Frog and Bug: Building Two-Agent Models from the Ground Up

Colin Fitzpatrick and Ellen Gaffney

Purpose

This lesson represents a crucial point in learning programming, switching from exercises to creating programs from scratch. The two activities involved in this lesson help scaffold the critical thinking required for being successful in this transition point, emphasizing starting with the simplest thing that could possibly work and the role of pseudocode in developing programs. Furthermore, these activities simulate practice of programmers in the real world, as they regularly model real world phenomenon, discuss how to capture those phenomenon through talking in pseudocode, and implement their models starting small and building up.

Overview

This lesson has two main parts: the group exercise and the pair activity. In the group exercise, the class will practice together defining a simple system and translating the rules of that system from pseudocode and code (Frog and Bug exercise). In the pair activity, students will receive a prompt, build pseudocode together, and code a new program (Rock, Paper, Scissors). For both, emphasis will be placed on "Do the Simplest Thing That Could Possibly Work" (DTSTTCPW), to start small and simple, get things working, and build out from there.

Student Outcomes

Students will be able to model (simple) real world situations through programming using class objects, discuss the decision-making process and limitations of simulations, and build a program from pseudocode.

Standards Addressed

MS-ETS1-1 - Define the criteria and constraints of a design problem with sufficient precision to ensure a successful solution, taking into account relevant scientific principles and potential impacts on people and the natural environment that may limit possible solutions.

MS-ETS1-2 - Evaluate competing design solutions using a systematic process to determine how well they meet the criteria and constraints of the problem.

MS-ETS1-4 - Develop a model to generate data for iterative testing and modification of a proposed object, tool, or process such that an optimal design can be achieved.

Time

1 class period (40 mins); alternative 2 class periods (see Teaching Notes below)

Level

8th Grade; Advanced Learning: Intro to Programming

Reach for the Stars is a GK-12 program supported by the National Science Foundation under grant DGE-0948017. However, any opinions, findings, conclusions, and/or recommendations are those of the investigators and do not necessarily reflect the views of the Foundation.

Materials and Tools

Measuring tape or rope
Laptops, with Python IDLE installed
Projector
Rock, Paper, Scissors Prompt

Preparation

Arrange desks into groups of even numbers (so that students may work in pairs). Leave space in the center of the room for the Frog and Bug activity.

Prerequisites

Students should be familiar with the basics of programming, including variable declaration, classes, functions, conditionals, and loops.

Background

This activity has you apply your developing skills in various areas of programming, including variable declaration, conditionals, functions, and classes. In order to succeed, you must carefully think through real world situations, describe them in writing (pseudocode), and then transfer that writing to programming. Today the emphasis is on capturing systems (writing pseudocode) and recreating them in code (transfer).

Teaching Notes

Introduce the background and format of the lesson at the beginning. Emphasis that the starting point today is not in programming, but in the real world. To illustrate this, begin with the Frog and Bug activity:

Frog and Bug

1. Solicit two volunteer students. Designate one as Frog (give her/him the measuring tape or rope), designate the other, Bug.
2. Ask the class how a Frog and Bug might interact in the real world. (Should hear responses like "Frog will try to eat the bug")
3. Move Frog to the center of the room, explain that this Frog is lazy and will not move from this spot.
4. Ask Bug to move anywhere in the room randomly (thinking ahead: using random in code) and then stop.
5. Ask class if the Frog can eat the Bug from this position. Ask how they know (goal is to connect the length of the tongue and the position of the bug to a conditional to be used later in code).
6. Repeat steps 4-5 until the Bug can be reached and is eaten. At that point, students may return to desks.
7. Using a whiteboard/chalkboard/projector, discuss and record the "rules" of the environment. Ask questions such as "How many creatures were there? Are they the same type? What properties must we know about them? How do they interact with the space? What constitutes a turn or round? How do we know when it is over?"
8. Using a projector, translate the pseudocode generated in 7 into code. For each piece of information, call on a new student to transfer it to code (or have the student update/edit a previously written piece of code). An example solution is provided.
9. After running the program, start a discussion around what could be done to make the Frog and Bug program more faithful to real life and what could be done to improve the current syntax, structure, and format of the program.

For the second half of the class, the students will work in pairs and go through the above process of moving from real world to pseudocode to code again:

Rock, Paper, Scissors

1. Have students break into pairs and hand out a RPS prompt sheet one per pair. Instruct the students to read through the prompt, discuss the criteria and constraints of the environment, and then record pseudocode for it. (Prompt: Rock, Paper, Scissors is a popular informal game that can be played between two people. The players face each other, count to three together, and then "shoot" by playing either a "rock," "paper," or "scissors" with his or her hand. Rock beats scissors, scissors beats paper, and paper beats rock; if the players shoot the same hand, they go again. Write pseudo code to represent one round of play between two players.) An RPS example solution is provided.
2. As pairs finish their pseudocode, hand out a laptop to each pair.
3. Have pairs transfer their pseudocode to code.
4. Have students turn in both their prompt handout with the written pseudocode, as well as submit their program electronically.

Alternatives – Depending on class pacing and students' command of basic skills, each activity could be "blown out" to take up an entire period itself:

To expand Frog and Bug, have the students work in groups to discuss the model and generate pseudocode. Then have each group share their pseudocode and enter a class discussion about differences/similarities across the pseudocodes. Construct a "class approved" version, combining from those provided, and then implement the program together. Put extra emphasis on program syntax and naming conventions for variables and classes.

To expand RPS, have the pairs write their pseudocode and swap with another pair, discussing the differences and similarities between them. Have students code their programs and then demo them in front of the class on the projector, with students talking through decisions they made. For early finishers, have them expand their program so it plays a "best-of-3" version of RPS.

Samples of complete pseudocode and working programs are included in the files for this lesson plan.

Assessment

Frog and Bug serves as a class introduction to the two main ideas in this lesson, DTSTTCPW and pseudocode. While not necessarily designed for assessment, questions raised and answers given during the activity can help gauge the room on how well the students are capturing transfer among the "real world", pseudocode, and code. Finishing with questions around articulating tradeoffs during decisions in generating the code and suggestions for how to improve the model also point towards beginning understanding of higher level concepts.

The RPS activity provides two artifacts for evaluation, the handout's pseudocode and the program itself. Each can be examined independently for key ideas/features, or looked at together to see how the model generated in pseudocode was transferred to code. In the latter case, it is worth noting any changes between the two, which marks evidence of using an iterative design process.

Additional Information

Example of Frog and Bug pseudocode:

Knowns: The frog has tongue length; the bug has distance from frog and whether or not it has been eaten.
Action: Initialize a bug and frog. While the bug is still uneaten, update the distance of the bug (randomly); if the bug position is equal or less than then length of the frog's tongue, the bug is eaten and the program ends.

Example of RPS pseudocode:

Initialize two players, each with a hand attribute and a play function that randomly selects and sets a players' hand attribute value from rock, paper, or scissors. Compare the hand values of the two players; if the values are the same, play again, if not, declare a winner based on the rules of the game (rock beats scissors, scissors beats paper, paper beats rock).