |
| Compiler warnings | 10 each |

# Submission

Due Date : 23:59:59 PM on Day 5

To submit the Project 1 assignment:

1. Clean and build the project in Visual Studio. ensure there are no errors or warnings.

2. Run the project in debug mode with leak detection turned on to ensure that all behavior is correct and there are no memory leaks.

3. Close visual studio.

4. Navigate to the project 1 folder (The folder that contains your Project 1 Visual Studio solution and project).

5. Delete all files and folders - except for the following:

* Your .sln file (this is the Visual Studio solution file)
* Your .vcxproj file (this is the Visual Studio project file)
* Your .cpp and .h files (these contain the code that you wrote)
* The .mnu files that you were given for test purposes.

**\*\* Any unnecessary files or folders you submit will cost you 10 points per file or folder\*\*  
\*\*Make sure you have your Windows environment set to show you hidden files and folders, or you may miss some of these\*\***

6. On your desktop, create a new folder with your name in the following format: "Last Name, First Name" - nothing else.

     Your Last Name - a comma - a single space - your First Name. Appropriate capitalization for proper names should be used.

     Examples : "Pollack, Joey"; "De La Paz, Christhian"; "Tjarks, Matthew".

7. Copy your Project 1 folder into the folder that you just created.

8. Compress the folder by right clicking on the folder with your mouse and selecting 'Send To->Compressed (Zipped) Folder'.

9. Submit the Compressed Folder.