Object-Oriented Programming with Jeroo in the Information Technology Classroom

Dean Sanders
sanders@mail.nwmissouri.edu
Computer Science/Information Systems
Northwest Missouri State University
Maryville, Missouri 64468, U.S.A.

Brian Dorn
dorn@cs.iastate.edu
Department of Computer Science
Iowa State University
Ames, Iowa 50011, U.S.A.

Abstract

Jeroo is a tool that helps novice programmers learn fundamental concepts of object-oriented programming. Specifically, Jeroo focuses on objects, methods, and fundamental control structures. The tool is a self-contained environment in which students write and execute programs to control the actions of Jeroos and their interactions with their environment. Simple animation and source code highlighting aid comprehension.

Objective data show that the use of Jeroo levels the playing field between males and females with respect to confidence and comfort levels in the course. Other data show reduced withdrawal rates in courses that use Jeroo. Observations and anecdotes indicate that using Jeroo helps maintain student interest and helps encourage experimentation among both novice and experienced programmers.

Jeroo is available at www.jeroo.org.

Keywords: object-oriented programming, objects first, introductory programming, pedagogy. microworld

1. INTRODUCTION

Computer programming has always been difficult to teach. The difficulty is exacerbated by a mismatch between the students' expectations and the reality of many courses. The students live in a world of multimedia and graphical user interfaces, but many textbooks, examples, and ancillary materials are rooted in text-based interactions. By making it relatively easy to construct a graphical user interface, VisualBasic has helped bridge the gap between the students' expectations and the reality of the courses. Unfortunately, the move to VisualBasic.NET, with its object-oriented programming style, has made computer programming more intimidating. Many students lose confidence in the early weeks of their first exposure to object-oriented programming, partly because many of the concepts are abstract and difficult to visualize or understand. What can we do to help our students master fundamental concepts of object-oriented programming, to capture their interest, and to reduce their anxiety about computer programming? A partial answer comes in the following recommendation "To help meet the challenge of contemporary technological change, we recom-

mend: (1) that business students be taught the fundamental principles of object-oriented systems; … and (3) that these students should be allowed to experience object orientation by active participation in an immersive object-oriented environment" (Towell, 2000). Jeroo provides this kind of environment.

Objective data from several course sections indicate that the use of Jeroo reduces the students' anxiety levels, reduces the withdrawal rates, and levels the playing field for males and females with respect to confidence and comfort in the course. Anecdotal evidence suggests that the use of Jeroo increases the students' interest, encourages experimentation, and helps them master important concepts faster than they did when Jeroo was not used.

2. OVERVIEW OF JEROO

Jeroo is both an engaging virtual animal and an integrated development environment. Comments and suggestions from many students and instructors helped give Jeroo its current form as it evolved from pencil-and-paper activities in the early 1980's to a Pascal-like tool named Jessica to the current object-oriented tool. Throughout this paper, the word Jeroo will refer to either the animal or the development environment, depending on the context.

The integrated development environment and microworld was designed to help novice programmers learn fundamental concepts of object-oriented programming. Specifically, Jeroo focuses on objects, methods, and fundamental control structures. The tool is a self-contained environment, in which students write and execute programs to control the actions of Jeroos and their interactions with their environment. Simple animation and code highlighting aid comprehension.

Jeroo is reminiscent of Karel the Robot (Pattis 1995) and its descendants (Bergin 1997,2002; Buck 2001), but it provides a smoother transition to other object-oriented environments, and appears to appeal to a broader range of students. The authors know of no other teaching tool of this type that supports a VisualBasic.Net style of programming.

Jeroo is suitable for several kinds of courses. Its apparent simplicity makes it suitable for computer literacy courses, but its intellectual content makes it an effective way to introduce important concepts in a computer programming course. Currently, Jeroo is being used in three different settings: beginning programming courses, computer literacy courses, and high school courses. Written in Java and tested under Windows, Mac OS-X, Linux, and Solaris, Jeroo it suitable for a wide range of laboratory environments. Jeroo was designed for novice programmers, but students who have completed a prior programming course also find to be Jeroo interesting and worthwhile.

There are four significant aspects of the Jeroo environment, the metaphor, the programming languages, the user interface, and the runtime behavior. These work together to create an effective teaching/learning environment.

The Metaphor

A Jeroo is a rare kangaroo-like animal living on Santong Island, a remote speck of land in the South Pacific. Jeroos have an unusual way of moving about their island; they can only hop in the four main compass directions. The island is also home to the large Winsum flowers that constitute the primary food source for the Jeroos. A Jeroo can pick flowers, carry them, plant them, and give them to an adjacent Jeroo. The Jeroos must be on the lookout for nets that have been set by a trapper who is seeking new specimens for a zoo. Fortunately, the Jeroos can detect nearby nets and can disable them by tossing flowers. Frequent rainstorms and erratic tides can produce inland bodies of water and even alter the island's coastline. Despite living on the island for untold millennia, the Jeroos are poor swimmers and must avoid all bodies of water

The metaphor changed dramatically as Jeroo evolved from its origins in the early 1980's. Positive and negative comments from students helped define the current non-technical, non-violent metaphor – a metaphor that is comfortable and easy to learn. All students react well to this metaphor.

Jeroo's Programming Languages

Students write programs to control the actions of up to four Jeroos and their interac-

tions with their immediate surroundings. Jeroo supports two different language styles, VB-style and Java-style. The former is a true subset of VisualBasic.Net. The latter mirrors the common subset of Java, C++, and C# except that a generic header is used for methods in the Java-style language. Either language style allows a teacher to say "The Jeroo language is really a subset of VisualBasic.Net (or Java or C++ or C#)", thereby avoiding the question "Why are we studying Jeroo instead of a 'real' programming language?"

Only one class is available within the language – the Jeroo class. Each Jeroo object has three attributes: its location, its direction, and the number of flowers in its pouch. These attributes, which define the state of a Jeroo, are always visible in the user interface. The Jeroo class has six constructors that allow a programmer to override default values for the attributes. These constructors provide a gentle introduction to the use of arguments and the concept of overloading.

There are no data types and no variables other than references to Jeroo objects. There are, however, eight predefined constants to indicate relative or absolute directions. These constants are used as arguments to predefined methods.

The Jeroo class has fourteen predefined methods: seven sensor (or Boolean) methods and seven action methods. The sensor methods allow a Jeroo to inspect its immediate surroundings. The action methods allow the Jeroos to move about the island, pick and plant flowers, give flowers to one another, and disable nets. A programmer can define additional action methods to extend the behavior of the Jeroos. The programmer-defined methods lack formal parameters, but they can invoke other methods and they can be recursive. The ability to write additional methods is an effective way to introduce modularity at an early stage.

The Jeroo language supports three fundamental control structures: *while*, *if*, and *if-else*. The VB-style language also supports *elseif*. The conditions for these structures are constructed from one or more sensor (Boolean) methods and the operators AND, OR, and NOT (&&, ||, and ! in the Java-style)

Figures 1 and 2 below show the same small Jeroo program in both language styles.

Figure 1
A VB.NET-Style Program

```
'User Methods
Sub turnAround()
  turn(RIGHT)
  turn(RIGHT)
End Sub

'Main Method
Sub main()
  Dim kim as Jeroo = new Jeroo(0,5)
  While (NOT kim.isWater(AHEAD))
    kim.hop()
  End While
  Kim.turnAround()
End Sub
```

Figure 2
A Java-Style Program

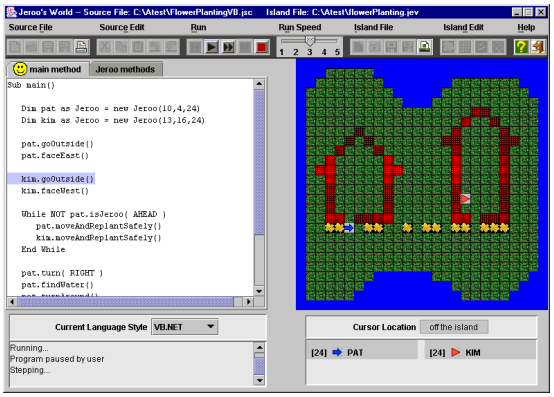
```
//User Methods
method turnAround()
{
  turn(RIGHT);
  turn(RIGHT);
}

//Main Method
method main()
{
  Jeroo kim = new Jeroo(0,5);
  while (! kim.isWater(AHEAD))
    kim.hop();
  kim.turnAround();
}
```

The User Interface

The user interface consists of a single window in which all components are visible at all times, making it easier to associate a program with its execution. Menus and a toolbar provide a familiar interface that the students can use with virtually no instruction. All features of Jeroo are accessible via menus, all except those that require a subsequent mouse action are available via the keyboard, and the most commonly used ones are accessible through a toolbar. The buttons on the toolbar are organized into logical groups, and the corresponding menus are positioned so they can serve as labels for the button groups. Figure 3 shows the

appearance of the user interface for a running program.

Figure 3
The User Interface for Jeroo



The left-hand portion of the window (A) consists of two tabbed panes for editing the source code. One tab is used for a main method in which Jeroos are instantiated and used at a high level to solve a specific problem. The other tab, marked "Jeroo methods," is used to create programmer-defined methods that extend the behavior of all Jeroos. The source code editor supports common editing features such as cut, copy, paste, undo, and redo. The editor also supports block commenting, block indenting, an optional autoindent, and the ability to change the size and style of the editor's font.

A language style box appears immediately below the source code editing panes. This box shows the current language style, and allows the programmer to switch to a different style for the current program.

The right-hand portion of the window (B) is a graphical representation of Santong Island. Using a point-and-click process, students can alter the shape of the island and can place flowers, nets, and water features on the island. This area also displays a simple animation of a running program.
A cursor location panel appears immediately below the island. This panel keeps track of the location of the cursor when it's over the island, thereby making it easier to locate specific cells on the island.

The interface also includes two status panels. The message area (C) displays status messages, syntax error messages during compilation, and errors that occur at runtime. Each error message is accompanied by a highlighting of the corresponding line of code. Keeping the novice programmers in mind, great care was taken to make the error messages as informative as possible. Typical messages are show in table 1 for a Jeroo named Kim.

Table 1
Typical Error Messages

Intended Code: Kim.hop() Actual Code: Kimm.hop() ------ Message ------ SYNTAX ERROR: Unknown identifier "Kimm" Check spelling and capitalization.
Intended Code: Kim.turn(LEFT) Actual Code: Kim.turn(LEFR) ------ Message ------ SYNTAX ERROR: Relative direction expected Found "LEFR" instead.
Intended Code: 'Kim avoids a net Actual Code: 'Kim hops into a net ------ Message ------ LOGIC ERROR: "Kim" is trapped in a net

During program execution, the Jeroo status panel (D) is continually updated to display the state of each Jeroo in the program. Beyond identifier names, this panel displays the direction of motion and number of flowers carried by each Jeroo. The third part of the Jeroo's state, its location, can be determined by viewing the island. After a runtime error, this panel shows the state of every Jeroo just prior to the error.

The "Help" menu includes a context sensitive language reference that explains the grammar of each language. The current language style is used to access the appropriate help files so that students are only presented with information that is pertinent to their program. These files are displayed in a separate window, allowing students to view references and source code side-by-side.

The Runtime Behavior
The environment comes alive when the runtime module is invoked using one of the "Run" buttons. First, the source code is automatically saved, and the language translation (compiler) sub-system examines the source code, reporting any syntax violations with custom tailored error messages. The offending line of code is also highlighted.

Once a program has no syntax errors, the visual execution (interpreter) sub-system generates a simple animation with simultaneous code highlighting that allows a student to visualize the connection between source code and the program's behavior. Students can change speeds or switch back and forth between stepwise and continuous execution while a program is running. As discussed earlier, the Jeroo status panel (region D in figure 3) shows the current state of every Jeroo.

Source code highlighting, animation, and status information create a rich learning environment. The user interface and animation are not only intuitive but also visually appealing enough to give the hint of a video game—engaging students and holding their attention longer than most programming environments and programming assignments. More importantly, the interface allows students to visualize the semantics of control structures and grasp interactions between methods.

3. EDUCATIONAL GOALS FOR JEROO

Four goals provided an overall context for developing the Jeroo software:

- engage students immediately,
- generate confidence in students,
- improve comprehension of specific topics, and
- develop crucial programming skills.

Three aspects of Jeroo help to engage students early: the metaphor, the design of the user interface, and the animated execution of programs. The animated execution gives Jeroo a feel reminiscent of a video game.

Confidence is built through the anthropomorphic nature of the Jeroo objects, the immediate feedback given by the animation and simultaneous code highlighting, and the ability to run programs stepwise or continuously at different speeds. These three features combine to give the students a strong visual confirmation when a problem has been solved, an ability to detect errors quickly, and a feeling of control over their use of Jeroo.

Jeroo is not intended to replace a general-purpose programming language. Instead, its scope is narrow, focusing on five specific topics: instantiating and using objects, sending messages to objects, writing methods to extend the behavior of objects, using selection and repetition structures, and writing Boolean expressions. The narrow and sharply focused scope allows students to master these concepts without the distraction of additional language features.

Finally, Jeroo was designed to help students develop crucial programming skills such as problem decomposition, solution composition, incremental development, and the design of test cases. Jeroo's anthropomorphic nature provides a strong frame of reference for decomposing problems and composing solutions. The visual feedback from a running program together with the ease of switching between coding and executing encourages incremental development. The visual nature of Jeroo makes it easier for the students to learn the principles of software testing.

Has Jeroo met these goals? Our classroom experiences, as well as those reported by others, indicate that each of these goals has, indeed, been met.

4. JEROO IN THE CURRICULUM

Jeroo in IS 2002.5

Jeroo can be particularly useful in a course such as IS 2002.5 "Program, Data, File, and Object Structures, as defined in the IS 2002 Model Curriculum. One purpose of this very full course is to introduce the students to the capabilities of a number of programming languages. Experience has shown that using Jeroo is an effective and time-efficient way to introduce several important programming concepts of both procedural and object-oriented languages.

Jeroo In A Programming Course

Using Jeroo (or similar tool) exclusively at the beginning of a course is a common practice. At the end of this unit, the students will understand the semantics of basic control structures, will be comfortable with the concept of instantiating objects and sending them methods, and will be able to design, implement, and use methods that define the behavior of a class of objects. These basic concepts can be revisited and extended after the transition to the primary language for

the course. Unfortunately, the transition to the primary language can be problematic. One problem is a significant reduction in the level of abstraction. At the end of the Jeroo unit, the students are writing programs that require significant modularization, thoughtful design, and several Jeroo objects. Now, they are working with a single method, *main*, and writing simple programs such as "convert dollars to euros". To the students, this appears to be a step backward. A second major problem is the interval between the time a topic is covered in Jeroo and the time it is revisited in the primary language. Even when the ties to Jeroo are exposed, there is a disconnection between Jeroo topics and the same topics in the primary language.

Interleaving Jeroo topics with related topics in the primary language alleviates the difficulties associated with covering all of Jeroo as a single unit. We recommend using Jeroo in four separate units. In each unit, Jeroo is used to introduce concepts that are immediately revisited and expanded in the primary language. The first unit introduces the concepts of algorithms and three programming topics: the role of the main method, object instantiation, and method invocation. In a VisualBasic.Net course, this is an excellent time to introduce the students to the designer, have them prototype a simple interface, and examine the generated code to show the similarities to Jeroo. The second unit uses Jeroo to introduce programmer-defined methods. This is followed by learning how to write call-back methods in VisualBasic.Net. The third and fourth units use Jeroo to introduce repetition structures and selection structures, respectively. Each of these is followed by coverage of the corresponding topics in the primary language for the course.

By interleaving the use of Jeroo with coverage of the corresponding topics in the primary language, the students see the relevance of Jeroo and appear to have a better understanding of those topics. As one student said "Everything goes back to Jeroo!" Concerns about having the students switch back and forth between Jeroo and another development environment have proven to be unfounded. They seem to have no trouble moving between environments.

5. PERCEIVED BENEFITS

The benefits of working with Jeroo can be organized into three broad categories: programming concepts, programming practices, and student satisfaction. The benefits pertaining to programming concepts and practices are derived from faculty observations, but student satisfaction is based on objective data.

Instructors who use Jeroo feel that students have a better understanding of objects, methods, and control structures. Moreover, they grasp these concepts faster than when Jeroo is not used. We have also noticed that students have developed a more mature style of programming than in the past. By the end of the semester, most students have begun to decompose problems and develop algorithms before they start to write code. They also tend to use diagrams and other visualization techniques to help understand problems and to explain them to others. We attribute the improvement in decomposition and algorithm development to the anthropomorphic nature of Jeroo.

Student satisfaction can be inferred from three sources: survey results, withdrawal rates, and comments. Student opinions were solicited via a questionnaire that was given to 97 students at Northwest Missouri State University. The questionnaire was given in five sections of the CS-1 course covering two semesters and three instructors. The survey solicited demographic information and asked the students to respond to questions on a five-point Lickert scale (1=low, 5=high).

One pair of questions asked the students to rate their confidence in their ability to learn computer programming, both before the course started and after the end of the Jeroo portion. The results are summarized in the graph in figure 4. The apparent increase in confidence is statistically significant. Using a paired t-test, the probability that there was no significant increase in confidence levels is less than 2.0×10^{-8}. The dramatic difference between males and females is evident in the graph.

Figure 4
Increase in Confidence Level

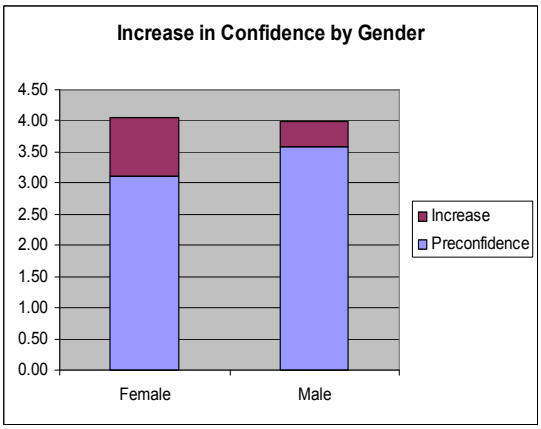
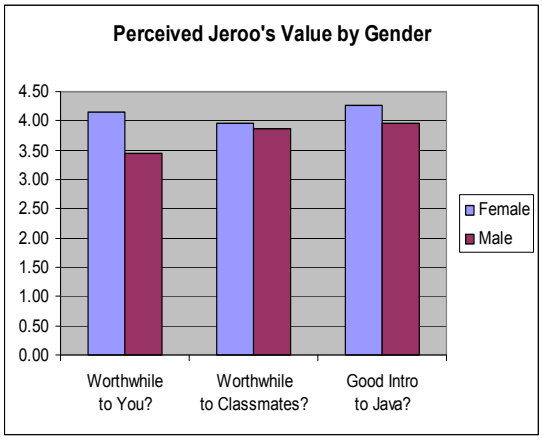


Figure 5
Jeroo's Perceived Value



A recent study (Beyer 2003) indicates that females have lower confidence than males in their computing abilities, even when controlling for quantitative ability. This is significant because many computing degrees, whether computer science, software engineering, information systems, or information technology, require some form of computer programming early in the curriculum, thereby making computer programming an early filter. If Jeroo can level the playing field with respect to confidence and comfort in the course, it may lead to an increase in the number of women who choose some form of computing as their major field.

A second pair of questions asked the students to rate their level of comfort about the course, both before the course started and after the end of the Jeroo portion of the course. A paired t-test shows that the probability that there was no significant increase in comfort levels is less than 1.4×10^{-8}. The pre- and post-comfort levels for males and females are slightly higher than the pre- and post-confidence levels, but the pattern of increase is the same as that shown in figure 4.

Finally, the students were asked to indicate whether or not they felt that Jeroo provided a good introduction to Java and to rate the value of Jeroo to themselves and their classmates. The responses to these questions are summarized in the graph shown in figure 5. As is evident from the graph, the female students rated Jeroo slightly higher than did the males.

Lower withdrawal rates and higher continuation rates are a second indication of student satisfaction. Jeroo is being used at the University of Wisconsin-Parkside (UW-P) in CSCI-130, "Introduction to Programming". This is a one-hour preparatory course for students who are not yet ready for the first programming course. Data for this course show a withdrawal rate of 19% in the four terms preceding the introduction of Jeroo, and a withdrawal rate of 0% in the three terms since the course switched to Jeroo. Data from the same terms show that the percentage of those who continued on to CS-1 increased from 28% to 35%.

Student satisfaction can also be measured by student behavior and comments. The authors have observed that many students, both novices and experiences students, like to write their own scenarios and corresponding programs. These range from programs that plant flowers in complex patterns to maze traversal programs. The enthusiasm is refreshing!

The students make many positive comments about Jeroo. The following student comment is typical.

> *Jeroo helped me tremendously. I took CSCI-241 (the CS-1 course) first without any prior programming experience and was lost from the first day. Taking the same class after experience with Jeroo is much easier to understand. I think Jeroo is a good way to ease into programming with Java. For me, it*

made the whole idea of instantiable (sic) classes and main methods a much simpler concept. Also, I think Jeroo is a good example of Object-Oriented programming which seemed to be a hard concept for me to grasp.

6. SUMMARY

Jeroo has proven to be an effective tool for teaching the concepts of objects, methods, and control structures to novice programmers. The metaphor and animated execution aid understanding and help generate interest. There is some evidence that the use of Jeroo leads to reduced withdrawal rates and increased self-confidence among the students. The increase in self-confidence is particularly dramatic among the female students. The syntax of Jeroo's programming language facilitates a smooth transition from Jeroo to VisualBasic.Net, Java, C++, or C#.

The uses of Jeroo are limited only by the instructor's imagination. Appendix A contains two creative examples that have been contributed by teachers who use Jeroo. Jeroo is available at www.jeroo.org.

7. REFERENCES

Bergin, Joe, Mark Stehlik, Jim Roberts, and Richard Pattis, 1997, Karel++: A Gentle Introduction to the Art of Object-Oriented Programming. John Wiley & Sons, New York.

Bergin, Joe, Mark Stehlik, Jim Roberts, and Richard Pattis, 2002, "Karel J. Robot: A Gentle Introduction to the Art of Object-Oriented Programming in Java." Retrieved September 27, 2004, from: http://csis.pace.edu/%7Ebergin/KarelJava2ed/Karel++JavaEdition.html.

Beyer, Sylvia, Krisina Rynes, Julia Perrault, Kelly Hay , and Susan Haller, 2003, "Gender Differences in Computer Science Students." SIGCSE Bulletin, vol. 35 no. 1, pp. 49-53.

Buck, Duane and David Stucki, 2001, "JKarelRobot: A Case Study in Supporting Levels of Cognitive Development in the Computer Science Curriculum." SIGCSE Bulletin, vol. 33 no. 1, pp. 16-20.

Dorn, Brian and Dean Sanders, 2003, "Using Jeroo to Introduce Object-Oriented Programming,." FIE 2003 Conference Proceedings, (CD Version) IEEE Catalog Number 03CH37487C.

Pattis, Richard, Jim Roberts and Mark Stehlik, 1995, Karel The Robot: A Gentle Introduction to the Art of Programming, 2nd ed. John Wiley & Sons, New York.

Sanders, Dean and Brian Dorn, 2003, "Jeroo: A Tool for Teaching Object-Oriented Programming." SIGCSE Bulletin, vol. 35 no. 1, pp. 201-204.

Towell, John, 2000, "MOO: An Active-Learning Environment for Teaching Object-Oriented Concepts In Business Information Systems Curricula." Journal of Information Systems Education, vol. 11, no. 3-4 Retrieved September 27, 2004, from: http://www.jise.appstate.edu/11/147.pdf.

APPENDIX A – SAMPLE PROGRAMS

LOOK HOMEWARD ANGEL: A BEGINNING ACTIVITY

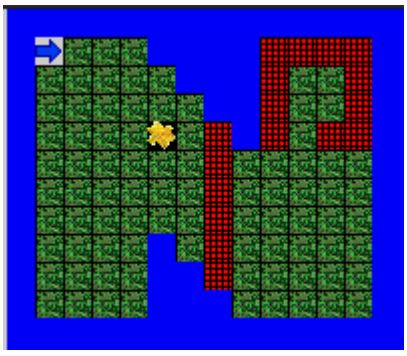
Acknowledgement
This activity based on one contributed by Dianne Meskauskas of Howard High School in Maryland.

Teaching Notes
This program is intended to be completed without the use of selection or repetition structures. Jeroo methods could be used, but they aren't necessary.

At this early stage, give the students an island file that looks something like figure A.1. Use a lot of water to reduce the overall size of the island.

Description
Angel is ready to leave school (location (0,0)) and head for home. Shortly before the last bell, the students were cautioned about the possibility that a North-to-South line of nets had been placed on the island. Write a program to get Angel from the East door of the school into his house of nets in the Northeast portion of the island. Since Angel has no flowers at school, he must pick up a flower on the way home, toss it into a net to get through the barrier, then go into the house.

Figure A.1
Suggested Island for
"Look Homeward Angel"



Solution

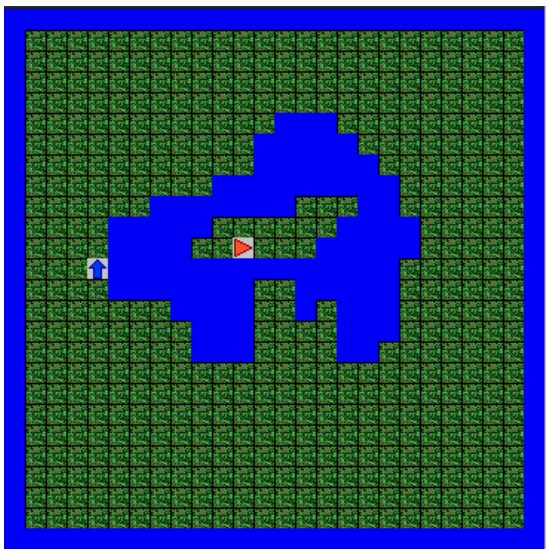
```
Sub main()
   Dim Angel as Jeroo = new Jeroo();

   '--- get the flower ---
   Angel.hop(3)
   Angel.turn(RIGHT)
   Angel.hop(3)
   Angel.turn(LEFT)
   Angel.hop()
   Angel.pick()

   '--- disable a net ---
   Angel.turn(RIGHT)
   Angel.hop(2)
   Angel.turn(LEFT)
   Angel.hop()
   Angel.toss()

   '--- go home ---
   Angel.hop(3)
   Angel.turn(LEFT)
   Angel.hop(3)

End Sub
```

WALK THE LAKE: AN ADVANCED ACTIVITY

Acknowledgement
This problem is adapted from one that was created originally by Erica Eddy of The University of Wisconsin – Parkside.

Statement of the Problem
Monsoon rains have created a lake in the middle of Santong Island. Furthermore, that lake contains an island. Two Jeroos were separated by the rains. One is on the main part of Santong Island, but the other is on the island in the middle of the lake. The goal of this program is to have each Jeroo explore the shoreline of the lake. Each Jeroo starts with one flower and the lake immediately to its right. Each Jeroo starts by planting a flower then travelling, keeping the lake on its right, until it returns to the flower. The Jeroo then picks the flower. Have the Jeroo on the main part of Santong Island walk the shore first, then have the one on the lake's island walk the shore. A representative layout is shown in figure A2.

Figure A2
A Possible Island Layout for
"Walk The Lake"



Solution

```
'===== main method =====

Sub main()

   Dim Pilgrim as Jeroo =
        new Jeroo(11,3,NORTH,1)
   Dim Mary as Jeroo =
        new Jeroo(10,10,1)

   Pilgrim.walkTheShore()

   Mary.walkTheShore()

End Sub  '===== main() =====


'=====Jeroo Methods =====

Sub walkTheShore()

   plant()
   walk()
   pick()

End Sub '===== walkTheShore =====

Sub walk()

   '--- determine next action ---
   If( NOT isWater(RIGHT) ) Then
      turn(RIGHT)
      hop()
   ElseIf( isWater(AHEAD) ) Then
      turn(LEFT)
   ElseIf ( isWater(RIGHT) ) Then
      hop()
   End If

   '--- walk recursively ---
   if( NOT isFlower(HERE) ) Then
      walk()
   End If

End Sub '===== walk() =====
```