**Assignment 1: Done By: Luke Stagno 100529375  
  
Completed Deliverable: Extension to already present Log (Transform.txt) – Added System Time**An issue I had with the Transform.txt file, which contained the position in the scene, was that it didn’t give the real-time statistic, which would make it very difficult to discern exactly when something happened in real-time during a single execution considering how one would have to add the displayed number to whenever the user ran the program, and impossible to regain and therefore use that information should the user forget when they executed the program, especially with multiple executions. Therefore, a real-time clock was added to be used alongside the time since execution, allowing the user to know when exactly an event – in this case the player’s position – occurred in both real-time and since execution.

**C++ Side:**  
To retrieve the system time, a const char\* function retrieveSystemTime() was used without parameters. In the function, 5 variables were declared: A time\_point variable to obtain the system time, a time\_t variable to store the value held by the time\_point, and a stringstream, string, and const char\* variable to convert the time\_t variable into something that would allow the information to be properly returned from the function to whatever called it.  
The stringstream variable converted the time\_t into a format that could be converted into a string variable using the .str() function, which could then be converted from a string to a const char\* with the c\_str() function. The const char\* pointer then returns its held value, the system time. This number of conversions and variables were needed since the conversion functions used could only be used with their specific inputs to output specific variable types to be used in later conversions.  
  
**C# Side:**  
To be properly and safely be used in C# scripts and therefore used directly by Unity assets, the value returned by the C++ Side function has to be converted from a char\* to an IntPtr, and called by the Update function in this format. Since this would output an integer-based address instead of the value itself and therefore be useless to the user, the value must be converted back into text using the PtrToStringAnsi() function format when actually being called by the Update function to return the pointer’s value to the original system time. The final string is added to the third variable containing the player’s position and time, as demanded by the Log’s parameters.  
  
Additionally, the system time is displayed whenever the user presses ‘t’ to activate the chat input field, by assigning a text box asset to the value of the retrieveSystemTime() function converted to text with the PtrToStringAnsi() function.  
  
**Referenced Material:**  
Using stringstream to store data like system time that can be converted to a string:  
<https://stackoverflow.com/questions/5193173/getting-cout-output-to-a-stdstring>  
  
Using <chrono> to get the system time in a preferred format:   
http://www.cplusplus.com/reference/chrono/system\_clock/now/   
  
  
**Bonus Deliverable: Dialogue System**  
**C++ Side:**Creating a dialogue system requires file IO, which required the use of the FILE data type to read the file, a char array to obtain the text file as well as a stringstream variable to act as a buffer that holds the information obtained from a specified text file, a string to convert the stringstream variable into a data type that can be converted into a const char\* pointer which is the data returned from the dialogue function. An integer variable is used to identify which line the file IO is currently on.  
An fget function, which takes 3 parameters being the char array, the size of the char array, and the file, reads through the entire file, skipping lines until it reaches the desired line number as specified in the integer variable.  
  
**C# Side:**A button ‘Dialogue Start’ calls the readLines function, applying the PtrToStringAnsi() function as in the logging system, which opens a textbox that displays the information from the readLines function. The ‘Dialogue Start’ button sets an integer variable that causes readLines to display the respective line according to the integer variable’s value. The ‘Dialogue Frwd/Bkwd’ buttons increase and decrease the integer’s variable value by 2 to either progress or regression the dialogue. The ‘Insult’ button acts as branching dialogue option, which sets the integer variable to a specific value which represents a decision by the user to bring up a separate topic than the default dialogue given by the NPC.