Using Jeroo To Teach Programming Concepts

Dean Sanders
Computer Science/Information Systems
Northwest Missouri State University
sanders@mail.nwmissouri.edu

Erica Eddy
Computer Science Department
University of Wisconsin – Parkside
eddy@uwp.edu

Abstract

Jeroo [6,11] is a tool that helps novice students 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 being used successfully at three different levels: beginning programming courses, computer literacy courses, and high school courses. This paper discusses classroom experiences in beginning programming courses.

Jeroo is available at www.jeroo.org.

Introduction

Computer programming has always been difficult to teach. The students must master abstract concepts and a myriad of details. Object-oriented techniques have made the first course even more intimidating. The teaching 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. What can we do to help students master fundamental concepts of object-oriented programming, to capture their interest, and to reduce their anxiety about computer programming? Jeroo has given us a partial answer.

Jeroo is an integrated development environment and microworld that is reminiscent of Karel the Robot [9] and its descendants [2,3,5]. However, Jeroo provides a smoother transition to other object-oriented environments, and appears to appeal to a broader range of students. Jeroo was designed to help students learn to instantiate and use objects, learn to design and implement methods, and learn about control structures. 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.

Jeroo is being used successfully at three different levels: beginning programming courses, service courses for non-majors, and high school courses. This paper discusses classroom experiences in beginning programming courses.

Overview of Jeroo

Jeroo is both an engaging virtual animal and an integrated development environment. Throughout this paper, the word Jeroo will refer to either the animal or the development environment, depending on the context.

Jeroo evolved over several years from pencil-and-paper activities, to a Pascal-like tool named Jessica, to the current object-oriented tool with its graphical user interface. Comments and suggestions from countless students helped mold Jeroo into its current form. Jeroo was designed for novice programmers, but students who have completed a prior programming course find Jeroo interesting and worthwhile.

Jeroo is suitable for several kinds of courses. Its apparent simplicity makes it suitable for computer literacy courses, but its intellectual content makes it a useful supplement to a course in computer programming. Jeroo's programming language styles make it suitable for use in a course where the primary programming language is Java, C++, C#, or VB.NET. Written in Java and tested under Windows, Mac OS-X, Linux, and Solaris, Jeroo it suitable for a wide range of laboratory environments.

There are four significant aspects of the Jeroo environment, the metaphor, the user interface, the programming languages, and the runtime behavior. These are discussed in the following subsections.

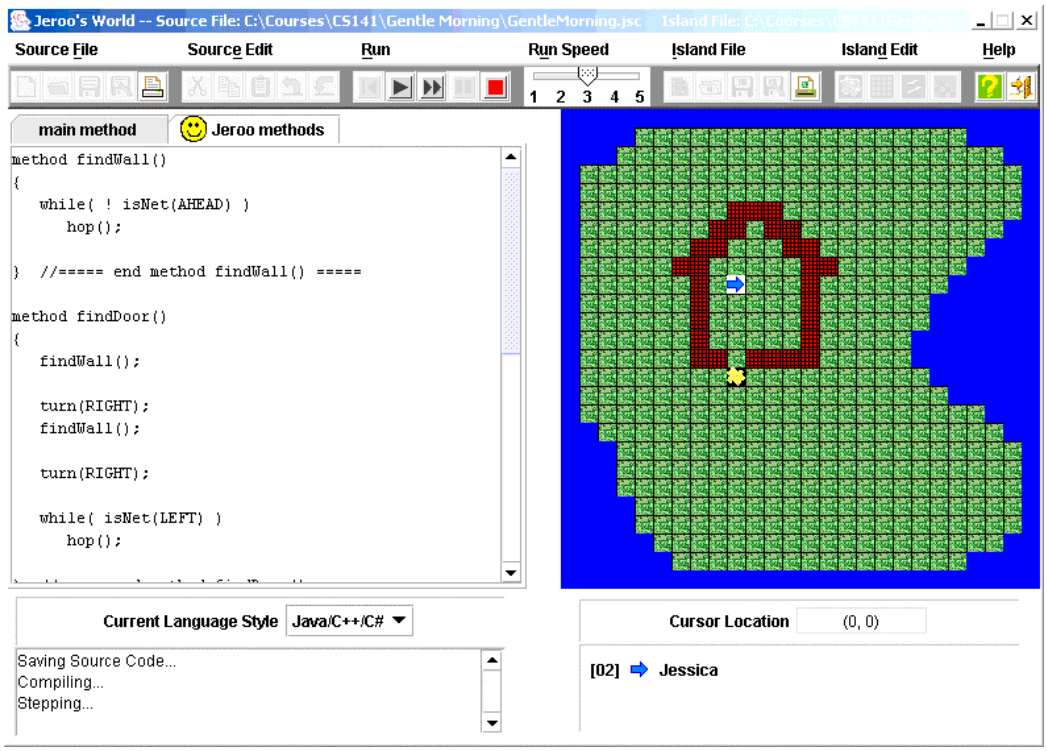
The Metaphor

A Jeroo is a 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 collector who is seeking new specimens for a zoo. Despite living on the island for untold millennia, the Jeroos are poor swimmers and must avoid the ocean and inland bodies of water.

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. Figure 1 shows the appearance of the user interface for a running program.

Figure 1: The User Interface for Jeroo



The left-hand portion of the screen contains tabbed panes for editing source code. One pane is used for a main method in which Jeroos are instantiated and used to solve a specific problem. The other pane is used for programmer-defined methods that extend the behavior of all Jeroos. The source editor supports common editing features together with unlimited undo/redo, block indenting, and block commenting.

The right-hand portion of the screen contains a representation of Santong Island. Students use a point-and-click process to alter the shape of the island and to place flowers, nets, and water on the island. A cursor location panel helps the students locate specific cells on the island. The island area also displays a simple animation of a running program.

The lower portion of the screen contains two status panels. One displays status messages, syntax error messages, and runtime error messages. The other is continually updated to display the state of every Jeroo in a running program.

Jeroo's Programming Languages

Students write programs to control the actions of up to four Jeroos and their interactions with their immediate surroundings. Jeroo supports two different language styles, a Java/C++/C# style and a VB.NET style. The syntax of the Java/C++/C# style language mirrors the common syntax of those languages. The only difference is that a generic header is used for methods in the Jeroo language. Jeroo's VB.NET style language is a faithful subset of VB.NET. This syntax allows a teacher to say "The Jeroo language is really a subset of Java (or C++ or C# or VB.NET)", 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 twelve predefined methods: six sensor (or Boolean) methods and six 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 conditions for these structures are constructed from one or more sensor (Boolean) methods and the operators &&, ||, and ! (AND, OR, and NOT in VB.NET).

The Runtime Behavior

A Jeroo program can be executed stepwise or continuously at one of five different speeds. Students can change speeds or switch back and forth between stepwise and continuous execution while a program is running. Each execution mode features animated execution and simultaneous code highlighting. A status panel under the island is continuously updated to show the current state of every Jeroo. In the event of a runtime error, this panel shows the state of every Jeroo just prior to the error.

Error messages and status messages are displayed in a status panel under the editor. Care has been taken to make the error messages as informative as possible. A typical message for a syntax error has the form "XXX was expected; YYY was found." When combined with highlighting of the offending source code, these messages make it easy for students to identify their errors.

Using Jeroo in a Pre-CS1 Course

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 Java-based CS1 course. The faculty had observed that students taking CS1 seemed to struggle with the object-oriented concepts much more than with the other portions of the course. In the Fall 2003 semester, the programming language for CSCI-130 was switched from QBASIC to Jeroo. The motivation for this change was to gently introduce the ideas of object-orientation in the hope that those who continued on to take CS1 would be more successful.

Both a lecture room and a computer lab are reserved for one 2-hour block per week to allow the instructor to vary the class between lecture and hands-on activities. CSCI-130 follows the content outline given in [11], but extends the time on Jeroo from 3-4 weeks to a full 14 weeks. The only text for the course was "Introduction to Jeroo" [10], which was duplicated and made available in the campus bookstore with the author's permission. The course outline appears in appendix A.

Using Jeroo in a CS1 Course

Jeroo is used in a beginning programming course at Northwest Missouri State University (NWMSU). This Java-based course is equivalent to CS101o in the objects first model of Computing Curriculum 2001 [7]. There are two ways to use Jeroo in such a course. Jeroo can be used exclusively for the first three or four weeks, or can be interleaved with instruction on related topics in Java (or other programming language). Based on experiences with both methods, the latter seems to be superior.

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 Java. Unfortunately, the transition to Java 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 Jeroos. Now, in Java, they are working with a single method, *main*, and writing simple programs such as "convert Fahrenheit to Celsius". 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 Java. Even when the ties to Jeroo are exposed, there is a disconnection between Jeroo topics and Java topics. Those who have used Karel-like tools have reported similar problems, informally.

Interleaving Jeroo topics with related Java topics alleviates the difficulties associated with covering all of Jeroo as a single unit. Jeroo is used in four units at NWMSU. In each segment, Jeroo is used to introduce concepts then those concepts are revisited and expanded in Java. The first unit introduces the concepts of algorithms and three programming topics: the role of the main method, object instantiation, and method invocation. At NWMSU, Java's main method is not introduced at this point; instead, the students use BlueJ [8] (DrJava [1] would work as well) to instantiate and send messages to library objects. The second unit uses Jeroo to introduce programmer-defined methods. This is followed by learning how to write complete classes in Java. The students design classes, instantiate objects, and send messages to those objects, but they don't write complete programs yet. 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 Java.

By interleaving the use of Jeroo with coverage of the corresponding topics in Java, 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 BlueJ were unfounded. The students had no trouble moving between these environments.

Perceived Benefits

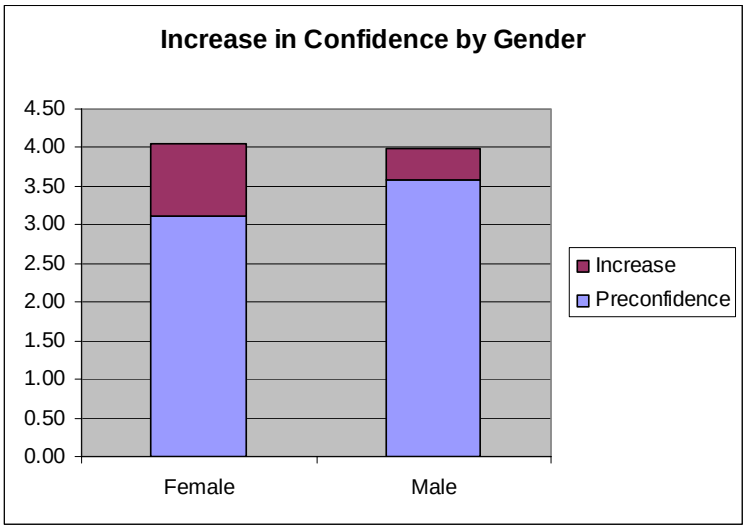
The benefits of working with Jeroo can be organized into three broad categories: programming concepts, programming practices, and student satisfaction. The benefits pertaining to programming concepts and practices are derived from faculty observations, but student satisfaction is based on objective data.

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 CS1 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 2. 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 2: Increase in Confidence Level

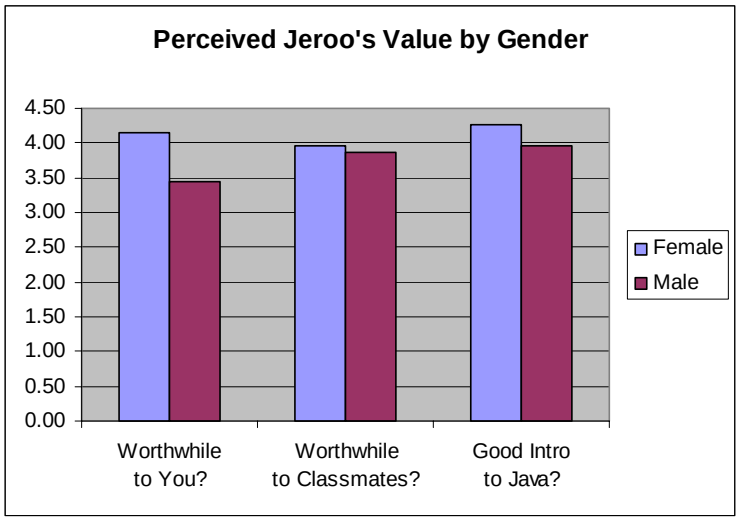


A recent study [4] indicates that females have lower confidence than males in their computing abilities, even when controlling for quantitative ability. If Jeroo can level the playing field with respect to confidence and comfort in the course, it may lead to an increase in the number of women who choose computer science 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 2.

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 3. As is evident from the graph, the female students rated Jeroo slightly higher than did the males.

Figure 3: Jeroo's Perceived Value



Lower withdrawal rates and higher continuation rates are a second indication of student satisfaction. Data for CSCI-130 at UW-Parkside show a withdrawal rate of 19% in the four terms preceding the introduction of Jeroo, and a withdrawal rate of 5% in the two terms since the course switched to Jeroo. Data from the same terms show that the percentage of those who continued on to CS1 increased from 28% to 32%.

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 comment from a student at UW-Parkside is typical.

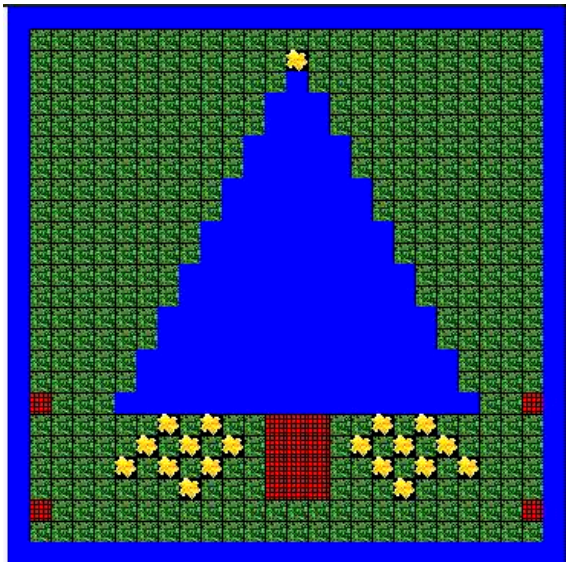
> "Jeroo helped me tremendously. I took CSCI-241 [the CS1 course] first without any prior programming experience and was lost from the first day. Taking the same class after experience with Jeroo is much easier to understand. I think Jeroo is a good way to ease into programming with Java. For me, it made the whole idea of instantiable 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."

Summary

Jeroo has proven to be quite popular at UW-Parkside and Northwest Missouri State University. It is an effective tool for teaching the concepts of objects, methods, and control structures to novice programmers. The animated execution and simultaneous code highlighting aid understanding. 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 Java, C++, C#, or VB.NET.

The uses of Jeroo are limited only by the instructor's imagination. Nets, water and flowers can be used to represent almost anything else. For example, the island layout shown in figure 4 was used on a final exam for CSCI-130 at UW-Parkside.

Figure 4: Christmas Island



References

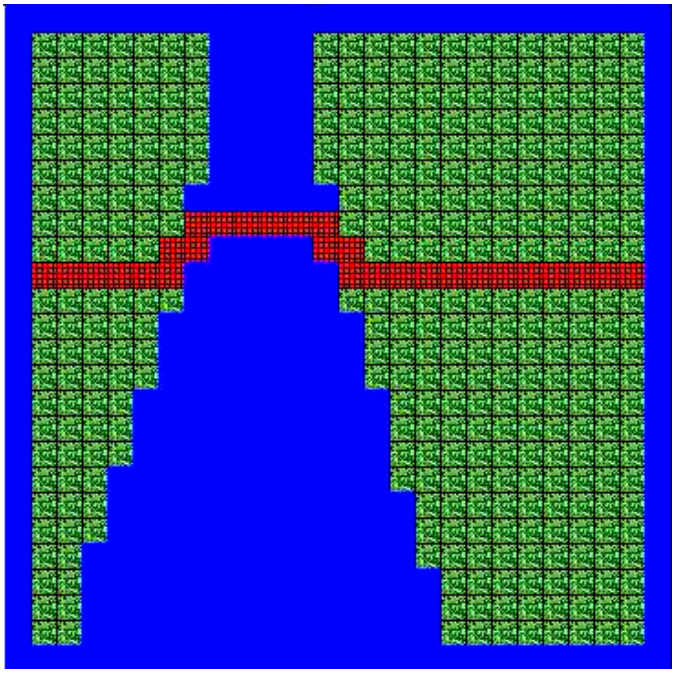
1. Allen, E. Cartwright, R., & Stoler, B. (2002). Dr.Java: A Lightweight Pedagogic Environment for Java. SIGCSE Bulletin, 35(1), 156-160.
2. Bergin, J., Stehlik, M., Roberts, J., & Pattis, R. (1997). Karel++: A Gentle Introduction to the Art of Object-Oriented Programming. New York: John Wiley & Sons.
3. Bergin, J., Stehlik, M., Roberts, J., & Pattis, R. (2002). Karel J. Robot: A Gentle Introduction to the Art of Object-Oriented Programming in Java. Retrieved March 11, 2004, from: http://csis.pace.edu/%7Ebergin/KarelJava2ed/Karel++JavaEdition.html.
4. Beyer, B., Rynes, K., Perrault, J., Hay, K., & Haller, S. (2003). Gender Differences in Computer Science Students. SIGCSE Bulletin, 35(1), 49-53.
5. Buck, D. & Stucki, D. (2001). JKarelRobot: A Case Study in Supporting Levels of Cognitive Development in the Computer Science Curriculum. SIGCSE Bulletin, 33(1), 16-20.
6. Dorn, B., & Sanders, D. (2003). Using Jeroo to Introduce Object-Oriented Programmming, FIE 2003 Conference Proceedings, (CD Version) IEEE Catalog Number 03CH37487C. ISBN 0-7803-7962-4.
7. Engel, G., & Roberts, E. (Eds.). (2002). Computing Curricula 2001 Computer Science: Final Report December 15, 2001. Los Alamitos, CA : IEEE Press.
8. Kolling, M. & Rosenberg, J. (1996). An Object-Oriented Program Development Environment for the First Programming Course. SIGCSE Bulletin, 28(1), 83-87.
9. Pattis, R. (1995). Karel the Robot: A Gentle Introduction to the Art of Programming (2nd ed.). New York: John Wiley & Sons.
10. Sanders, D., (2003). Introduction to Jeroo. Retrieved March 11, 2004, from http://www.jeroo.org.
11. Sanders, D., & Dorn. B. (2003). Jeroo: A Tool for Teaching Object-Oriented Programming. SIGCSE Bulletin, 35(1), 201-204.

Appendix A: CSCI-130 Schedule

Week	Description
1	• Basic first-day-of-class information • Demonstration of how to install Jeroo on a home computer • Demonstration of the Jeroo programming interface • Students type a simple program in the lab to compile and run
2-3	• Algorithms and vocabulary, including object-oriented terms • Students write algorithms to solve Jeroo problems and perform common human activities such as purchasing pretzels from a vending machine
4-5	• Constructors and their purpose; discussing state and arguments • Action methods for introducing the process of sending messages to objects • Introduction to a variety of errors and error messages Specific activities allow students to learn how to interpret error messages and helps reduce their stress level when errors occur in their own programs. Students are now ready to start with simple programs and progress to more complicated ones. From the start, they are told to use good programming style, including proper comments, indentation and naming conventions for objects.
6	• User-defined methods • Students learn to create their own methods, including proper naming conventions. • Students develop programs with multiple Jeroos that use the same user-written methods. Doing so helps the students learn that methods are not object-specific. Appendix A contains an assignment that is suitable for this unit of the course.
7	Two-part (Midterm) exam • Short answer portion • Programming portion in which the students correct, then extend, a Jeroo program that has been provided by the instructor
8-9	• Sensor methods (Boolean methods) and their parameters. • Nested if-statements • Logical operators (not, and, or) • One assignment asks the students to write different scenarios for the Jeroos where an if-statement would be useful.
10-11	• While loops, including infinite loops • Rewriting of earlier programs (refactoring).
12-14	• Combining if statements with while loops • Students develop more flexible programs that will work with several different island layouts • Algorithms are designed and discussed before the programs are implemented
15	• Final exam structured like the midterm

Appendix B: The Bridge Assignment

Begin with the island file named Bridge.jev as shown in the picture. The island contains a river that runs north/south, and is wider at the bottom of the island. A path beginning at location (9,0) leads to a bridge across the river, and on to the Eastern Ocean. The path and bridge are covered with nets.



Your program will need 2 Jeroos. The first Jeroo should begin at the left side of the path. It will cross the path and bridge, throwing flowers at each plank (net) and then will plant a flower in place of that plank. It will cross the entire bridge without going into the water, or making any unnecessary turns or flower tosses.

The second Jeroo will sit at the right side of the path waiting until the first Jeroo makes its way there. When the first Jeroo finishes replacing the last net with a planted flower, it should have one flower left. It should give that flower to the waiting Jeroo, then hop out of the way so that the second Jeroo can cross the flower bridge in the opposite direction picking each flower it comes across.

You need to make a number of decisions in this project:

- Decide on the first Jeroo's location, and how many flowers it will need to make its way across the bridge. Remember, it must replace every net with a flower and have exactly one flower left to give to the second Jeroo.
- Decide on the second Jeroo's location so that it will not interfere with the first one.
- Create methods that seem appropriate for the task. When writing the methods, be sure to comment them with their purpose at the top of the code.

Your work will be graded on:

- Use and quality of comments
- Decisions made regarding Jeroo placement
- Decisions made regarding new methods that you create