In, AAAI Spring Symposium on Using AI to Motivate Greater Participation in CS, March 20008, AAAI Press

Engaging Computing Students with AI and Robotics

Deepak Kumar, Doug Blank Computer Science Bryn Mawr College {dkumar, dblank}@cs.brynmawr.edu

Tucker Balch, Keith O'Hara, Mark Guzdial College of Computing Georgia Institute of Technology {tucker, kjohara, guzdial}@cc.gatech.edu

Stewart Tansley Microsoft Research Microsoft Corporation stansley@microsoft.com

Abstract

In this paper we describe a new curriculum for a CS1 course that uses personal robots as a context for learning introductory computer science. Students learn several computing and AI-related concepts in the process of exploring and designing robot behaviors. We believe that the use of personal robots and engaging examples can provide a good foundation for learning computing and hence serve to attract a more diverse body of students into the computing disciplines. In this paper, we describe how we have embedded numerous AI concepts in the design of our curriculum.

Introduction

Computer science (CS) has seemingly lost its appeal to many of today's students, and personal robots perhaps can help find it again. Paradoxically, even as computing has permeated every aspect of our lives, computer science as a field of study is often seen as disconnected from these same lives. To reestablish the connection between student interests and the myriad career and intellectual possibilities provided by contemporary studies in CS, the Institute for Personal Robots in Education (IPRE, roboteducation.org) was created in July 2006.

IPRE is developing a personal robot, software, and modern curricula to help teach introductory computing courses (Blank 2006). Our vision is that the text for such an introductory course would come bundled and shrink-wrapped with a ready-to-run personal robot in the same price range as current CS1 texts. Having an artifact—in this case a ready-to-run robot—provides intrinsic motivation to both the instructor and the student to explore the science and engineering behind it. Students engage in learning computing for reasons today that are very different from those traditionally identified: such as fun, curiosity, diverse range of applications that portray computing as a helpful discipline, and to show off to family and friends. A project of such undertaking requires careful attention to all aspects of the course design: the robot, software, course materials, and the lab environment.

In this paper we will focus on the design of the curriculum for a CS1 course based on our concept of personal robots. We will outline a number of AI concepts that have been key in making the course interesting and engaging. Besides the use of a robot itself, we have incorporated a number of AI concepts embedded in the course materials, e.g. robot control paradigms, vision, game playing, learning, etc. These concepts are permeated throughout the course primarily to provide a context for engaging and interesting robot behaviors and applications. While many of these concepts traditionally come from the domains of AI, we do not explicitly identify them as such as AI ideas- we introduce these concepts stealthily and naturally. There is, additionally, a full section devoted to AI itself in the latter part of the course. In the next section, we summarize our overall approach to the CS1 curriculum design process.

Approach to the curriculum

The key underlying motivations that are driving the development of the curriculum are presented first.

A personal robot Every student gets her/his own personal robot. Our vision is that students registered in a CS1 course taking this approach will go to the bookstore and purchase his/her own personal robot, just like they would purchase a text for the course (See Figure 1). Additionally, since the student owns the robot it can be personalized and used in future classes.

Figure 1. Student Purchasing a Personal Robot

Let the needs of the curriculum drive the design of the robot The design of the personal robot is motivated by the requirements of our new curriculum and is an outcome of feedback obtained from students in our pilot course offerings. It is important to understand that we place the robot firmly in the role of a motivational context – teaching robotics is not our intent; teaching computing concepts by illustrating them in a robotic context, is the approach.

Use tools that are easy to use, scale with experience We want the students to use the tools (computer, programming language, IDE, etc.) to be such that they are not designed specifically (and only) for use in CS1. We want the entire programming environment to be pedagogically scalable to broader contexts. This way, concepts acquired in the new CS1 easily carry over into more advanced computing situations without the need to change the programming environment.

Robot as a peripheral Instead of directly programming the robot at the robot's level (often a microcontroller), the student uses the full power of a personal computer for development and debugging. In this mode, the desktop or notebook computer commands and queries the robot over a wireless tether.

Create an accessible, engaging environment for new, diverse population of students It is well acknowledged today that the introductory computer science curriculum is broken and is in need of a major overhaul. With that wide acknowledgement also comes a wide range of proposed 'solutions'. Ours is one of them. We are taking the issue of accessibility to a wider population as our primary goal. In addition, we have been able to introduce many sophisticated topics using a simple environment. Often, the really interesting topics (such as AI and robotics) are reserved for only advanced students. We hope to allow introductory students to experience some of the excitement such topics bring to computing. On the other hand, we are also mindful of the adverse affects the use of certain technology, and pedagogical examples can have on people from different gender and backgrounds. Our curricular materials attempt to address these issues as well.

Computer Science ≠ programming While programming is central to our approach to CS1 we are also conscious of avoiding the misperception that programming is all there is to computer science. Students from the new CS1 should come away with a solid understanding of the scope of computing, the role of programming in it.

Make computing a *social activity* There is an explicit attempt in our approach to make computing a social activity. By this we mean that we will strive, in our curriculum, to make every aspect of the learning process a collaborative and social activity. Students learn from each other and by working on their robots in their own environments (dorm hallways, dining halls, study spaces, labs) and interacting with others in meaningful ways.

Make computing a medium for creativity Creativity is central to robot design and we intend to include several creative aspects into the curriculum. Examples include exercises that demonstrate robot behaviors like dances, choreographed movements, music and song generation, movie making, game playing, robot application design, etc. Most exercises will be open ended (i.e. correctness of the output of a program is not determined by a limited set of output) and encourage students to experiment, play, and be creative.

Performances vs. competitions All robot exercises in the course include demonstrations. However, the demonstrations are going to be depicted and evaluated as performances and not as competitions among peers. We have found that competitions tend to attract only a minority of the student body and serve to deter many students. However, a non-competitive, collaborative, and social environment encourages learning and motivates students to strive for higher goals. If an individual professor chooses to take a competitive approach, many of our materials and technologies are still nonetheless appropriate and usable, it is just not our recommended approach.

The Robot & Software

Figure 2. The Scribbler Robot with IPRE Fluke Board

For the pilot offerings of our courses, we have been using the Scribbler robot by Parallax Corp. (www.parallax.com) combined with an add-on board called Fluke, designed by IPRE (See Figure 2). This package costs approximately $110 and has the following features: IR obstacle sensors, three light sensors, IR line sensors, a stall sensor, a color camera, two programmable LEDs, a two-frequency tone

generator, Bluetooth wireless communication, and a pen port that can be used for scribbling on the floor. The add-on board transmits color images, and at faster rates: grayscale, windowed, and color segmented images. Students write programs on a host computer that communicates with the robot over a Bluetooth connection (with a range of 100 meters).

The programming language used is *Python*, along with our API called *Myro* (for *My Ro*bot) which provides easy to use and yet powerful abstractions for robot programming and control. Additionally, it incorporates easy to use features for text-to-speech, creating dialog boxes, image and multimedia processing, IM-style communication, and other advanced features.

Myro is inspired by many of the ideas in Pyro (Blank *et* al, 2006). Myro extends Pyro as an environment more specifically focused on teaching programming, where as Pyro has wider goals. Further, Myro expects to leverage new technologies such as Microsoft Robotics Studio (micorsoft.com/robotics), bringing extended flexibility to work with different hardware beyond Scribbler and other programming languages beyond Python. Further discussion on these aspects is outside the scope of this paper.

Where is the AI?

In the design of the text for our introductory course, we have embedded several AI concepts and techniques as well as several non-AI topics (IPRE 2007). The outline of chapters of the text is shown below:

1. The World of Robots
2. Personal Robots
3. Building Brains
4. Sensing From Within
5. Sensing The World
6. Insect-like Behaviors
7. Control Paradigms
8. Sights & Sounds
9. Robot Vision
10. Artificial Intelligence
11. Computers & Computation
12. The World of Computing

In the first chapter, students explore the world of robots using the Mars rovers, *Spirit* and *Opportunity*, as motivating examples. Students are introduced to their own personal robots: they give their robot a name, personalize it, and learn how to control them manually through a game pad controller. Students study the history of robots, and are introduced to some of the state-of-the-art robot applications (lawn mowing, vacuuming, surgery, etc.). As an exercise, students use a game pad controller to explore a pyramid structure (see Figure 3).

In the next two chapters students learn some basic programming concepts: *names/variables*, *values*, *functions* and basic *program* structure. Students use Myro abstractions for robot movements (forward, backward, stop, etc.). A program is introduced as the *brain* of the robot. In exercises, students create more sophisticated robot movements by incorporating them into robot dances.

The chapters on sensing focus on *reactive robot control*. Students learn the functionalities of various sensors, how to obtain their values, and combined with the *if*-statements, learn to write several *smart* robot behaviors. Beginning with a reactive approach is also advocated by Martin (Martin, 2007).

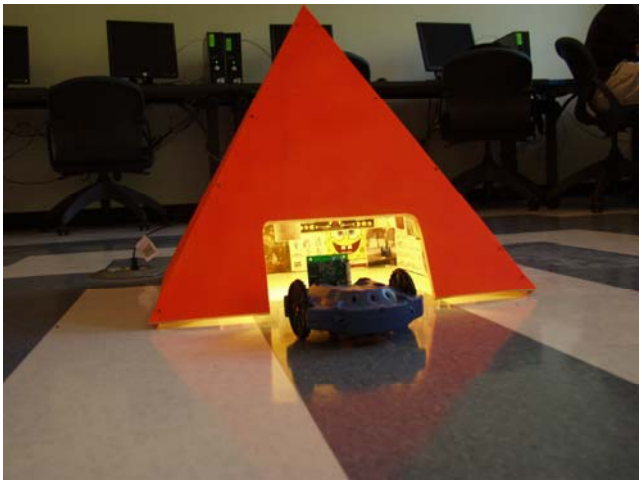

Figure 3. Exploring a pyramid exercise.

The *Insect-like Behaviors* chapter introduces the idea of creating/programming robot behaviors using the *Braitenberg paradigm* (Braitenberg 1984). That is, the same behaviors that were programmed using reactive control can also be accomplished by means of simple mathematical transformations. Students learn to design several Braitenberg-style robot behaviors and indulge into a little bit of synthetic psychology. An example *brain* for the simplest vehicle is presented below:

```
def main():
   # Braitenberg vehicle#1: Alive

   # For 60 seconds do the following…
   while timeRemaining(60):
     l = getLight("left")
     r = getLight("right")
     motors(normalize(l), normalize(r))
```

Students can then explore writing different normalization functions to observe changes in behaviors.

The chapter on *Control Paradigms* introduces the concepts of *behavior-based control* (Arkin 1998). Students learn how to program robot behaviors using the subsumption

paradigm. Below, we show the skeletal control structure of a single-threaded subsumption-based program:

```
# A simple subsumption-style brain
def arbitrate():
   for behavior in behaviors:
      output, T, R = behavior()
      if output:
         return T, R

# behaviors, ordered by priority
behaviors = [seekLight, avoid, cruise]

def main():
   while True:
      T, R = arbitrate()
      move(T, R)

main()
```

In the above program, we take advantage of Python's notion of *names* that essentially render functions as first-class objects. The complete program above would require students to write the functions `seekLight`, `avoid`, and `cruise`. In the design of our programming examples, we also try to illustrate the view of creating well-structured programs. We take full advantage of the advanced, yet easy to use, features of the Python language.

The chapter on *Artificial Intelligence* introduces the field of AI and provides a context for learning AI concepts using robots. We present examples from *natural language processing*, *case-based reasoning*, *game playing*, and also *machine learning*. Some examples are presented as demos showing what is feasible while others are directly implementable by students. For example, in discussing natural language processing, we illustrate the following interaction with the robot:

```
User: do you see a wall?
Scribbler: No

User: Beep whenever you see a wall.
User: Turn right whenever you see a
      wall to your left.
User: Turn left whenever you see a wall
      to your right.
User: Move for 60 seconds.
```

[The Scribbler robot moves around for 60 seconds turning whenever it sees a wall. It also beeps if it sees a wall.]

We provide an implemented system with which students are able to enter natural language commands and watch their robots respond to them. While we discuss the architecture of this implementation at an abstract level, the details are beyond the level of this course. However, it provides the students first-hand experience with AI programs that are essentially built using the techniques they have learned earlier in the course. Sentences entered are parsed using facilities provided in NLTK (Bird and Loper 2004) and analyzed into a dynamic subsumption control architecture (Blank *et al* 2007, Walker 2007).

Examples of case-based reasoning are illustrated in the domain of the Rock-Paper-Scissors game (using case histories of past games to beat the human player). Machine learning is illustrated using backprop networks to learn the kinds of behaviors that students programmed in earlier chapters. Myro includes an abstract neural network modeling module, *conx* that enables students to explore these ideas without getting bogged down by the details of neural network implementations (Blank *et al* 2006). Using abstract functions, students can design and specify a network architecture, provide learning parameters and then watch their robots/networks learn the behaviors.

Additional topics which are often reserved for only advanced computing students can also be explored. For example, using easy-to-use functions for image processing allows students to explore topics in real-time computer vision (Guzdial, 2005). In the chapter on *Robot Vision* students are challenged to have their robot find the pyramid and approach it. One solution to this problem is to identify the pyramid in an image. Luckily, the pyramid happens to be fluorescent orange. Using a feedback algorithm, the students can write a short program to move towards the pyramid:

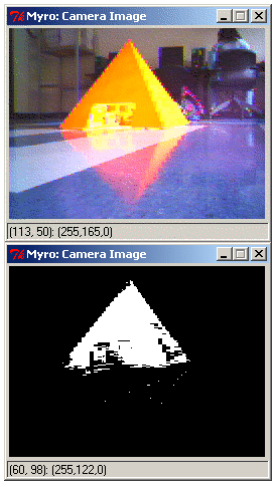

Figure 4: Image Processing

```
pic = takePicture()
show(pic) # Figure 4, top picture
xs, ys, count = 0, 0, 0
for pixel in getPixels(pic):
    r, g, b = getRGB(pixel)

    if r > 250 and b < 100 and g > 130:
```

```
        setColor(pixel, white)
        xs += getX(pixel)
        ys += getY(pixel)
        count += 1
    else:
        setColor(pixel, black)

show(pic) # Figure 4, bottom picture
```

Thus, numerous AI concepts are implicitly or explicitly embedded into the design of the curriculum: reactive control, Braitenberg-style control, behavior-based control, natural language processing, case-based reasoning, game playing, machine learning, and image processing. There are additional chapters on other non-AI topics: sound, music, communications, computation, etc.

In taking a fresh view of introductory computing courses our goal has been to design a coherent approach to the use of personal robots as a context for learning computing. In that way, we have been open to any domains and examples that lend themselves naturally to the context. As can be seen, AI (and other domains) lends themselves naturally to this design framework. What we are able to incorporate has been a result of conscious deliberations on the introductory nature of the course and serving the underlying philosophy of our approach as outlined earlier. We have discovered that the resulting course takes full advantage of several advances: in hardware design, software design, robotics, and AI. The course syllabus goes well beyond a traditional CS course as recommended by ACM Curriculum 2001 (Computing Curricula 2001) and introduces a healthy dose of interesting and engaging ideas for motivating students to learn computing.

Pilot Courses and Assessment

The materials have been used in six pilot offerings of CS1 courses: at Bryn Mawr College in spring and fall 2007 with 60 students (nearly all women), at Georgia Tech in spring, summer, and fall 2007 with over 200 students. Spring 2007 was a particularly opportune time to study the Georgia Tech class because the same lecturer was in both our robotics and non-robotics introductory course sections. We studied both sections using the same survey instruments. Our evaluation had three stages:

- We conducted a midterm survey to gather open-ended comments on what students thought about the classes.
- We used the survey comments to develop an interview script that we used with three students in the robotics section of the course at Georgia Tech.
- We analyzed the interview scripts to identify themes—opinions or attitudes that we wanted to explore in the course. We constructed a final survey, which was completed by participants on both campuses and in both versions of the course.

Extensive feedback (both qualitative and qualitative) was collected from all completed courses. The results obtained from students can be summarized in these main points:

- Students learned CS concepts through robots
- Robots made learning experience more hands-on, tangible, and exciting
- Most frustrating parts were dealing with robot hardware inconsistencies
- Viewed CS as a type of logic and problem solving; requiring patience & thought
- Discovered that CS and robots are applicable to the real world

The goal of our pilot offerings is to refine our curriculum, help evolve the software, and to use the feedback obtained from students to design the specifications of the eventual personal robot that will be used in subsequent offerings. The initial results are encouraging, and informative. Our plans in the near future include expanding the pilot offerings to other institutions and to hold workshops to train faculty in the use of this approach.

Beyond CS1

As we proceed with our project, we are also aware of the attractiveness of our software and robot platform for other upper-level courses. We have already seen some adoption in computer science and other engineering departments. We ourselves have found places where advanced students can use the robots and software for their research projects (Blank *et al* 2007, Walker 2007). Our goal of making the entire design pedagogically scalable contributes directly towards these applications that go beyond the introductory curriculum.

Another important goal of our project is to make the Myro API available for several other robot platforms, much like we did for our earlier work on Pyro (Blank *et al* 2006). This will no doubt increase the viability of our work beyond CS1 courses.

On Engaging Students

The issue of attracting, and retaining students into the computing disciplines lies at the heart of the current enrollment crisis in computing (Vesgo 2005). The issue is multi-faceted and therefore will require multi-faceted approaches and solutions. There have been several studies devoted to examining these issues. Several key factors have been identified as deterrents for students to enter the field of computing (Margolis and Fisher 2003, Burger *et al* 2007). Many of these studies also provide recommendations on overcoming the barriers, especially to attract women into computing. In our minds, one of the key finding has been the alignment of course content to student

interests to increase student engagement that can have a positive impact on students choosing to enter computing as a major in college (Bair and Marcus 2007, Akbulut and Looney 2007).

Introductory computing courses in the undergraduate courses serve as a gateway into the computing curriculum. It is therefore imperative to pay special attention to the design of these courses. Our approach represents an attempt to provide interesting and diverse range of examples and exercises where the focus is on the context of the applications (robot behaviors in most cases) and not on the specific programming features one has to master. The latter are a side-effect of this activity. This provides an innovative pedagogical approach and it challenges students in unique ways. Most of the tasks assigned to the students are attainable and provide a basis for supportive and positive feedback to students. These factors have been identified as significant parameters that can lead students to pursue further studies in computing (Akbulut and Looney 2007). It has also been identified that exposure to creative computer applications is essential to compensate for high school experience in computing that mostly involves pedestrian uses of computers (Bair and Marcus 2007).

Summary

In this paper, we have described how we have taken advantage of the engaging aspects of AI robotics concepts to motivate students in a CS1. The curriculum deviates from traditional approaches to CS1, and yet provides a comprehensive and engaging treatment of traditional CS1 concepts. In many ways, the curriculum goes beyond the traditional notion of a CS1 syllabus. Yet, the key driving factor in the design of this curriculum is the exploratory and engaging nature of robots. We believe that the use of personal robots and engaging examples can provide a sound foundation for learning computing and also serve to attract a more diverse body of students into the computing disciplines.

Acknowledgements

This work is being supported by grants from Microsoft Research, Georgia Institute of Technology, and Bryn Mawr College.

References

Akbulut, A. Y. and Looney, C.A. 2007. Inspiring Students to Pursue Computing Degrees. *Communications of the ACM (CACM)*. Volume 50, Number 10. October 2007.

Arkin, R. 1998. *Behavior-Based Robotics*. MIT Press.

Bair, B. and Marcus, M. 2007. Women's Interest in IT: The Fun Factor. In (Burger *et al* 2007).

Blank, D; Kumar, D.; Marshall, J.; and Meeden, L. 2007. Advanced Robotics Projects for Undergraduate Students. In *Robots and Robot Venues: Resources for AI Education*. AAAI Spring Symposium Series Technical Report SS-07-09. 10-15. AAAI Press.

Blank, D; Kumar, D.; Meeden, L.; and Yanco, H. 2006. The Pyro Toolkit for AI and Robotics. *AI Magazine*, 27(1):39-50.

Blank, D. 2006. *Robots Make Computer Science Personal*. Communications of the ACM, Volume 49, issue 12, pages 25-27. New York: ACM Press. [CITE THIS]

Bird, S. and Loper, E. 2004, NLTK: The Natural Language Toolkit. In *Proceedings of the 4th International Conference on Natural Language Processing (ICON)*, 11-18. Allied Publishers.

Braitenberg, V. 1984. *Vehicles: Experiments in Synthetic Psychology*. MIT Press.

Brooks, R. 1986. A robust layered control system for a mobile robot. *IEEE Journal of Robotics and Automation* RA-2(1):14-23.

Burger, C.; Creamer, E.; and Meszaros, P (editors). 2007. Reconfiguring the Firewall: Recruiting Women to Information Technology across Cultures and Continents. AK Peters.

Computing Curricula 2001. Journal on Educational Resources in Computing. Volume 1, Issue 1. ACM Press.

Guzdial, M. 2005. Introduction to Computing and Programming Python: A Multimedia Approach. Pearson Prentice Hall.

IPRE. 2007. *Learning Computing with Robots*. Text in development. wiki.roboteducation.org.

Margolis, J. and Fisher, A. 2003. *Unlocking the Clubhouse*. MIT Press.

Martin, F. 2007. Real Robots Don't Drive Straight. In *AAAI Spring Symposium, Robots and Robot Venues: Resources for AI Education*.

Vegso, J. 2005. Interest in CS as a major drops among incoming freshmen. *Computing Research News 17*, 3 (May 2005); www.cra.org/CRN/articles/may05/vegso.

Walker, A. 2007. Natural Language Interaction with Robots. http://wiki.cs.brynmawr.edu/?page=Natural Language Interaction with Robots. Senior Thesis. Bryn Mawr College.