The Title of our research is I-ECS Inclusive Exploring CS Curriculum Enhancement as Face-to-Face and Online Support for Visually Impaired, High School Students.

Participants: Caril Carrillo, Danniel Fehrenback, Lindsey Ellis, Scott Jordan, Stephanie Ludi, Tommy Suriel

Abstract

(R.I.T) Rochester Institute of Technology, proposes the Inclusive Exploring Computer Science (I-ECS) curriculum as a set of curricular and tool-based enhancements for visually impaired high school students. I-ECS will be conducted via face-to-face and online support for cohorts of visually impaired high school students. While innovations have resulted in assistive technology for people with disabilities, gaps remain in the level of participation in university-level computing degree programs by the visually impaired. A significant factor is the lack of precollege preparation in courses that promote success in computing degrees. Isolation, the lack of role models and access to resources contribute as well.

The proposed project strives to increase the participation of students with visual impairments through better preparation and support. The overall goals are to increase access to an established curriculum (Exploring Computer Science) and success within the area of computing for students with visual impairments at the high school level. The development of programming tools and online modules will provide access where gaps exist in current software. The project team will assess the degree to which original Exploring Computer Science (ECS) can be made accessible to high school students who are visually impaired. The comparison in student achievement and attitudes about Computer Science will also be conducted between students using the ECS curriculum and those using the I-ECS curriculum in order to evaluate the similarity of the two curricula. By design, the I-ECS project intends to increase participation in computing by individuals with visual impairments at all socioeconomic levels via student and educator support. The project will involve students, parents and educators from around the nation. The project will be evaluated using student and parent surveys, student interviews, student projects made through I-ECS, and system data collected on the I-ECS website (e.g. times and duration of use, online collaboration and interaction).

Introduction

“Estimates vary as to the number of Americans who are blind and visually impaired. According to one estimate, approximately 10 million people in the United States are blind or visually impaired. Other estimates indicate that one million adults older than the age of 40 are blind, and 2.4 million are visually impaired. Over the next 30 years, as the baby-boomer generation ages, the number of adults with vision impairments is expected to double. Recent figures also indicate that only 46% of working-age adults with vision impairments and 32% of legally blind working-age adults are employed.”[1]

This project helps the visually impaired student to be better in computer science. We found the best softwares used nowadays to help the visually impaired use computers. We found

softwares, like Jaws, iSonic, Voice Over, Jbrick, all these softwares are free and help people be in the same side of normal people.

The activities in the Curriculum help the students understand very important concepts in the field of computer science. At the beginning we have the students perform activities that do not require the use of computers to understand computer science concepts. Later we have the students use the different softwares named above to do other activities regarding the field of computer science.

We expect that the students with these activities and softwares gain a very good knowledge of what is computer science and its basic concepts.

Related Work

"An ACE for Visually Impaired Students in Computer Science

Enter Project Accessible Computing Education (ACE), an NSF-funded initiative at the Rochester Institute of Technology (RIT). The project is designed to help prepare visually impaired middle school and high school students participate in computer science programs at the collegiate level. Project ACE's ultimate goal is to increase the number of visually impaired students pursuing degrees in computer science and give them the foundations they need to be fully successful in their studies and beyond.

The project is focused on three areas: better preparation for visually impaired students before college, support for these students as they face challenges in computing that other students do not, and educating teachers in how to best help these students learn and achieve".[2]

Approach

First we started with the CS Unplugged activities which do not require the use computers.

Activity 1: Introduction to iSonic and Sonification.

This activity makes use of iSonic (http://www.cs.umd.edu/hcil/audiomap/), a program that uses sonification to display geographic data to blind users. It can be used by both sighted and visually impaired users. It is designed from the ground up to be completely accessible.It will only work on Windows.

A drawing tablet is helpful on the last step of the tutorial. It must be set up so that the drawing surface is proportional to the screen (e.g. touching the same spot on the tablet always touches the same spot on the screen) and to avoid inadvertent right-clicks or zooms. Here we detail the instructions for a Wacom Bamboo tablet.

If the tablet is not sensitive over the entire tablet area use something tactile to mark off the boundaries of the sensitive area so the students know where the edges and corners of the sensitive area/screen are. Wikkistix can be attached and removed easily. The following is a guided tutorial for students to familiarize themselves with the tool, learn how sonification can display data, and at the end use the tool to find information about the United States. Advise the students to not use the help menu on F1 (it doesn't seem to work correctly), but to feel free to use F10 to browse the menu and check out the different commands and shortcuts.

Activity 2: Preparation of Peanut butter Jelly sandwich

In this activity we have the students give instructions to the mentors to prepare a peanut butter Jelly Sandwich. The students assume that the mentors are like computers or robots and the mentors are supposed to act as if they were robots or computers doing exactly what they student instructs. This activity helps the students understand all the details that a programmer have to keep in mind when programming.

Activity 3: Act out Turing Test

The AI debate hinges on a definition of intelligence. Many definitions have been proposed and debated. An interesting approach to establishing intelligence was proposed in the late 1940s by Alan Turing, an eminent British mathematician, wartime counterspy and long-distance runner, as a kind of "thought experiment." Turing's approach was operational—rather than define intelligence, he described a situation in which a computer could demonstrate it. His scenario was similar to the activity described above, the essence being to have an interrogator interacting with both a person and a computer through a teletypewriter link (the very latest in 1940s technology!) If the interrogator could not reliably distinguish one from the other, the computer would have passed Turing's test for intelligence.

For this Activity, we have two of the mentors to hide in a certain room. One represents a human and the other one a robot. The Students give questions to these mentors to answers. The questions are about random topics. After the mentors gave their answer the students have to guess which one is machine and which one is human.

The questions are

1. What is the name of Bart Simpson's baby sister?
2. What do you think of Roald Dahl?
3. Are you a computer?
4. What is the next number in the sequence 3, 6, 9, 12, 15?
5. What do you think of nuclear weapons?
6. What is 2 × 78?
7. What is the square root of two?
8. Add 34957 to 70764.
9. Do you like school?
10. Do you like dancing?
11. What day is it today?

12. What time is it?
13. How many days are there in February in a leap year?
14. How many days are there in a week?
15. For which country is the flag a red circle on a white background?
16. Do you like to read books?
17. What food do you like to eat?

Activity 4: Candy bar Activity

Divide the students into groups of 2 or 3. Give each group a candy bar.
Explain that their task is to determine how many "breaks" it will take to break the candy bar into 12 equal pieces. One break of one piece of the candy bar will result in that one piece being divided into two pieces. Demonstrate a "break" by breaking the bar into two pieces. Every time a piece of chocolate is broken, that counts as a separate break- even if you stacked the pieces together.
At this point, have each student write in their journal the number of breaks they think it will take to break the bar into 12 equal pieces. This should be done without talking to their partner or group members.
Working together with their partner or group, have the students discuss and then write their plan for solving the problem. They may revise their guess at this point.
Once this is completed, the students should implement the plan by opening the candy, breaking the candy, and counting the number of breaks it takes to get 12 equal pieces.

The purpose of this activity is to introduce the students to the concept of Sorting Algorithms.

Activity 5: Handshake activity

Students follow the next instructions:

Handshake Problem: Assume there are 20 people in a room, including you. You must shake hands with everyone else in the room. How many hands will you shake? If there are N (where N > 0) people in the room, how many hands will you shake?

Have the people line up in the room. The first person in the line walks down the line and shakes hands with all of the people in the line and then leaves the room. Count the number of handshakes and add to the total (Have students count off).
The next person in line walks down the line and shakes hands with all of the people left in the line and then leaves the room. Count the number of handshakes and add to the total. This continues until there are only 2 people left. They shake hands and leave together. Increase the total by one.

Once the answer is known for 10 people, look for a pattern. Try the process for 5 people, 2 people. See if the pattern holds.

Carry out the plan: Using your plan, show your work and your solution.

Now add up the number of handshakes: 9 + 8 + 7 + 6 + 5 + 4 + 3 + 2 + 1 + 0 = 45

The purpose of this activity is to teach the students the concept of Summation which is used a lot in computer science.

Activity 6: Towers of Hanoi Puzzle

We gave the students a brief description of what the towers of hanoi puzzles is like:
An old legend tells of a Hindu temple where the pyramid puzzle might have been used for the mental discipline of young priests. The legend says that at the beginning of time, the priests in the temple were given a stack of 64 gold disks, each one a little smaller than the one beneath it. Their assignment was to transfer the 64 disks from one of three poles to another, with one important rule: a large disk can never be placed on top of a smaller one. The priests worked very efficiently, day and night. When they finished their work, the myth said, the temple would crumble into dust, and the world would vanish.
In 1883, Edouard Lucas, a French mathematician, invented a game called the Tower of Hanoi (sometimes referred to as the Tower of Brahma or the End of the World Puzzle). The game begins with a number, for example of 3 discs, arranged on one of three poles. Each disc is smaller than the disc below it. The object is to move all the discs from the starting tower to one of the remaining towers. Only one disc can be moved at a time, and a larger disc can never be placed on top of a smaller one. Use the lowest number of possible moves.

The following table records the minimum number of moves required for the number of original discs.

Number of Discs	Number of Movements
3	7
4	15
5	30

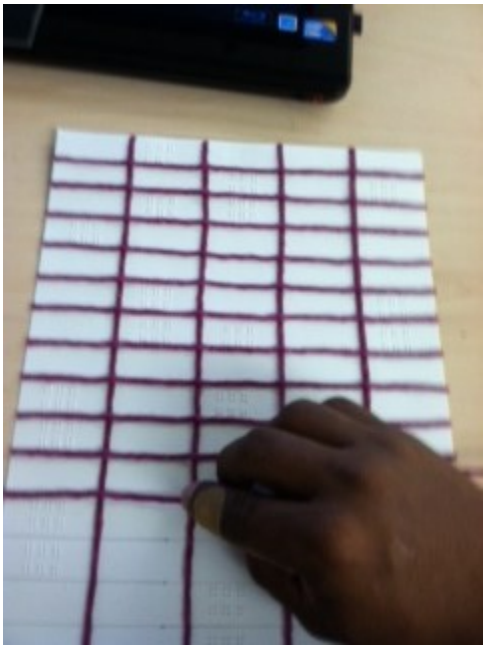
This lesson reinforces the four main phases in the problem-solving process.
Objectives: The students will be able to:
Solve a problem by applying the problem-solving process.
Express a solution using standard design tools.
Determine if a given solution successfully solves a stated problem.
Student Activities:
Work individually to learn the history and rules of the Towers of Hanoi puzzle.
Discuss how long this puzzle might take to complete for different sizes
Work with elbow partner to complete the activity.
Discuss solutions.

Activity 7: Unplugged Counting in Binary

This lesson introduces the binary number system and how to count in binary. Students will learn how to convert between binary and decimal numbers in the context of topics that are important to computer science.
Objectives:
The students will be able to:
Count forward and backward in binary.
Explain why binary numbers are important in computer science.
Use binary digits to encode and decode messages.

Worksheet Activity: Working With Binary The binary system uses zero and one to represent whether a card is face up or not. 0 shows that a card is hidden, and 1 means that you can see the dots.
For example: 010001 = 9 Can you work out what 10101 is? What about 11111? What day of the month were you born? Write it in binary.
Find out what another student's birthday is in binary.
Using the symbol key document and the symbol cards that you were given, determine what the values are. You will also need to use the morse code sound files on a computer. Extra for Experts: Or you could surprise an adult and show them how they only need a balance scale and a few weights to be able to weigh those heavy things like suitcases or boxes!

Using a set of rods of length 1, 2, 4, 8 and 16 units show how you can make any length up to 31 units. To do this activity , we build a paper with a grid drawn in it using wikkistix.



Activity 8: Model Binary Search

For this activity we use two identical, *sorted*, and *large* stacks of index cards with words in braille and writing. Give one stack to a pair of two students and ask them to pick any word, keeping it in order.
Choose another pair of students to count how many times you pick a word from the stack to try and find their word.
Start by using a linear search. It should not take long for students to suggest that this is not a good strategy. Ask them to provide a better strategy.
Guide them to binary search.
Discuss the number of guesses required and how this is similar to the tower building problem.
Comparison of linear and binary search.
Linear—start at the beginning, look at each item until you find it or there is no more data. Data can be sorted or not.
Binary—look at middle item, eliminate the half where the value is not located. Find the new middle element and continue the process until you find it, or there is no more data. Ask students to describe what is necessary in order to use a binary search—the list must be sorted.

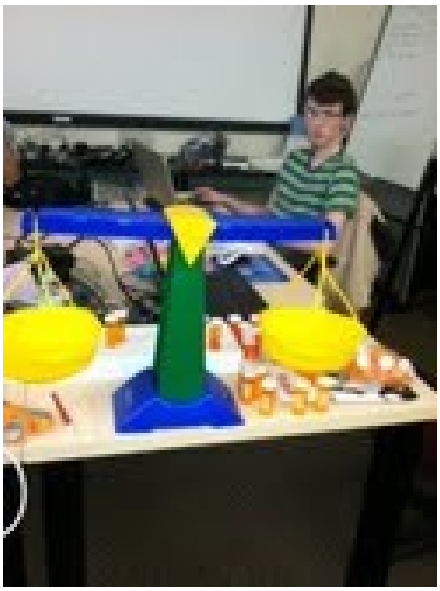
Have students provide examples of where each type of search is appropriate and why.

Activity 9: CS unplugged: Lightest & Heaviest (explore sorting)

Instructions for the students to follow:

Aim: To find the best method of sorting a group of unknown weights into order. You will need: Candy , 8 identical containers, a set of balance scales. What to do:
Fill each container with a different amount of Candy. Seal tightly.
Mix them up so that you no longer know the order of the weights.
Find the lightest weight. What is the easiest way of doing this? Note: You are only allowed to use the scales to find out how heavy each container is. Only two weights can be compared at a time.
Choose 3 weights at random and sort them into order from lightest to heaviest using only the scales. How did you do this? What is the minimum number of comparisons you can make? Why?
Now sort all of the objects into order from lightest to heaviest.
When you think you have finished, check your ordering by re- weighing each pair of objects standing together.
Selection Sort One method a computer might use is called selection sort. This is how selection sort works. First find the lightest weight in the set and put it to one side. Next, find the lightest of the weights that are left, and remove it. Repeat this until all the weights have been removed. Count how many comparisons you made. Extra for Experts: Show how you can calculate

mathematically how many comparisons you need to make to sort 8 objects into order. What about 9 objects? 20?

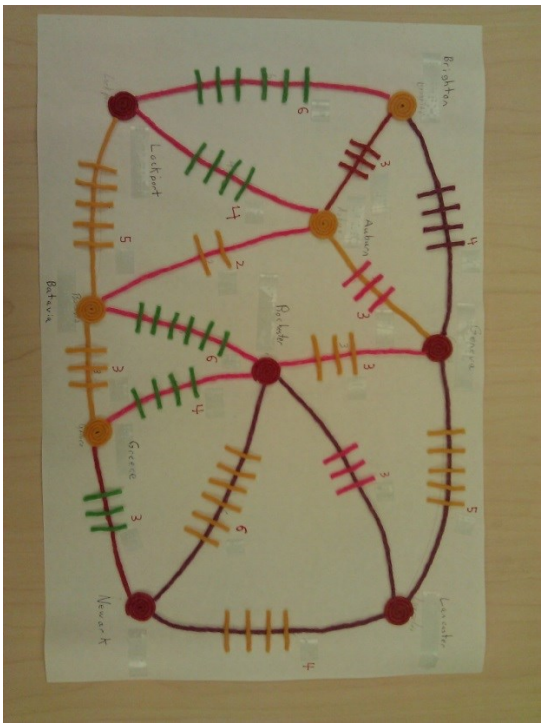


Activity 10: Minimum Spanning Tree

Sometimes a small, seemingly insignificant, variation in the specification of a problem makes a huge difference in how hard it is to solve. This activity is about finding short paths through networks. Here we are allowed to introduce new points into the network if that reduces the path length.

The goal is for students to be actively involved in some way and for all students to be able to describe shortest path strategies. What follows is the minimal suggestion.

For visually impaired students we use a tactile graph on paper that allows graph information to be read in printed text and braille as well as providing information through touch by using wikkistix. A sample is included below. The “houses” or cities are represented as circles/spirals with their names written in print and braille next to them. Edges between cities are straight lines made of wikkistix. Their weights are represented by the number of bars going across them, by a printed number, and a braille number near the bars.



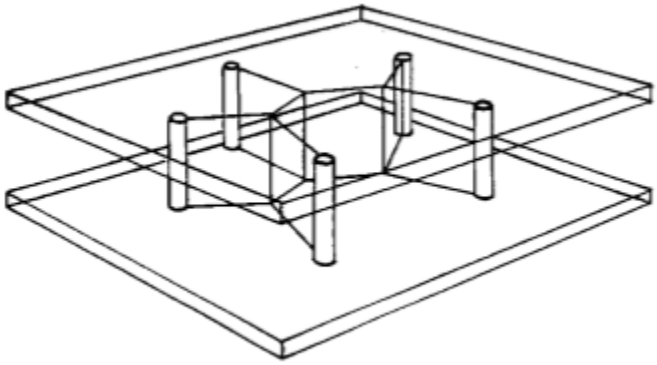
Activity 11: Steiner Tree activity

Steiner trees are much harder because you have to decide where to put the extra points. In fact, rather surprisingly, the difficult part of the Steiner tree problem is not in determining the precise location of the Steiner points, but in deciding roughly where to put them. Soap films do that very effectively, and so can computers. Surprisingly, finding minimal Steiner trees can save big bucks in the telephone business.

The networks that we've been talking about are minimal Steiner trees. They're called "trees" because they have no cycles, just as the branches on a real tree grow apart but do not (normally) rejoin and grow together again. They're called "Steiner" trees because new points, Steiner points, can be added to the original sites that the trees connect. And they're called "minimal" because they have the shortest length of any tree connecting those sites. It's interesting that while there is a very efficient algorithm for finding minimal spanning trees a greedy one that works by repeatedly connecting the two closest so- far-unconnected points—there is no general efficient solution to the minimal Steiner problem.

Another interesting activity is to construct soap-bubble models of Steiner trees. You can do this by taking two sheets of rigid transparent plastic and inserting pins between them to represent the sites to be spanned. Now dip the whole thing into a soap solution. When it comes out, you will

find that a film of soap connects the pins in a beautiful Steiner-tree network.



In this activity the students will use chairs to represent the different pins and a string to connect them, they use scissors to use as much string as they thought they needed to find a solution.

Simulations and Experimental Setup and Results

Activity 12: Walk like a Robot

Instructions:

Choose one student to be a "robot" or tell students that you will be the robot. Choose a starting point and an ending point between which the "robot" must navigate. Make sure the path is not direct.

Tell the class that they must direct the robot from the starting point to the ending point using only five commands:

- Turn left 90 deg.
- Turn right 90 deg.
- Take a step forward with the left foot.
- Take a step forward with the right foot.
- Stop.

Students can take turns or work as a group. The robot should only follow those five commands and not respond to other commands. Tell students to be careful with the robot and not walk it into walls or barriers. (The robot should stop before it hits a barrier such as a wall).

At some point, remind students about loops. They can tell the robot to repeat a command or a block of commands such as "repeat: take a step forward with the left foot, take a step forward with the right foot until you are at the wall"

Point out that this is frequently what is done in dancing and choreography—sequences of steps are repeated.

Activity 13: Programming of NXT Robots

- Introduce the features of the JBrick Mindstorms NXT Software
- The students will be able to Recognize the parts of the Mindstorms NXT software.
- Explain the different types of icons in the common palette and how to use them.
- Explain the different types of icons in the complete palette and how to use them.
- Recognize the parts of the JBrick software.
- Explain the difference between software errors and hardware errors.
- Explain the difference between logical errors and syntax errors.
- Interface: the parts of the JBrick Mindstorms NXT software
- Follow Project Zero document.
- A simple program from the complete palette
- Student Activities:
- Discuss how the programs were created in the NXT brick and how they behaved compared to expectations.
- Listen to explanation of Mindstorms NXT software and respond to questions.
- Give ideas to teacher as s/he writes small programs in the software.
- Learn how to use JBrick Software
- Begin the Project Zero activities.
- Teaching/Learning Strategies:
- Ask students what they programmed the robot to do. Get several answers. Did it do what they expected? Why or why not? Would it be a good idea to use the JBrick NXT Program interface to write all their programs? Why not? (It can only take 5 commands in a program.)
- Projecting the teacher's screen, launch the Mindstorms NXT JBrick software. Show the students where the tutorials are in the Robot Educator section and how to open a new program. Describe all parts of the interface.
- With student input, use the common palette to build a small program. Ideally, use a variety of the blocks of the common palette, explaining what each one does as you use it. For example, if you wanted to build a program that told the robot to wait until the touch sensor was touched, then move forward for one rotation then listen and if a loud sound occurs, then display a smiley face and play a sound otherwise move forward, it would look like this:
- With student input, build a small program. Ideally, you should use several different features of the NXT system such as using the different sensors and motors, as well as producing audio.
- Save the program and download it to an NXT brick. Make sure the brick is set up to do the actions—have one built with the driving base and any necessary sensors. Demonstrate the running of the program.
- Modify the program and download it again. Try to make mistakes during this period and show how to debug the program by frequently testing it, downloading extra blocks, and also making mistakes such as having disconnected blocks. During this part have students try to work with the software themselves and follow along with you.

- Open a new program and switch to the complete palette. Show the differences in the two palettes. With student input, write a new program using the blocks of the complete palette. Show the differences in controlling the program. Make sure to show how to wire things in the data hub. For example, a program that runs the motors for a random amount of time would look like this:
- Make sure to make mistakes and demonstrate how to solve problems with the software such as mis-wiring ports. Have students try these features at their seats as you do it. Point out the similarities between programming the NXT software and what they did in the last unit with Scratch.

Project Zero

For the following activities, in each of the codes the students use two libraries: MotorUtilities.NXC and SensorUtilities.NXC which were created to help the students understand better the functions commonly used in the language of nxc. In these libraries most of the basic functions defined in the language of NXC are redefined with a different name which is considered easier for the students to understand.

Instructions for the students:

The requirements for the activity models and programming are the following for each team:

- 1 Lego Mindstorms Education kit
- 1 Lego Education kit
- JBrick (http://code.google.com/p/jbrick/)
- NXT firmware installed, version 1.05 or later (http://mindstorms.lego.com/en-us/support/files/Firmware.aspx)
- Project Zero Student Activity write-up
- Student Cheat sheet
- Materials for challenges

To run this activity, the following Lego model components should be built for or by each team:

- 2-Motor Base.lxf
- 2-Wheel Caster.lxf
- Sensor Chassis – Arm.lxf
- Sensor Chassis – Front.lxf

These will be combined into the following models for each activity:

- Activity Base.lxf
- Straight Model.lxf
- 2 Basic Activity Notes

The edit, compile, download and run sequence is repeated each time a programming change is made. The students will pick this up fairly quickly, but you will need to walk them through it the first few times. Common errors and things to be aware of:

- Make sure the student is saving their work (i.e. MotorActivity1-1.nxc) back into the Project Zero folder where they opened MotorActivity.nxc and are using the NXC (*.nxc) "Save as type" option in the Save Current File dialog.
- The USB cable not being attached when attempting a download
- The USB cable is attached, but the NXT brick is not connected (Tools->Find Brick) File is not being saved with an *.nxc extension (Save as type)

The Download command automatically saves and compiles the program before downloading
A downloaded program of the same name replaces an existing program on the brick. Repeat the download prior to debugging a program to make sure the BricxCC program and NXT program are in synch.

Challenge Activities

Below are solutions to the different challenges that the students will get to accomplish. With each solution are tips on how to help the students if they get stuck, what common problems students may face, and what mistakes may occur in student implementations.
These are by no means the exact solutions that students should achieve, as there are many ways to accomplish each activity.

Challenge #1 - Bump'n Turn

Objective: To create a robot that will move forward until it runs into something. It will then back up and turn, then continue moving forward.
Setup: This robot uses the same chassis that you've been using throughout the activity. On the front of it is the touch sensor. You may want to stick an axle with a gear on the end into the sensor to provide a larger touch area. You will use this touch sensor to determine when the robot has run into something.

Bump'n Turn Solution

```
/* BumpnTurn.nxc */
// Include utilities for motors and sensors
#include "MotorUtilities.nxc"
#include "SensorUtilities.nxc"
// The entry point for your program
task main()
{
  // Configure the touch sensor
  ConfigSensor(S1, SENSOR_TYPE_TOUCH, SENSOR_MODE_BOOL);

  // Initialize variables for storing sensor values
  bool sensorValue = false;

  // In a continuous loop
  while(true)
```

```
  {
    // Read sensor values
    sensorValue = CheckSensor(S1);

    // If we touch something, move backwards and turn
    if(sensorValue)
    {
      // Coast
      Off(MOTOR_AC);

      // Go backwards
      RotateMotors(MOTOR_AC, 60, 90);

      // Turn 90 degrees
      RotateMotors(MOTOR_AC, 60, 180, -100);

      // Clear the outputs
      Coast(MOTOR_AC);

      // Clear the value
      sensorValue = false;
    }

    // Move forward
    MotorsForward(MOTOR_AC, -75);
  }
}
```

The primary problem areas that come up are as follows:

- Not recognizing the need to check the sensors value continuously in the loop.
- Not Coasting or Offing the motors when changing from going forwards to reverse, rotating, or the like.
- For solutions that turn using MotorsForward or MotorsReverse, not calling Wait to make sure the motors get a chance to move for a designated period of time or distance.
- Configuring the Touch Sensor in any mode other than SENSOR_MODE_BOOL can result in it being difficult accurately tell when the sensor is pressed in or not.

Challenge #2 - Avoidance

Objective: Program the NXT robot to move forward and turn when it comes close to an object.
Setup: You will be using the same robot from the previous activity. Simply replace the touch sensor with an ultrasonic sensor.
Open the file Avoidance.nxc. It contains an outline of the program you will complete by following the hints given in the commented areas.

Avoidance Solution

```
/* Avoidance.nxc */
// Include utilities for motors and sensors
#include "MotorUtilities.nxc"
#include "SensorUtilities.nxc"
// Any defines you need
#define THRESHOLD 50
// Main task
task main()
{
  // Configure the ultrasonic sensor
  ConfigSensor(S1, SENSOR_TYPE_LOWSPEED_9V, SENSOR_MODE_RAW);

  // Initialize variables for storing sensor values
  int ultraValue = 255;
  int turnRatio = 0;

  // In a continuous loop
  while(true)
  {
    // Read sensor values
    ultraValue = CheckSensor(S1);

    // If we see something, turn until we don't anymore
    if(ultraValue <= THRESHOLD)
    {
      // Halt
      Off(MOTOR_AC);

      // Turn
      RotateMotors(MOTOR_AC, -75, 23, 100);

      // Clear the motors
      Coast(MOTOR_AC);
    }
    // Otherwise, move forward
    else
    {
      // Go forward
      MotorsForward(MOTOR_AC, -75);
    }
  }
}
```

Common problem areas:

- Not recognizing the need to check the sensors value continuously in the loop.

- Not Coasting or Offing the motors when changing from going forwards to reverse, rotating, or the like.
- For solutions that turn using MotorsForward or MotorsReverse, not calling Wait to make sure the motors get a chance to move for a designated period of time or distance.
- Configuring the Ultrasonic Sensor in any mode other than SENSOR_MODE_RAW can result in it being difficult to ascertain the real distance between the sensor and an object.
- Having the motors always try to move forwards in the loop unconditionally. Unlike the previous challenge, this is necessary to insure that the robot does not run into anything. Otherwise, the robot will lurch forward followed by braking and turning, which will wear on the motors and drain battery power quickly.
- Having either too large or small a THRESHOLD value can make it hard for the robot to behave correctly. The ultrasonic sensor can detect accurately from 10cm to around 100cm out. Anything outside those boundaries can become problematic.

Challenge #3 - Going the Distance

Objective: Program the NXT robot to move forward towards an object from a fixed starting point. Use the movement of the robot to measure the distance between the starting point and the object, while insuring that that robot stops when it reaches the object.
Setup: You will be using the same robot from the Bump'n Turn activity. Furthermore, to check the distance, you will need to know the circumference of your wheels. You will be given some code for special functions that you will need to use, provided below.

Going the Distance Solution

```
/* GoingTheDistance.nxc */
// Include utilities for motors and sensors
#include "MotorUtilities.nxc"
#include "SensorUtilities.nxc"
// Any defines you need
// Entry point to your program
task main()
{
  // Reset the rotation counters on the motors
  ResetRotationCount(MOTOR_AC);

  // Configure the sensors
  ConfigSensor(S1, SENSOR_TYPE_TOUCH,
               SENSOR_MODE_BOOL);

  // Declare variables for the sensors and motors
  bool touchValue = false;
  int numRotations = 0;
  int rotationCount = 0;

  // In a continuous loop
```

```
  while(true)
  {
    // Get the current rotation count
    rotationCount = -MotorRotationCount(OUT_A);

    // Read the sensor values
    touchValue = CheckSensor(S1);

    // If we've gone a new rotation
    if(rotationCount >= 360)
    {
      // Increment the number of rotations
      numRotations++;

      // Reset the rotation count
      ResetRotationCount(OUT_AC);
      // Play a tone
      PlayTone(100, 500);
    }

    // If the touch sensor is hitting something
    if(touchValue)
    {
      // Stop the motors
      Off(MOTOR_AC);

      // Break out of the loop
      break;
    }

    // Otherwise, move forward
    MotorsForward(MOTOR_AC, -50);
  }
}
```

Dancing Robot

In this activity, students work in pair to program a robot to dance or move according to a music of their choice. For this activity and the rest of the robotics activities they use the libraries from project zero.

Follow the line and hit the ball

In this activity, students work in pair to program a robot to follow a black straight line over a white surface using two light sensors. After the robot reaches the end of the line make the robot

hit a small plastic ball . For this activity the students have to design the robot so that it hit the ball as if it were holding a baseball bat. They also use the project zero libraries.

Simulations and Experimental Setup and Results

The first thing that we did was to format and adapt the Inclusive Exploring computer science curriculum so that the activities in it are accessible to the visually impaired. The lab was equipped just with I-Macs, therefore we had to partition the computers to use windows and mac because there are some softwares like JAWS, iSonic and Jbrick that were only runnable on windows.

We had to download each of these softwares and install them in each computer. We had to download their drivers to make them run.

Later we built each of the robots that the students used and disassembled them. We stored the disassembled robots in different bags to make it easier for the students to reassemble them. We also simulated each of the activities the students had to perform with the robots.

Setup

We organized each of the activities from the curriculum by unit. Then we organized the student files and put them in each of the computers the students were going to use along with the softwares. We organized all the physical material that they used by units (eg. towers of Hanoi, Minimal Spanning tree map, Binary code cards etc..). We move all the computers to another lab in which they did the first half of all the activities; they had more space in this lab.

The I-Macs that we used have the following characteristics:

- 2.7 GHz quad-core Intel Core i5 processor (Turbo Boost up to 3.2GHz) with 6MB L3 cache
- 1 TB (5400-rpm) hard drive, 8 GB (two 4GB) of 1600MHz DDR3 memory
- 21.5-inch (diagonal) LED-backlit display with IPS technology; 1920-by-1080 resolution
- NVIDIA GeForce GT 640M graphics processor with 512MB of GDDR5 memory
- Mac OS X Mountain Lion

Results

Activity 1: Introduction to iSonic and Sonification.

The students learned how to use the software iSonic. They learned the different ways in which they can get information about the states using the program. They learned specific information about the amount of people that lives in each state that are visually impaired. They also learned how sort the information in the program to find out what state has the highest amount of visually impaired people by age and other different classifications. The only issue about this software is that it can only be used in the Windows operating system.

Activity 2: Preparation of Peanut butter Jelly sandwich

In this activity, since the students didn't have much experience in programming, they didn't do very well. When writing the instructions they didn't take into account many details that a computer has known about before executing. For example most of them didn't as for the peanut butter jar to be open before spreading it over the bread.

Activity 3: Act out Turing Test

In this activity, they did pretty well. They were able to Identify very quickly which one was the robot.

Activity 4: Candy Bar Activity

At the beginning they did pretty bad. They had no Idea of how to start. With the guidance of the mentors they all were able to think of good algorithm to solve to problem. However each of their algorithms were different.

Activity 5: Handshake activity

They did pretty good, it was very easy for them. We later explained to them what was the concept behind it and they understood immediately.

Activity 6: Towers of Hanoi

Since this activity starts with a low amount of disks, they were able to learn the algorithms pretty quiclky and solve the problems fast enough.

Activity 7: Unplugged Counting in Binary

The students did pretty good. They all learned how to count in binary very quickly.

Activity 8: Model Binary Search

At the beginning they didn't understand what the concept in this activity was. After we explained to them the concept of binary search, they were able to do the activity a lot quicker because they understood the concept.

Activity 9: CS unplugged: Lightest & Heaviest (explore sorting)

In this activity, they started very well using the binary search concept. Later we introduced them to the different sorting concepts which they also used to solve the problem and understood very well.

Activity 10: Minimum Spanning Tree

They did very well, they found the optimal solution pretty fast.

Activity 11: Steiner Tree activity

They didn't do very well at the beginning because the had never learned the concept before. After we explained to them the concepts and techniques that had to be applied in this activity they did much better.

Activity 12: Walk like a Robot

They didn't do well in this activity because it took them more than five commands. It seems that for this activity they didn't read well the instructions. The students didn't take into consideration things like obstacles or necessary steps.

Activity 13: Programming of NXT Robots

Dancing Robot

In this activity the students didn't do very well at the beginning because they didn't know how to adjust the depth of the sound sensor. For example sometime the robot would move with the sound of its own motors. Due to this, they stopped using the sound sensor and simply tried to make robot move according to the sound of music and its timing.

Follow the line and hit the ball

In this activity at the beginning they did pretty bad. Later on, just one of them was able to complete the task very well. The same student was the only one able to make the robot follow a curved black line. Later the rest of the students were able to make the robot follow the line in a straight line. Finally, they did pretty good in having the robot hit the ball at the end of the line.

Conclusion

In conclusion, we can say that in most of the activities the students did very well. This project helped the students learned very well some of the most important concepts in computer

science. They only part in which they had issues was in programming which as usual is the most difficult area in the field of computer science.

Bibliography

[1] Questions and Answers About Blindness and Vision Impairments in the Workplace and the Americans with Disabilities Act. http://www.eeoc.gov/facts/blindness.html

[2] An ACE for Visually Impaired Students in Computer Science. http://www.nsf.gov/news/news_summ.jsp?cntn_id=112729

http://www.exploringcs.org/curriculum