The Eighteenth International Symposium on Artificial Life and Robotics 2013 (AROB 18th '13),
Daejeon Convention Center, Daejeon, Korea, January 30-February 1, 2013

Non-Event Driven Graphics API for Programming Education

Kenneth J. Mackin

Tokyo University of Information Sciences, Chiba 265-8501, Japan
(Tel: 81-43-236-1329, Fax: 81-43-236-1329)

mackin@rsch.tuis.ac.jp

Abstract: Current multitasking window-based operating systems have adopted an event-driven model to support graphics and user interface control in application programs. But for beginner programmers, the idea of threads and event handling can be difficult to grasp, thus preventing beginner programmers for creating graphics applications, leaving programmers in the early stages to create simple text based console applications. In this research, a non-event driven graphics API for programming education is proposed. By applying the proposed graphics API, programmers can create graphics and real-time user interface applications by simple state-request method calls not requiring any event handling or event call-back methods. The proposed graphics API also supports text based console I/O such as print line and line input, so beginner programmers can shift from console based applications to graphics applications without any paradigm changes.

Keywords: BASIC graphics, Graphics programming, programming education

1 INTRODUCTION

After the introduction of Graphical User Interface (GUI) by Xerox PARC in 1973, computer operating systems have moved from Character User Interface (CUI) centric designs to GUI-centric designs. As GUI-centric application development became popular, GUI widget toolkits were introduced, in which 'GUI widgets', or GUI components such as buttons and text fields, are controlled through an event-driven programming model.

Current multitasking window-based operating systems, including UNIX X Windows and Microsoft Windows, have adopted this event-driven model to support graphics and user interface control in application programs. Naturally, most of current programming languages, such as C and Java, which support graphical user interfaces, use the event-driven API model.

In an event-driven programming model, an 'event' is an action that triggers some program code, such as a mouse click or a key stroke. A program registers which event triggers which program code, called event-callback or event-handler. The event-driven model allows the programmer to concentrate on only the related event for a particular code, and facilitates independent GUI design and reusable GUI widget programming. In recent programming languages such as Java, the event dispatching routine, which receives the event signal from the underlying operating system, manages the list of event handlers, and calls the appropriate event-handler for the appropriate event, is hidden altogether from the user, and GUI programmer simply programs the event-handler and waits for the event-handler to be called.

But for beginner programmers, the idea of threads and event dispatching can be difficult to grasp, leading the beginner to program something that he/she does understand what is happening. This initial barrier prevents beginner programmers from creating graphics applications, leaving programmers in the early stages of education to create simple text based console applications which are not event driven.

Text based console applications have limited value in current GUI-centric operating systems, so there is a natural need for an intermediate step between text-based programming and event-based GUI programming to ease the jump between the two paradigms. Also, a GUI program can be much more useful than a text-based program and closer to 'real' applications in a GUI operating system environment, which can greatly improve the motivation and satisfaction of the beginner programmer during programming education.

In order to overcome this problem, a non-event driven graphics application programming interface (API) for programming education is proposed in this research. By applying the proposed graphics API, programmers can create graphics and real-time user interface applications by simple state request method calls not requiring any event handling or event call-back methods. The proposed graphics API also supports text based console I/O such as print line and line input, so beginner programmers can shift from console based applications to graphics applications without any paradigm changes and minimal code changes.

2 NON-EVENT DRIVEN GRAPHICS API

The proposed non-event driven graphics API, named Basic Graphics, is implemented as a Java language class. Basic Graphics receives its design hints from 1980's BASIC language graphics approach. In particular, many of the graphics commands are based on F-BASIC by Fujitsu Limited.

Basic Graphics consists of only 1 public class, jp.ac.tuis.basic.BasicGraphics class. Basic Graphics class consists of methods to draw text, graphics and images, retrieve line input from the keyboard, and retrieve keyboard and mouse states. Additionally Basic Graphics includes features to play music and beep, as well as create network connections with other Basic Graphics applications and send and receive network data. Basic Graphics applications can be run as a stand-alone Java application or a Java Applet.

Example 1 below is an example of a hello world program written in Basic Graphics.

```
//Example 1
import jp.ac.tuis.basic.*;
public class HelloWorld{
  public static void main(String[] args){
    BasicGraphics g = new BasicGraphics();
    g.println("Hello, world!");
  }
}
```

Example 1 will create a window and print "Hello, world!" on the top of the created window.

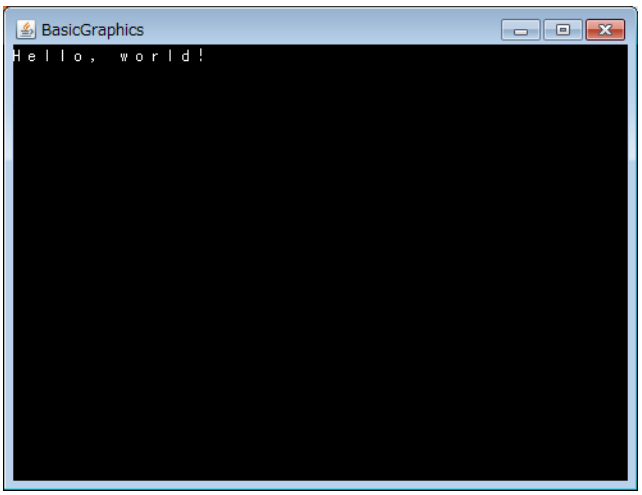

Fig. 1. Hello world program using Basic Graphics

Example 2 below is an example of a simple animation program.

```
//Example 2
import jp.ac.tuis.basic.*;
public class Animation{
  public static void main(String[] args){
    BasicGraphics g = new BasicGraphics();
    for(int i=0; i<40; i++){
      g.locate(i,10);
      g.print('X');
      g.sleep(200);
      g.locate(i,10);
      g.print(' ');
    }
  }
}
```

The locate(x,y) method in Example 2 defines the x,y coordinates of the location of the following print() method. The program will wait further execution in the sleep() method for the given milliseconds. In this example, an 'X' will move from the left of the screen to the right.

Example 3 below is an example of a program to retrieve keyboard input.

```
//Example 3
import jp.ac.tuis.basic.*;
public class Type{
  public static void main(String[] args){
    BasicGraphics g = new BasicGraphics();
    while(true){
      char key = g.inkey();
      if(key != 0){
        g.print(key);
      }
    }
  }
}
```

The inkey() method returns the key of the pressed keyboard key. If no key is pressed, then inkey() returns 0 (zero).

By using the proposed Basic Graphics API, the programmer can draw graphics and retrieve key input events (and similarly mouse events) without using the event-driven model. The proposed model relies only on the simple main method architecture of console based

application programming which the programmer is familiar with. Example 3 is also a simplified model of a event dispatcher, and can be built upon to describe the implementation of a event dispatching loop later in the programming education.

3 EXPERIMENTAL USE IN PROGRAMMING EDUCATION

The proposed Basic Graphics API was introduced to 10 2rd year college students who have had 1 year prior Java programming education, but could not understand event handling and were not able to create GUI applications yet. After a 30 minutes lecture on how to use the Basic Graphics API, the students were given a graphical programming assignment (Fig.2) to complete. All participating students were able to complete the assignments within 60 minutes. Some of the students were able to create individual graphical applications on their own (Fig.3,4). The participating students are gave positive feedback on the usefulness of the Basic Graphics API, it was clear that the motivation and satisfaction of the students had been improved.

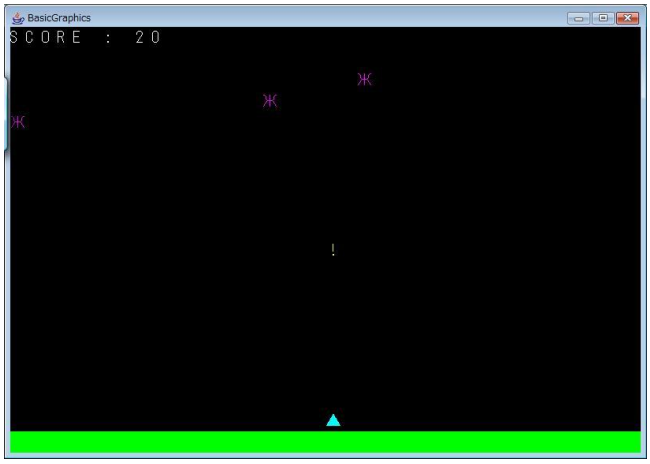

Fig. 2. Graphical programming assignment

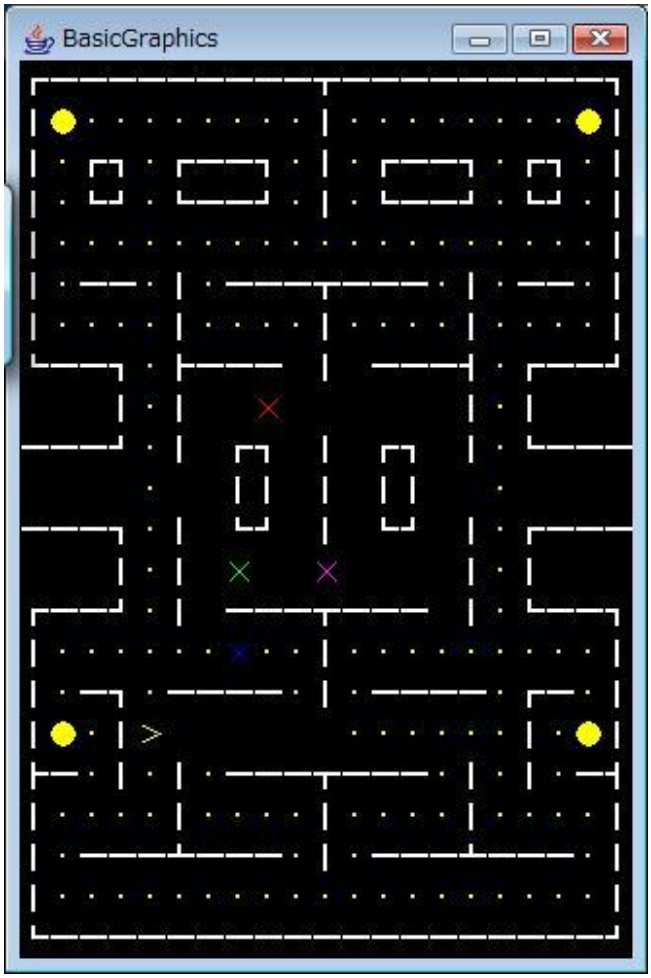

Fig. 3. Example graphical application of students

Fig. 4. Example graphical application by student

4 CONCLUSION

In this research, a non-event driven graphics API for programming education is proposed. By applying the proposed graphics API, programmers can create graphics and real-time user interface applications by simple state request method calls not requiring any event handling or event call-back methods. The proposed graphics API also supports text based console I/O such as print line and line input, so beginner programmers can shift from console based applications to graphics applications without any paradigm changes.

The proposed API was introduced to 2nd year college students with text-based console programming experience to verify the effectiveness of the proposed method. All participating students were able to complete the graphical application assignment in the total 90 minutes period, and the motivation and satisfaction of the students had been improved.

For future works, a complete self-learning tutorial for using the Basic Graphics API will be developed, to allow students to use the training material at their own learning pace.

REFERENCES

[1] Ephraim P. Glinert (ed) (1990), Visual Programming Environments: Paradigms and Systems. IEEE Computer Society Press