Journal of Computer Science and Information Technology
June 2015, Vol. 3, No. 1, pp. 35-41
ISSN: 2334-2366 (Print), 2334-2374 (Online)
Copyright © The Author(s). All Rights Reserved.
Published by American Research Institute for Policy Development
DOI: 10.15640/jcsit.v3n1a3
URL: http://dx.doi.org/10.15640/jcsit.v3n1a3

Incremental Programming

Hieu D. Vu[1]

Abstract

Computer programming or just programming is a primary tool for programmers to write programs to solve business problems or building application software. For students who study computer science or software engineering, there is a need to have strong background in programming before they can continue in advanced studies. Through the years, computer experts, software engineers have developed many methods, techniques for developing software systems and programming. This paper present a simple technique for programming effectiveness called Incremental Programming.

I. Introduction

Computer programming is a very important aspect in Software Engineering and Computer Science. It is the primary tool for programmers to write software for an application or to solve a business problem. As computer technology improves rapidly, the demands for new applications also increases, Software Engineering has developed techniques for building large, sophisticated software systems for government and industrial applications. One of the evolutionary developing software systems is the incremental approach that was suggested by Mills (Mill *et al.*, 1980). This approach is a mean of reducing the reworks in the development process and receiving feedbacks quickly from the users, customers. This paper introduces a programming technique called "Incremental Programming" that can be used to write and test computer programs to go along with incremental development approach.

[1] Ph.D, American University of Nigeria, 98 lamidoZubairu Way, Yola by pass, P.M.B. 2250, Yola, Adamawa state, Nigeria. Email: hieu.vu@aun.edu.ng

II. Background

The first published software development model was the ‘Waterfall model’ was suggested by Winston Royce in 1970. It was divided into five principal phases: Requirements definition, System and software design, Implementation and unit testing, Integration and system testing, and Operation and maintenance. This classic ‘Waterfall model’ proved to be costly and time consuming that led to newer evolutionary software developments such as: Incremental model, Spiral model, Reuse-oriented model [1]. One of the new software development approach is the Agile method. This method became widespread, popularity and can be useful for anyone in the software development community such as: developers, team leaders, project managers, testers, and technical writers. In fact, Agile method is an umbrella term for describing several agile methodologies such as: DSDM, XP, TDD… All of them follow these principles:

1. The highest priority is to satisfy the customer through early, continuous delivery of quality software.
2. Welcome changes, even late in the development.
3. Deliver software frequently: two weeks to two months.
4. Users and developers work together throughout the project.
5. Face-to-face conversation is the best way to convey information.
6. Working software is the primary measure of the progress.
7. Pay attention to technical and good design.
8. Simplicity. The amount of work done is more important.

DSDM (Dynamic Software Development Method) was developed to provide framework for RAD (Rapid Application Development) method. DSDM has the following features:

1. User involvement
2. Iterative and incremental development
3. Increasing delivery frequency
4. Integrating tests at each phase
5. Acceptance of the product depends on fulfilling requirements

XP (Extreme Programming) emphasizes teamwork that includes managers, customers, and software developers. This method improves software development project in five areas: communication, simplicity, feedback, and courage. XP is regarded as a good general purpose method for software development and project teams can use it as it is or modify it to suit the environment and the reality [2]. XP programming emphasizes on team work that includes managers, customers, and developers. The team organized around the problem to solve it as efficiently as possible [3]. The incremental programming technique is design for writing, testing a single program by a programmer or in pair-programming environment.

III. Incremental Programming

Incremental programming is a technique that based on incremental software development concept and could make programming becomes easier and more interesting since coding and testing steps are interleaved. Basically, incremental programming relies on top-down and functional decomposition programming techniques and a "Print Result” ready function or method (or maybe not if each function contains statement(s) to print the result). This technique follows these steps:

1. Add a new function (method in Java) to the program
2. Run the new, incremented program to obtain new result
3. Make correction or adjustment as needed
4. Repeat step 1 through 4 until no-more function to write (end of program)

The other program that includes a print-result function can be done in a similar fashion.

1. Add a calling statement that invokes the corresponding function.
2. Add the function that performs the job
3. Add print statement(s) in the print-result function
4. Run the new, incremented program
5. Make correction or adjustment if necessary
6. Repeat steps 1 through 5 until the program completed

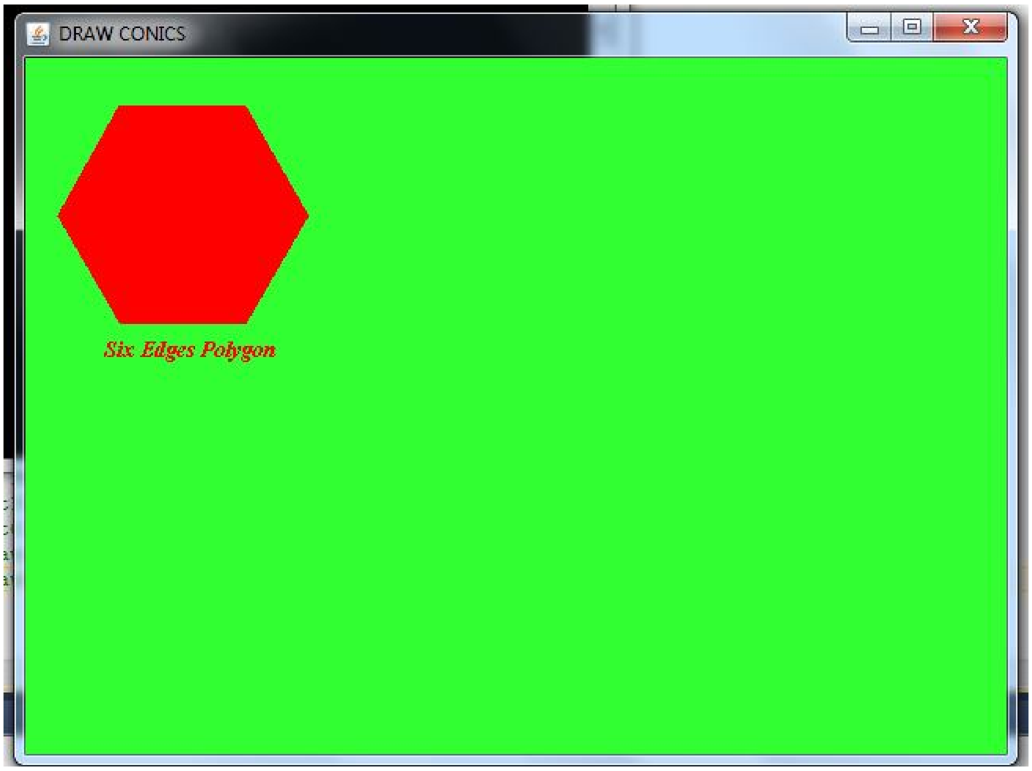
IV. Example

Draw some conic shapes. This example does not require a separate print result function, since each conic shape has its own print statements. Applying these steps above:

1. Write a new function

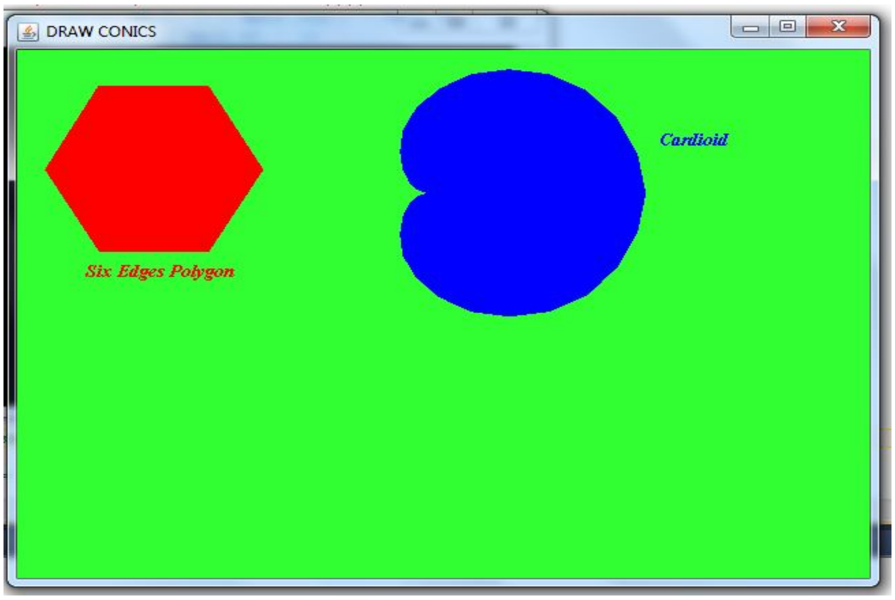
```
    //--------------------------------------------------
    //-          Method to pain the Polygon()          -
    //--------------------------------------------------
        protected void paintComponent(Graphics g){

        super.paintComponent(g);
        super.setBackground(new Color(50, 255, 50));

        int x, y;
         Polygon poly = new Polygon();   //Define polygon object

for(int i = 0; i <6; i+=1){
                    x = (int)(100 + 80 * (Math.cos(i * 2 * Math.PI / 6)));
                    y = (int)(100 + 80 * (Math.sin(i * 2 * Math.PI / 6)));

        poly.addPoint(x, y);          //Add vertices to polygon
            }
        g.setColor(Color.red);
g.fillPolygon(poly);             //Paint the polygon

         //Select font to draw string
         Font font1 = new Font("Serif", Font.BOLD + Font.ITALIC, 14);
g.setFont(font1);
g.drawString("Six Edges Polygon", 50, 190);
       }
   }
```

2. Run the program to obtain the result

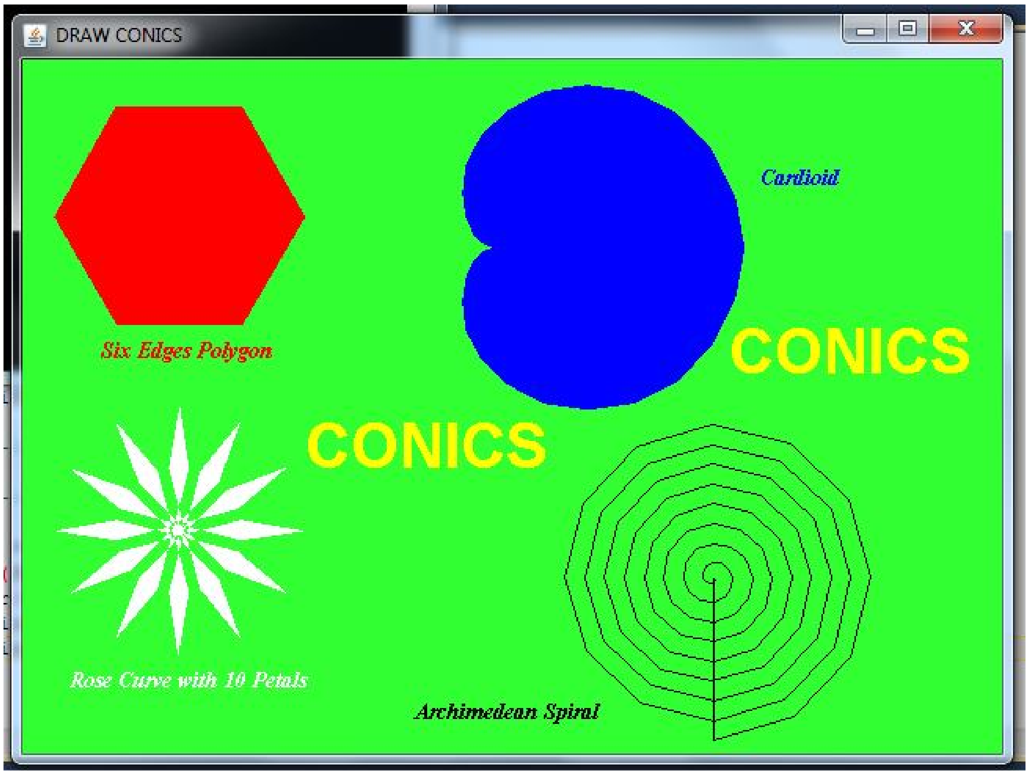


3. The result is correct, repeat the process and increment to next function by adding the next function to the end of the program (before two '}')

```
//----------------------------------------------
//- Draw Cardioid: f(theta) = K(1 + cos(theta)) -
//----------------------------------------------

Polygon cardioid = new Polygon();        //Define cardioid object

for(int i = 0; i <30; i++){
          x = (int)(300 + (80 * (1 + Math.cos(i * Math.PI / 15)) *
                              (Math.cos(i *  Math.PI / 15))));
                    y = (int)(120 + (80 * (1 + Math.cos(i * Math.PI / 15))*
                                         (Math.sin(i * Math.PI / 15))));

          cardioid.addPoint(x, y);         //Add vertices to cardioid
                    }
g.setColor(Color.blue);
g.fillPolygon(cardioid);            //Paint the cardioid
g.drawString("Cardioid", 470, 80);
```



We continue the process until no-more function to add that when the program completed. The final output should look like:



The example above illustrates programming is fun, interesting and easy to write, and debugging (make correction, adjustment). This technique also ensure the quality, correctness of the program, each function is tested, making adjustment before moving on to the next one.

V. Conclusion

Today, many professions require some types of computer programming. It becomes a required skill. One question arises “Everyone should learn programming”? This is only true within group of traditional programmers, while most people use application packages [4]. However, to be a good programmer required to learn programming languages and programming skills. This paper can be used as a guide for students, learners who want to learn computer programming.

References

Ian Sommerville (2001), Software Engineering (6e), Addison-Wesley 2001, pages45-54

http://www.codeproject.com/Articles/604417/Agile-software-development-methodologies-and-how-t 22/1/15 8:10

http://www.extremeprogramming.org/23/1/2015 2:55pm

Felleisen, Finder, Flatt, Krishnamurthi (2001), How to Design Programs, MIT Press 2001, pages xvii-xx