**Functionality Outline**

Oil Well Simulation

**Program Assignment** 1

**Prepared By**

Jaiden Gann

[jmg0053@uah.edu](mailto:Jtw0014@uah.edu)

**Prepared for**  
**Mr. James Williamson**  
**CS 307, Object Oriented Programming**  
**Computer Science Department**  
**University of Alabama in Huntsville**

**1.0 System Overview…………………………………………………………………………………………….3**

**2.0 Relevant Terms and Acronyms………………………………………………………………………..3**

**3.0 Object Functionality………….…………………………………………………………………………….3**

**3.1 OilWellSimulation………………………………………….……………………………………4**

**3.2 OilWellList…………………………………………………………..………………………………5**

**3.3 OilWell………………………………………………………………………………………………..6**

**3.4 Sensor……………………………………………………………………………………….………..7**

**3.41 HoleDepth………………………………………………………………….………...8**

**3.42 BitDepth………………………………………………………………….…….…..…8**

**3.43 RateOfPenetration………………………………………………….…….………8**

**3.44 PumpPressure……..………………………………………………..………………9**

**3.45 CasingPressure………..…………………………………………..…………..……9**

**3.46 FlowOut……………………………………………………………….………..…..…9**

**3.47 TorqueMax…………………………………………………………….……..………9**

**3.48 MudPitVolume………..……………………………………………………………10**

**3.5 Display…………………………………………………………………………………………..……10**

**1.0 System Overview**

The purpose of the program is to create a simulation of software used to monitor sensor readings on any number of Oil Rigs. The Sensor data will be constantly fed to a display unit for managers and other workers to monitor the health and progress of Oil Wells in use. Special attention will be made to make the program as object oriented as possible so that it can be expanded and changed easily and at any time.

**2.0 Relevant Terms and Acronyms**

*Rig* – Refers to a single oil well. Can be used interchangeably with “well”

*ID* – Refers to a number sequence used to Identify each well. Will succeed the name of the owning company

*Hole Depth* – Depth of the hole that has been drilled. Measured in feet (ft)

*Bit Depth* – Current depth(location) of the drill head. Measured in feet (ft)

*Crust* – Refers to the earth being drilled

*ROP* – rate of penetration. Used to indicate how fast the drill head is drilling through the crust. Measured in feet per hour (ft/hr)

*Off Bottom* – indicates the drill bit is not currently drilling and ROP is 0 ft/hr

*Pump Pressure* – Pressure in the lubricant(mud) pump. Measured in pounds per square inch (PSI)

*Casing Pressure* – Pressure in the drill bit casing itself. Measured in pounds per square inch (PSI)

*Flow out* – Mud(lubricant) flowing out of the bit casing. Measured in percentage of maximum flow (%)

*Torque Max* – Maximum torque that can be safely applied to the drill bit. Measured in kilo-foot pounds (kft-lbs)

*Mud Pit Volume* – Volume of mud(lubricant) available. Measured in barrels (BBL)

**3.0 Object Functionality**

***Main.cpp***

*void Main()*

Create OilWellSimulation object

Call OilWellSimulation->SimulationMain

Exit(0)

***OilWellSimulation.cpp***

(perhaps unfinished)*OilWellSimulation()* ---constructor

Ask user for file name to be read

Create instance of Dataparser using name input by user.

Create instance of OilWellList, passing in pointer to DataParser

Create Instance of Display

Delete instance of DataParser

~*OilWellSimulation()* ---destructor

Delete both instances of Display and OilWellList

*SimulationMain*()

Seed RandomNumberGenerator.

Initialize bool done, int hit, char ch, double thisTime, double outputTime, and a \_timeb Struct

Loop(until done Boolean reads true)

Compare current time to previous time

If half a second has past

call OilWellList->updateList()

check to see if key has been pressed

if key has been pressed,

check ch value.

If 0,

Set done = true

Break;

Else,

call DisplayMenu()

If 5 seconds have past

Call DisplayReport()

*DisplayMenu()*

Display->KeybindDisplayWellMenu(oilWellList)

*DisplayReport*()

Display->reportDisplay()

***OilWellList.cpp***

*OilWellList(dataParser\* data)*

wellCount = data->getWellCount()

Instantiate list

Declare a pointer for an oilWell

Loop (wellcount times)

Call oilWellCreator and pass in data argument

Set oilWell pointer to the created pointer.

Insert into list at end

Delete pointer

*~OilWellList()*

Deletes the list

*OilWellCreator(dataParser\* data)*

Creat pointer for OilWell

Instantiate every variable that is to be passed in to the parser well creator.

Call oilFieldDataParser::getWellData, pass in the three addresses.

Instantiate an oilWell, passing in the 3 variable addresses.

Return oilWell pointer

*ChangeDisplayStatus(char\* wellID)*

*Create pointer to oil well*

(unfinished) use the argument as a key to search list for the oil well

call member function with oil well pointer to change bool DisplayStatus in oil well

*OilWellListDisplay()*

Iterate through the List, calling each list’s display function.

*searchOilWellList(char\* well)*

uses argument as a key to search the list for the appropriate oil well node

oilWell\* *getList()*

*return m\_pHead;*

*ListUpdate()*

Iterate through the well list and call each well’s update function

***OilWell.cpp***

*OilWell*(char\* ID)

Initialize each member variable, and use ID to set m\_cpWellID

Declare all variable types to be used to store sensor info

Instantiate a list of type sensor

Make a Sensor pointer

Loop(wellcount times)

Call dataParser::getSensorData, using all the sensor variables

Set pointer to that sensor

Add it to the list

~OilWell()

Delete list of sensors

*WellUpdate()*

Make a bool variable, set to true.

Loop(while bool is true)

Call each well’s update function

Set bool to function’s return value.

Return true if at end of list

**Sensors.cpp**

**DisplayReport.cpp**

Access list of well and sensor info

printDisplay() //probably pass the lists

format output

**UserInterruption.cpp**

Import conio and stdio

Set a variable to \_kbhit() to check key press

If(key hit)

Grab the character

**TimerLoop.cpp**

import sys/types and sys/timeb and time

\_ftime(pointer to struct)

set up variables for current time and output

set up time struct

grab the start time

convert to double

set up next 5 sec interval

while() //setup internal loop

get current time

convert to double

if (next 5 sec interval)

printReport()