# CSCI 1480 University of Central Arkansas Lab 7 Out class Assignment

**Topic:** Containment - 5 Points  
**Reading:** Ch. 14

***\*Note: Include the following set of comments at the top of your source code for all assignments.***

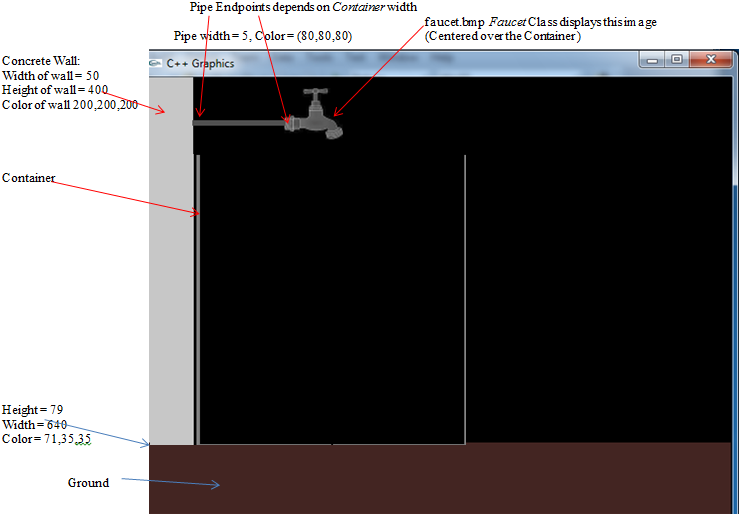
***//Your Name***

***//Assignment # (Example: Lab7Out)***

***//Brief description of the purpose of the program (Example: //Calculates the area of a square.)***

**Overview:**

Lab7Out is a continuation of Lab7In. Lab7In will be modified to fully simulate the flowing of a fluid through a faucet and subsequently filling up the Container created in lab7in. The initial scene drawn by the *Simulation* class is shown below:

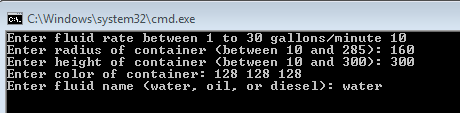


**For our simulation, assume that 1 mm = 1 pixel.**

The radius of the cylinder is a variable and is entered by the user. The height of the cylinder is a variable and is entered by the user. The cylinder is simply drawn as a cross-section – showing only the left, right, and bottom borders. The suggested thickness of the cylinder walls are 5.

Note that a pipe and faucet are now displayed over the container. The nozzle of the faucet should be approximately centered over the Container as shown.

The main function will now prompt for the radius/height/color for the Container along with the fluid type and fluid flow rate as shown below:



Note that the fluid flow rate is in gallons per minute but the dimensions of the Container are assumed to be in mm. The fluid name will determine the color of the fluid – water will be *blue* (0,0,255) oil will be a *brown* color (142,70,70), and diesel will be *purple (255,0,255)*.

The central idea of the Simulation is to illustrate that for a given radius, as the flow rate changes, the fluid will rise at a different rate – given as - for the Container.

Two additional classes are added to lab7out – *Faucet* and *Fluid*. The *Faucet* class is declared below:

#include "GenPoint.h"

#include "Fluid.h"

class Faucet

{

private:

GenPoint pos;

Fluid fluid;

public:

Faucet();

void turnOn();

void turnOff();

void setPosition(GenPoint upper);

GenPoint getPosition();

void setFluid(Fluid fluid);

Fluid getFluid();

void draw();

};

The purpose of the *Faucet* class is to center the *faucet.bmp* image over the Container and then to turn on (or turn off) the Faucet which simulates the water flowing from the Faucet into the Container. The *setPosition* method sets the x/y coordinates of *faucet.bmp* to be centered over the Container, while the *draw* method only draws the *faucet.bmp* image at the position specified by *setPosition*.

The suggested y-coordinate for the pipe 50,

The suggested y-coordinate for the faucet is 12.

The suggested equation for computing the x coordinate for displaying the faucet.bmp image approximately centered over the container is given as:

faucetx = (radius - FAUCET\_WIDTH) + 50;

Where FAUCET\_WIDTH is the width of the faucet.bmp which is 64 pixels.

The *turnOn* method of the Faucet class will draw a line from the faucet nozzle representing the stream of fluid filling up the container. The stream of fluid is colored dependent on the name of the fluid as entered by the user.

The *Fluid* class maintains information about the fluid to be dispensed by the Faucet and is declared as follows:

#include <string>

#include "GenPoint.h"

#include "Color.h"

using namespace std;

class Fluid

{

private:

GenPoint start;

GenPoint end;

int fillLine;

string fluidName;

public:

Fluid();

void setStart(GenPoint start);

void setEnd(GenPoint end);

void setFluidName(string name);

void setFillLine(int fillLine);

GenPoint getStart();

GenPoint getEnd();

int getFillLine();

Color getColor();

string getFluidName();

};

The *Fluid* class only maintains data about the fluid to be dispensed by the *Faucet* class. Note that the *Faucet* class contains a private data field pertaining to the *Fluid* class.

Once the above information has been entered by the user, the *draw* method for the *Simulation* object is invoked. The draw method will draw the Concrete Wall, the Ground and the Container just as implemented in lab7in. Additionally, the *Faucet* and the pipe connecting the Faucet to the wall will also be drawn by this method.

An example of how the Simulation should appear after its draw method is invoked is shown on page 1.

The Simulation requires the computation for - this will be the amount the fluid will rise within the Container for each second. The computation of this for a Cylindrical Container is given as follows:

Our simulation requires that water *gradually* rises within the cylinder. The speed that the water rises in the cylinder is dependent upon 2 parameters:

The radius of the cylinder

The flow rate of water (gallons/minute) entering the cylinder.

The volume of the cylinder is given by

***For each second***, the change in height with respect to the change in volume (i.e., the flow rate in gallons/min) can be given as

Since is equal to 1 second, we can rewrite the equation as

Where rate as entered by the user in gallons/min. Solving for gives equation (1):

(1)

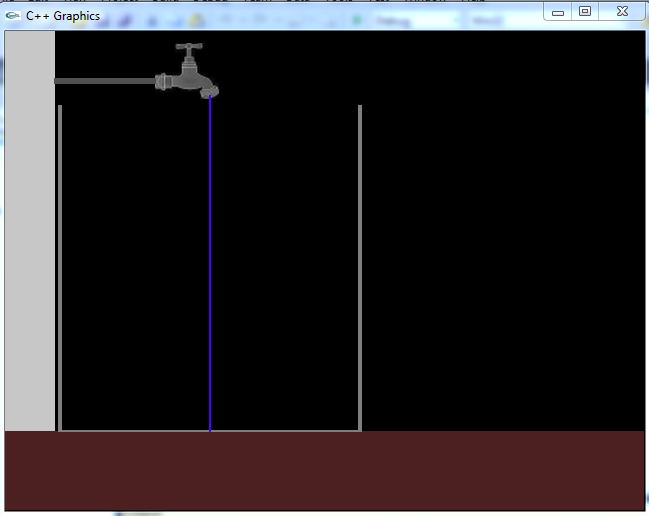
You will need to convert gallons/minute to mm3/sec using the following conversion:

mm3/sec =( 3785411.78 \*gpm)/60

Then plug this converted value in for in order to obtain .  **is how much the water level should rise (in mm) in the cylinder for 1 second**.

Once the *draw* method completes, the *start* method of the *Simulation* object will be invoked (all this is done in main) thus starting the simulation of filling the Container with a *fluid* (*water* for this example). The simulation of water filling the Container is described below:

1. The *turnOn* method of the *Faucet* class is invoked first within the *Simulation* *start* method. This method will draw a line representing a stream of fluid that is initially drawn from the faucet to the bottom of the Container as shown below. The color of the Stream is determined by the type of Fluid as entered by the user – *water* is entered for this example so the stream is drawn with a blue color. An example of how the initial simulation should look after the *turnOn* method is invoked is shown below:



1. The next step is to *gradually* fill up the Container based on the fluid flow rate as entered by the user. The appearance of the fluid rising within the Container is accomplished by drawing a rectangle the same color as the fluid that has the width of the Container and a height of as computed above in equation (1).

**For our simulation, assume that one complete iteration of the for-loop is equal to 1 second. (Of course the for-loop executes in less than a milli-second, but for simplicity it is fine to assume this for our simulation).**

**For the computed , simply draw a rectangle with this amount for each iteration. The total # of iterations should be equal to height/.**

1. **Note**: For certain flow rates (or Container dimensions) some values of **will be less than 1mm.** You will need to add logic to your for-loop to handle this situation and assure that the simulation will still work.
2. **Note**: Adding a 100ms delay (e.g., Sleep (100)) to your for-loop will provide a more realistic simulation by forcing your program run slower. A mosaic illustrating the filling of the tank with water is shown below:

|  |  |  |
| --- | --- | --- |
| \\psf\Home\Desktop\Screen Shot 2014-06-17 at 6.13.46 PM.png | \\psf\Home\Desktop\Screen Shot 2014-06-17 at 6.14.09 PM.png | \\psf\Home\Desktop\Screen Shot 2014-06-17 at 6.14.21 PM.png |
| \\psf\Home\Desktop\Screen Shot 2014-06-17 at 6.14.39 PM.png | \\psf\Home\Desktop\Screen Shot 2014-06-17 at 6.14.56 PM.png | \\psf\Home\Desktop\Screen Shot 2014-06-17 at 6.15.09 PM.png |

|  |  |  |
| --- | --- | --- |
| \\psf\Home\Desktop\Screen Shot 2014-06-17 at 6.15.45 PM.png | \\psf\Home\Desktop\Screen Shot 2014-06-17 at 6.16.01 PM.png | \\psf\Home\Desktop\Screen Shot 2014-06-17 at 6.16.15 PM.png |

Once the Container is full, invoking the *turnOff* method of the *Faucet* class results in the blue (for this example) stream to be removed. This is illustrated at the end of the simulation as shown

Have the program repeat as many times as desired.

**Class Declaration/Descriptions**

Declare GenPoint class as shown below:

class GenPoint

{

private:

int x;

int y;

public:

GenPoint();

GenPoint(int x, int y);

void setPoint(int x, int y);

int getX();

int getY();

};

Description of GenPoint follows:

**Data Fields**

int x; //X-Coordinate of the point – valid values between 0 and 639

int y; //Y-Coordinate of the point – valid values between 0 and 479

**Methods**

GenPoint();

Default Constructor – initializes x/y private data fields to 0

GenPoint(int x, int y);

Parameterized Constructor – initializes x/y data fields to the input parameters x and y.

void setPoint(int x, int y);

Setter for x/y data fields.

int getX();

Getter for x.

int getY();

Getter for y.

Color Class Declaration:

class Color

{

private:

int r;

int g;

int b;

public:

Color();

Color(int r, int g, int b);

void setColor(int r, int g, int b);

int getRed();

int getGreen();

int getBlue();

};

Color Description:

**Data Fields**

int r; //Red component of color – valid values between 0 and 255

int g; //Green component of color – valid values between 0 and 255

int b; //Blue component of color – valid values between 0 and 255

**Methods**

Color();

Default Constructor – sets all data fields to 0.

Color(int r, int g, int b);

Parameterized Constructor – sets data fields to the corresponding input parameters.

void setColor(int r, int g, int b);

Setter for all private data fields.

int getRed();

Getter for Red data field

int getGreen();

Getter for Green data field

int getBlue();

Getter for Blue data field

Declare Container Class as shown below:

#ifndef CONTAINER\_H

#define CONTAINER\_H

#include "GenPoint.h"

#include "Color.h"

class Container

{

private:

GenPoint pos;

int radius;

int height;

Color color;

public:

Container();

void setPosition(GenPoint pos);

void setRadius(int radius);

void setHeight(int height);

void setColor(Color color);

GenPoint getPosition();

int getRadius();

int getHeight();

Color getColor();

void draw();

};

#endif

Description follows:

**Data Fields**

GenPoint pos;

Upper left position of the Container as shown on page 1.

int radius;

Radius of the container

int height;

Height of the container

Color color;

Color of the Container

**Methods**

Container();

Default Constructor – sets all data fields to 0.

void setPosition(GenPoint pos);

Setter for the upper left position of the Container

void setRadius(int radius);

Setter for the radius of the Container

void setHeight(int height);

Setter for the height of Container

void setColor(Color color);

Setter for the Color of the Container

GenPoint getPosition();

Getter for the upper left position of the Container

int getRadius();

Getter for the radius

int getHeight();

Getter for the height

Color getColor();

Getter for the Color

void draw();

Draws the Container based on the upper left position indicated by pos, the radius, and the height.

Declare Simulation class as shown below:

#include "Faucet.h"

#include "Container.h"

class Simulation

{

private:

Faucet faucet;

Container container;

int waterRate;

public:

Simulation();

void setContainer(Container container);

Container getContainer();

void setFaucet(Faucet faucet);

Faucet getFaucet();

int getWaterRate();

void setWaterRate(int waterRate);

void draw();

void start();

};

Simulation Description:

**Data Fields**

Container container;

Container object

**Methods**

Simulation();

Default Constructor – sets the Container data fields to 0

void setContainer(Container container);

Setter for Container

Container getContainer();

Getter for Container

void setFaucet(Faucet faucet);

Setter for Faucet

Faucet getFaucet();

Getter for Faucet

int getWaterRate();

Getter for waterRate

void setWaterRate(int waterRate);

Setter for WaterRate

void draw();

Draws the Concrete Wall, the Ground and the Container. Also invokes the draw method of the Faucet class to draw the Faucet approximately centered over the Container. Also draws a pipe from the wall to the Faucet.

void start();

Starts the simulation. Simulation continues until the Container is full. Steps are outlined on pages 5 thru 7.

**Description for Fluid Class**

**Data Fields**

GenPoint start;

Starting coordinate for the stream of fluid – should be the positioned at the nozzle of the faucet.

GenPoint end;

Ending coordinate for the stream of fluid – should be positioned at the bottom center of the Container.

string fluidName;

A string representing either water, oil, or diesel. This is used for primarily setting the color of the fluid as the Container fills.

int fillLine;

The top y-coordinate for the container. This represents the highest point that the fluid can rise inside the container.

**Methods**

void setStart(GenPoint start);

Setter for starting coordinate of the stream of fluid

void setEnd(GenPoint end);

Setter for the ending coordinate of the stream of fluid

void setFluidName(string name);

Setter for the fluid name for the Fluid – either water, oil, or diesel.

GenPoint getStart();

Getter for starting coordinate of the stream of fluid

GenPoint getEnd();

Getter for the ending coordinate of the stream of fluid

Color getColor();

Gets the color for the stream of fluid dependent upon the fluid’s name

string getFluidName();

Getter for the fluid name.

void setFillLine(int fillLine);

Setter for the fillLine of the Container.

int getFillLine();

Getter for the fillLine of the Container.

**Description for Faucet Class**

**Data Fields**

GenPoint pos;

The upper left coordinate where the faucet.bmp image is displayed. This coordinate should result in the faucet nozzle approximately displayed over the Container.

Fluid fluid;

The fluid object

**Methods**

Faucet();

Default Constructor – initializes all data fields to 0

void turnOn();

Draws a line (representing the fluid’s stream) from the nozzle of the faucet to the bottom center of the Container.

void turnOff();

Removes the line (representing the fluid’s stream) from the nozzle of the faucet to the fillLine of the Container.

**Hint**: Set the color of the stream to Black (0,0,0) from the starting position of the fluid to the fillLine of the Container.

void setPosition(GenPoint upper);

Sets the upper left coordinate of the faucet. This position should approximately center the faucet over the Container.

GenPoint getPosition();

Getter for the upper left coordinate of the faucet.

void setFluid(Fluid fluid);

Setter for the Fluid object.

Fluid getFluid();

Getter for the Fluid object.

void draw();

Draws the Faucet so that it is approximately centered over the Container.

Guidelines for main

1. Declare Simulation, Container, GenPoint(s), Color(s), Fluid, and Faucet objects
2. Prompt the user for the Containers radius/height/color, the fluid’s flow rate, and the fluid’s name.
3. Build the Fluid object first (set all data fields appropriately)
4. Build the Faucet object (set all data fields appropriately)
5. Build the Container object (set all data fields appropriately)
6. Build the Simulation object (set all data fields appropriately)
7. Invoke the draw method of the Simulation object
8. Invoke the start method of the Simulation object
9. Allow program to repeat as many times as desired.

**What you Submit**

Submit all .h and .cpp files along with main.cpp (You can zip these files up) via Blackboard.