Midterm Report Documentation

Required Components

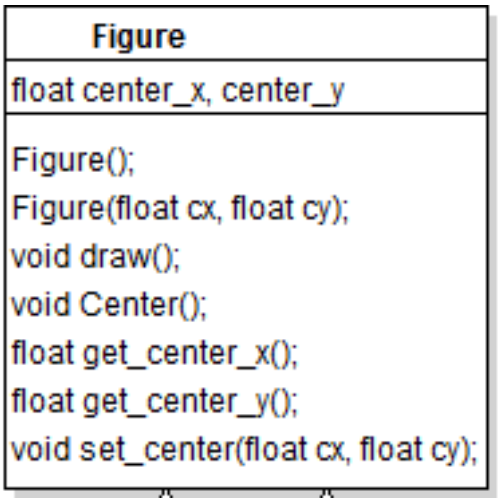
1. **<u>Please add the title of your project, your name, and peopleSoft ID in the beginning of your report</u>**
2. Describe the design of your (at least) three classes (35%)
3. Show the relationship among the classes (25%)
4. Provide Use Cases for the to-be-developed program based on the minimum requirements of the project option. (40%)

Recommended Components

- Pseudocode(s) for critical functionality of the program (can be derived from use cases and the class design)

Suggestions and examples for the design of classes

It is recommended to use the class diagram of UML (ZyBooks 11.10) to illustrate the design of each class. The following is an example of such a diagram. There are different tools that allow you to create such a diagram, including MS Word. You can also use different colors to indicate different access modifiers for different members (e.g., green for public members, red for private members, and blue for protected members). If you use the class diagram as follows, please provide the description for each member function about what it is for.



The description of the member functions of the Figure class:

Figure() // default constructor that sets the center of a figure to be (0, 0)

Figure (float, float) //overloaded constructor with two parameters to set the center of a figure

void draw(); //draw the 2D shape given its corners in order

void Center(); //Relocate the 2D shape to its current center and redraw the shape. It will call the draw() function

float get_center_x(); // return the x coordinate value

float get_center_y(); // return the y coordinate value

void set_center(float, float); //set the center of the figure based on the input two parameters

If you do not want to create the class diagram, you can simply list the member variables and member functions for each class. The following is an example.

class Figure //a base class that provides the common attributes and functions of all 2D shapes

--- member variables (protected) ---

- float center_x //to store the x coordinate of the center of a figure
- float center_y // to store the y coordinate of the center of a figure

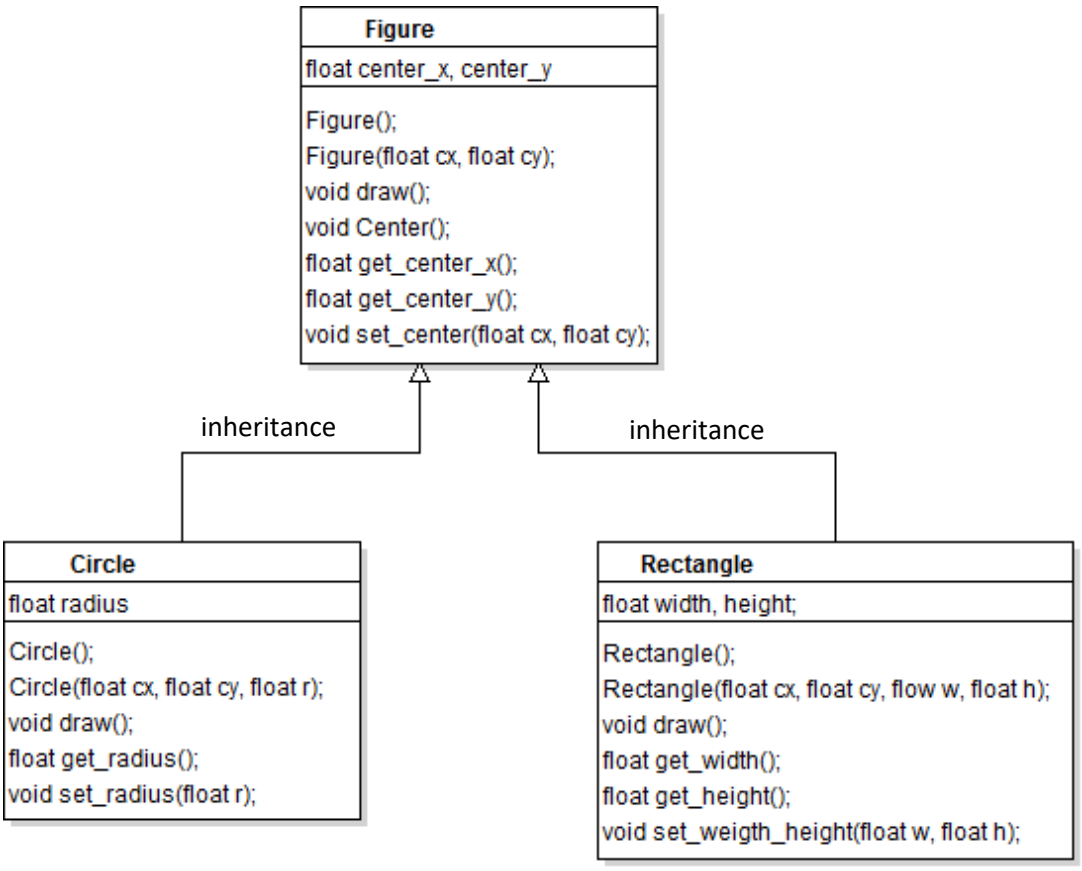
--- member functions (public) ---

- Figure(); // default constructor that sets the center of a figure to be (0, 0)
- Figure (float, float); //overloaded constructor with two parameters to set the center of a figure
- void draw(); //draw the 2D shape given its corners in order
- void Center(); //Relocate the 2D shape to its current center and redraw the shape. It will call the draw() function
- float get_center_x(); // return the x coordinate value
- float get_center_y(); // return the y coordinate value
- void set_center(float, float); //set the center of the figure based on the input two parameters

Suggestions and examples for the description of the relations among classes

Again, it is recommended to use class diagram of UML to describe the relations among different classes. Different arrows (arrow heads) represent different relations. Please refer to (https://www.visual-paradigm.com/guide/uml-unified-modeling-language/uml-class-diagram-tutorial/) for more details about the arrow heads.

Midterm Report Documentation



Please also describe why you think a class should be the derived class of another class in your design. Your argument should be logical as in the above example. In the above example, the argument is that a circle with different radii and at different centers can be described using the Circle class. It is a special example of all possible 2D shapes that the Figure class describe. Therefore, it is logical to derive the Circle class from Figure to inherit Figure's members while adding the radius attribute to the Circle class along with other functions related to the radius attribute.

Alternatively, you can use plain English to describe the relations among classes. For example, you can say

The Circle class is derived from the Figure class. This is because … [Add your argument here]

The Rectangle class is also derived from the Figure class. The reason is similar to the one for the Circle class. Specifically, … [Add your argument here]

You can describe the object composition relation similarly.

Suggestions and examples for the Use Cases of the program

Use Cases are important for you to decide and design how your program will be used for different functionality based on the user input. Typically, each functionality should correspond to one particular use case. UML provides use case diagrams. However, for this report, you can use

a more detailed step-by-step procedure to describe an individual or complex use case. This will help you think about how some functionality will be implemented. In general, the individual cases should be a complete interaction with your program that include some unique options and reflects the major components needed to complete the tasks.

Consider carefully the required interactions between your classes and the way that someone would use your program. They may start the game, load some save file, do some operations, save the file and then exit. This will inform the way you design menus and the functionality you would like to have in your program.

The following provides the description of a few use cases of a 2D graph editing program.

Use case 1: Open/Load a file for a 2D shape

The program shows a user menu

The user selects the option for loading a 2D shape from a file (e.g., by entering an option number)

The program outputs a message asking the user to enter the file name

The user enters a file name

The program attempts to load the file. If the file does not, output a message to remind the user to enter a correct name; otherwise, the program will load the 2D shape and show it on the screen for the user to manipulate

Use case 2: Move a 2D shape

The program shows a user menu

The user selects the option for editing a 2D shape

The program shows the current loaded 2D shapes

The user selects a 2D shape on screen (e.g., via mouse clicking)

Based on the pixel clicked by the mouse cursor, the program determines whether it is enclosed by a 2D shape. If it is NOT, wait for the user to select a shape; otherwise, highlight the selected 2D shape (assuming that there is no overlapping shapes). If there are multiple shapes, highlight each with a different color in their order in the list of shapes.

The user holds the left bottom of the mouse and moves the selected shape

The program updates the center of the selected shape based on the mouse movement and redraw the shape

Suggestions and examples for the pseudocode (Optional)

Midterm Report Documentation

A pseudocode helps describes the individual steps or process of an algorithm. For this project, some functionality may need to start with the design of a pseudocode before its implementation due to its complexity. However, not every functionality needs a pseudocode if it is straight forward to implement. There are some conventions of writing a pseudocode, which you will gradually learn in the later courses. The following is an example of pseudocode for the sorted insertion algorithm that we implemented (we assume the input always contains exactly 10 integers.

Name of the Procedure: Sorted Insertion

Input: 10 integers separated by whitespaces
Output: An integer array storing the 10 integers in the ascending order

```
Local variables: int array[10];
                 int i, j, pos;
Algorithm:
    for i from 0 to 10
        receive a user input integer _in

        if i==0
            array[0] ← _in
            continue to the next iteration
        else
            pos ←-1
            //find the first element in array[] whose value is larger than _in and update pos
            for j from 0 to i
                if array[j] > _in
                    pos ←j
                    break the loop
                endif
            endfor
            if pos < 0
                array[i] ← _in
            else
                //move each element from i-1 to pos one position to the right in array[]
                for j from i-1 backwardly to pos
                    array[j+1] ← array[j]
                array[pos] ← _in
            endif
        endif
    endfor
```

Note. The left-arrow symbol is assignment (we use = in code.) The variable names are all generic and need not be the ones used in the actual code. The idea is to present a level of detail on the code that shows the main steps of the program without language specifics.