Lecture 01

- Objectives, Expectations and Place of 201 in our menu of intro. courses
- Textbook reading guide
- Program a sequence of print outs.
- Program a sequence of Turtle drawing strokes.
- Making it work yourself
- Plan of first 2 projects and 3 labs
- (If time, what we did)

Objectives, Expectations and Place of 201 in our menu of intro. courses

Chief Learning Goal

Given a description or your own dream of what a computer might do, YOU WRITE/CREATE a program that demonstrably makes the computer do it.

Expectations

- Multimedia Computing Curriculum
  - Georgia Tech; Computer Sci. Education Research
  - Your programs make cool visual effects RIGHT AWAY!

    Relate computing with numbers to today's digital multimedia
  - The theme of LOCATION
    - **Computing with data that says WHERE something, typically other data, can be found.**
      - You will work with X,Y coordinates for LOCATIONS WHERE in a DIGITAL PICTURE, colors making up the picture can be found. Arrays (a difficult topic) are taught early. A digital picture IS a 2-dimensional array!

Intro. reading:

The whole preface, INCLUDING the material for teachers.

explains/motivates content and approach—what you might get out of the course, the book and why.

But you will not USE that content directly.

SO: START PROJ. 1 FIRST!!

1. We and the book try to explain “concepts” just when they are needed, not bundle the concepts first, then apply them together later.
2. Arrays are covered VERY EARLY (Ch. 3)

Project 1 reference reading

See the pages listed in the assignment!!
You MUST USE some of the
DETAILS
written on those pages, and given in lectures too.

Spelling and punctuation
REALLY REALLY MATTER!

`tu.turn(90);` NOT **`tu.Turn( 90 ).`**

```
tu.turn( 90 );
tu.Turn( 90 ).
```

What are the two differences?

WRITE and BOOKMARK:

`http://www.cs.albany.edu/~sdc/CSI201`

It's the Universal Resource LOCATOR that locates the course Web site.

It has no useful information, but WHERE on the WWWeb the info you will need can be copied from.

Advanced and internal operations of computing
involve data (like an array index) that says
WHERE something, typically other data,
can be found.

Needed materials

- Calculator: ONLY SOMETIMES (usually not)
- BRING IN A SIMPLE CALCULATOR ON WEDNESDAY
- So you can imitate the Rosie Computers of WW II.

Needed materials

- iClicker (Version 1 OK)
- Guzdial&Ericson textbook:
  - You don't need and SHOULDN'T USE the CDROM.
  - Shared/used textbooks OK
  - Exams will NOT ALLOW BOOKS (1 sheet notes only) so sharing isn't a problem
- (BY MONDAY!) Way to use computers for lab follow up homework, experimentation, project assignments and submit work to Blackboard.

Expectations

- We assess this (and most other college level CS major subjects) by having you

Demonstrate SKILLS:

This course has very little memorization of facts or “concepts”
or questions asking you to
“demonstrate understanding”
by
“explaining something”

Expectations

- Guided textbook reading.
- Explaining solved problems: How a given program works, step by step, in detail.
- Answering iClicker questions in lecture.
- Some in lecture hands-on activities, some with groups or partners.
- Problets (new for UA)
- Lab (in-lab plus followup homework)
- Programming Projects
- READ ALL ASSIGNMENTS LINE-BY-LINE. NEVER COPY CODE YOU ARE ASKED TO WRITE YOURSELF!!!

Expectations

- Lab (in-lab plus follow up homework)
- Programming Projects
  NEVER COPY CODE YOU ARE ASKED TO WRITE YOURSELF!!!

(Expectations) What do you do?

NEVER COPY CODE YOU ARE ASKED TO WRITE YOURSELF!!! What if you saw some???

The goal is learning.

Put away any notes containing code.

(RE)WRITE solution code YOURSELF

by

REPRODUCING THE IDEAS your helpers told or showed you!

Your responsibilities

- Get your own or begin using the UA Librarys' DrJava programming software BY MONDAY!!
  - See it now. Practice in THIS WEEK's LABS.
  - TAs and I will ***try*** to help you with your own computers, but we can't guarantee anything.
  - NO DELAYS ACCEPTED!!! Use Libraries for now!!
  - Start with your own computer TODAY if you MUST do your homework off-campus!!!
- Access the public web site whenever you study.

`http://www.cs.albany.edu/~sdc/CSI201`

My responsibilities

- Present easy-to-follow instructions to guide you to practice the thinking, lookup/recollection skills, worked example analysis, and problem-solving work of computer science (beginning with programming).
- Make the instructions easy-to-follow FOR ANYONE who actually followed the PREVIOUSLY GIVEN instructions.
- Give improved explanations and instructions when anyone WHO FOLLOWED THE PREVIOUS INSTRUCTIONS tells me new material is hard to follow.
- Design and test-drive a sequence of exercises, labs and projects to BUILD UP your ability to meet the course objectives (like your muscles...only when you DO THEM)

CSI201 vs others in UA CCI

- Computer Science MAJORs worldwide lead to professions and graduate studies that start and use COMPUTER PROGRAMMING deeply.
- We serve the needs of you or your classmates who REALLY WANT to actually PROGRAM COMPUTERS, to become software development professionals or computer scientists, or want to work closely with them.
- 201 plus CSI310 (Data Structures) are the only particular required courses for the CS minor.
- Worldwide, any major or minor with the CS name MEANS a person studied programming at least through data structures.

Alternatives to CSI201

- Computer Science minors must take 201 and 310 eventually, but can start with easier courses. (201+310+any 4 CSI, with at least one 300 level or above. So you can take 3 100 level CS courses before 201, 310 and your upper div. elective! OR take 201+310+4 300-400 level courses.)

Program a sequence of print outs.

Live Coded Demo! (fun to watch a prof. struggle and make mistakes)

All live coded material, sometimes with added comments, WILL BE PUBLISHED on the public web site.

Program a sequence of Turtle drawing strokes.

- Study an example of doing it in detail.
- Do a small example in lab.
- Figure out, build step-by-step, try, debug and finish a challenging sequence yourself!

  Due as Project 1, 1 week from Wednesday!

LOGO Video Links

Turtle Drawing: Short Example

Turtle Drawing: How to figure out the example in the homework assignment.

- Number each method call.
- Mark and identify by number

WHICH METHOD CALLS

MADE THE TURTLE

DRAW **WHICH** LINES OF THE B

- I will scan and post my pencil scratchings!
- YOU do the same for the E--BY YOURSELF!!!

Plan...

- I'll demonstrate

  figuring out

  HOW the code on the Proj 1 handout

  MAKES

  a Turtle DRAW a B

- YOU do that for the E

  so you will understand

  how to do the homework!

Project Assignments Soon

- Proj1 (NOW) & this week's Lab: you program ONLY a sequence of method calls.
  - LAST TIME it will be that simple!
- Proj2 (due Sept 19??) MUST
  - create YOUR OWN methods (not just use given ones)
  - your own methods MUST have parameter
  - you program calling your own methods.
- Labs 2, 3 & Proj2: Program calculations with variables.

Surprising Examples

- http://www.youtube.com/watch?v=xMzojQFyMo0
- After about 1 hr of play, I discovered that each turn was a little under or over 120 degrees, and three successive turns were a little different, makes a cool figure. I think they increased length a little after each line.
- Circle of Golf Clubs Demo
  - You'll write programs LIKE THIS and beyond for your midterm exam!!! (October 10).

What is the best way to learn from a worked example (like my B E drawing code)

A) Figure out, MENTALLY, what it does, by concentrating ONE LINE OR COMMAND AT A TIME, together with notes you wrote about the results you already figured out from the previous lines or commands.

B)Carefully type it in, compile, fix syntax errors due to mistyping, and then run it, to see it do what it does.

Making programs work yourself

- Get (download/install) the Java Development Kit (if your computer doesn't have it already)
  - Lab, Library computers and Macs have it already
- Get (download) the DrJava (Beta version!) Software (if your computer doesn't have it already)
  - Lab and Library computers have it.
- Get the “Book Classes”
  - In Lab you will get them onto your UA “C:” drive.
  - On your computer, you download them.
- Set DrJava's CLASSPATH to your Book Classes folder. (Instructions in Lab and the text.)

Testing your setup

- DrJava for generic Java
  - Launch Drjava
  - Type and save a Hello program in Definitions pane.
  - Click Compile and then Run
  - All is well if you see the Hello printed.
- DrJava for the Book Classes
  - Make sure the CLASS PATH is set.
  - Add to your Hello program right under the println method call, Compile, Run

```
World w = new World( );
```

  - All is well if there are no error messages and you see a new window appear.

Points to remember

1) A program is a written SEQUENCE of actions the computer performs exactly in the ORDER they are written.

2)The actions (so far) result from CALLING

(a kind of commanding, invoking, making-happen)

a named potential behavior (a METHOD) FOR or ON a particular OBJECT, where that object can DO that behavior.

```
tu.turn(180);
```

Points to remember

1) program... written SEQUENCE of actions ...

2)The actions result from CALLING

(a kind of commanding, invoking, making-happen)

a named potential behavior (a METHOD) FOR or ON a particular OBJECT that can DO it.

```
tu.turn(180);
```

The Turtle object is referred to by **`tu`**

The method name is **`turn`**

A Turtle can turn.

It can turn a given **parameter** number of degrees.

The Albany Way
(NOT Georgia Tech!)

- All Java code you write is to be PART OF A
  - Complete Java Application Program..
  - that you can show to someone else...
  - can have that person RUN your program...
  - and everyone can **SEE AND DESCRIBE**

WHAT IT MAKES THE COMPUTER DO

- **(Do NOT, by hand, command the computer by typing Java into the Interactions Pane...your work is lost and that's not programming!!)**

What's the Albany Way?

To see, by experiment, what a line of Java does, we:

(A) Edit the main method of our Hello program with the definitions pane, save, compile and run.

(B) Type it in DrJava's interaction pane and press ENTER.

Next Class

- A more fundamental view of a programmed computer...
- It IS a human or robotic person or agent...
- that follows a recipe of instructions...
- one step at a time...
- to copy, calculate, erase and rewrite...
- data (ultimately numbers in binary)...
- WRITTEN ON PAPER LEDGER SHEETS or...
- stored in (today's) mostly silicon memory chips.

Expectations

- Lab (in-lab plus follow up homework)
- Programming Projects

NEVER COPY CODE YOU ARE ASKED TO WRITE YOURSELF!!!

What if
somebody shows you part of a solution, or
you work on it or see it in a group?
...a group led by a tutor, TA; or just friends?
THAT'S FINE! The goal is learning.

REPRODUCE THE IDEAS your helpers told or showed you!

What if you can't?

Figure out with friends what questions to ask!

Talk to somebody who really understands the subject, who asks you specific content questions to see where your understanding reaches and ends, and leads you to understand more.

this takes time: so start all projects early!

Expectations

Beginning the 2nd or 3rd lab, we will teach and DEMAND:

Your 201 computer files be ALONE, under a DEDICATED folder/directory for 201

EACH Lab and Project be under ITS OWN, SEPARATE sub-folder or directory.

SEPARATE VERSION sub-sub-folders are to be used to preserve and then SUBMIT FOR GRADING

the HISTORY

of your work on the project.

(Expectations) Version History

1) With a version history, your old work is saved, you won't lose it. You can go back to a good start after you (inevitably!) mess it up.
2)Serious software developers today ALL use version history or control of some sort (except perhaps for tiny, not-worth-saving kinds of work), but most students don't...so...you can impress interviewers!
3)Version histories are one basis for world-wide software development teamwork.

Version History: Links for the Curious or Skeptical

- Wikipedia article
- GIT
- SVN
- Mercurial
- It's built into software like MS Office (could be embarrassing or get you sued!)

Expectations

How will your learning of programming skills will be (eventually) ASSESSED—that means for a GRADE?

EXAMS! abeit imperfect..

Midterm and Final exam questions will sample how well you can write code LIKE but not the SAME AS that you have been expected to puzzle out in Project (and lab) work. (Or analyze code we give..)
IF YOU ACTUALLY EXERCISED YOUR BRAIN ON OUR HOMEWORK: The questions will be easy.
IF YOU MAINLY COPIED, EVEN IF YOU READ and reread your copy: Your answers will show us your are “clueless”

Plan of first 2 projects and 3 labs

- Project 1
- Lab 1
- Project 2
- Lab 2
- Lab 3

(if time) What our program does

- Make a new **`World`** object
- Use variable `w` to refer to it.
- Make a new **`Turtle`** object in the **`World`** referred to by **`w`**. Use variable **`tu`** to refer to it.
- Run, in the programmed sequence, dozens of method calls ON that same **`Turtle`** object, which make that **`Turtle`** draw stuff by moving and turning.
  - Those method calls have the **syntax**:

`tu.turn(` some number **`);`** or
`tu.forward(` some number **`);`**

What is "**`Turtle`**" ? Hint: Think of how we used the word **`Turtle`**.

A) **`Turtle`** is the **kind or species of objects** the program made one copy of. By referring to that **`Turtle`** by **`tu,`** the program commanded it to draw stuff by moving and turning.

B) **`Turtle`** is **one particular object** capable of drawing on command.

- Make **a** new **`World`** object
- Use variable `w` to refer to it.

- Make a new **`Turtle`** object in the **`World`** referred to by **`w`**. Use variable **`tu`** to refer to it.
- Run, in the programmed sequence, dozens of method calls ON **that** same **`Turtle`** object, which make **that** **`Turtle`** draw stuff by moving and turning.
  - Those method calls have the **syntax**:

`tu.turn(` some number **`);`** or
`tu.forward(` some number **`);`**

What is "**`Turtle`**" ? Hint: Think of how we used the word **`Turtle`**.

A) **`Turtle`** is the **kind or species of objects** the program made one copy of. By referring to that **`Turtle`** by **`tu,`** the program commanded it to draw stuff by moving and turning.

B) **`Turtle`** is **one particular object** capable of drawing on command.

Object Oriented Programming 1

- Potential actions a computer might do...
- are METHODS that belong to an OBJECT...
- think: an OBJECT, like a real live turtle animal,
- has potential BEHAVIORS...like walk forward and turn...
- we programmers can program it (the poor turtle) to DO for us.
- We program the computer to draw a line...
- by programming one Turtle to move forward.
- Our program CALLS that Turtle's forward METHOD.

What is "`Turtle`" ? Hint: Think of how we used the word `Turtle`.

A) `Turtle` is the **kind or species** of **objects** the program made one copy of. **class**

By referring to that `Turtle` by `tu,` the program commanded it to draw stuff by moving and turning. **A program could have made MANY DIFFERENT Turtles, of the same KIND.**

B) `Turtle` is **one particular object** capable of drawing on command. **NO!**