introduction to the course

• my pitch:
  – This course is ultimately about control!!!!
  – Learn how to control computers and a surprisingly large number of devices and other seemingly non-technical components that you encounter in your everyday life!
  – Today, technology is ubiquitous—learn how to control it before it takes control of you!
• but seriously:
  – introduction to computer programming using the C++ language
  – uses game programming as a context (i.e., the basis for examples and some of the assignments)
  – not open to students who are enrolled in or have completed CIS 1.10 or 1.20 or 2.80 or 15 or 16
  – cannot be taken simultaneously with or before CORC 1312 (CC 3.12)

topics

• the following topics will be covered in 10 curricular units:
  – displaying simple information and remembering it (output and data, units I-II)
  – dealing with complex information (arrays and strings, unit III)
  – reading simple information and making decisions about it (input and control structures, units IV-VI)
  – doing interesting things with all kinds of information (searching and sorting, unit VII)
  – behaving efficiently (functions, unit VIII)
  – organizing programs (simple classes, units IX-X)
about me

• undergrad: Barnard, CS major/English minor, BA 1985
• 10 years of industry experience working as a scientific and business programmer
  – MIT Lincoln Laboratory (http://www.ll.mit.edu/)
• grad school: Brandeis University, MA 1997, PhD 2000
• previous teaching:
  – Monash University, Melbourne, Australia (Summer 2000)
  – University of Melbourne, Melbourne, Australia (Summer 2000)
  – Boston College, Massachusetts (Fall 2000–Spring 2001)
  – Columbia University (Fall 2001–Spring 2005)
  – Brooklyn College, Fall 2005– ...
• research interests:
  – multi-agent simulation
  – human/multi-robot teams
  – educational robotics

brief overview of game programming

• using a computer to tell a story or play a game
• elements of game programming

game versus narrative

• narrative
  – storyboard describes a story line or a script
  – movement from one screen in the storyboard to another is strictly linear (i.e., you only
go forward from one to the next; you can’t go backwards)
   □□□□□□□□□□
  – the content of a narrative is scripted
• game
  – storyboard describes game play
  – movement from one screen in the storyboard to another can be recurrent (i.e., you can
move backwards and forwards and skip around)
   □□□□□□□□□□
  – the content of a game is programmed (elements may also be scripted, but the game as
a whole is programmed, and portions of the program invoke any scripted content)

Many of today’s more sophisticated video games have story elements in them, but don’t
confuse “narrative” with “context”.

• Who has control?
  – user-controlled versus author-controlled
  – In a narrative, the control belongs solely to the author. The user (viewer) can’t do
anything except “play” and “stop”.
  – In a game, the control is shared between the author and the user. The author designs
the screens and types of actions, but the user, through playing the game, controls
which screens follow which, based on the actions the user performs.
• How will it end?
  – In a narrative, there is only one ending. No matter how many times you watch a movie,
it will always end the same way. Dorothy will always make it home in the “Wizard of
Oz”, and Cinderella and Julia Roberts will always marry their princes.
  – In a game, the author designs multiple possible endings, and the actions of the user
determine which ending happens.
elements of game design

• characters
  – called sprites, also called agents
  – some are avatars—these represent the user explicitly; i.e., these are user-controlled
  – sprites or agents can represent the user or can be their own autonomous
    (self-controlled) entities; i.e., these are game-controlled
  – when designing a game, you need to decide what kind of sprites will be in your game
    and how they will be controlled
• “levels”
  – some games have different modes of play, called levels, that are typically characterized
    by their difficulty
  – the first level that a new user encounters is typically easier than later levels
  – user’s progress from easier to harder levels as they gain more experience with the game
  – each level can be characterized by different content, visual and audio aspects, user
    activity, etc.

scoring

• most games typically have a numeric mechanism by which users are awarded points for
  accomplishing certain tasks
• some games take points away if the user does bad things

intrinsic versus extrinsic motivation

• Intrinsic motivation means that the scoring in the game is based on performance
  directly related to the user’s activity in the game.
• Extrinsic motivation means that the scoring in the game is based on performance
  indirectly related to the user’s activity in the game.
• For example, “Carmen San Diego” is a geography game where users track a criminal
  around the world. Users are given geographic hints that indicate where the criminal is
  hiding. The user’s knowledge of geography directly influences how well s/he tracks the
  criminal, and how well s/he does in the game. This is an example of intrinsic
  motivation. In “Baseball Math”, the user is asked to solve mathematical equations.
  Every correct answer generates a hit in a simulated baseball game. The user’s
  knowledge of mathematics indirectly influences how well s/he plays baseball, but a
  user’s knowledge of baseball has no impact on his/her performance. This is an example
  of extrinsic motivation.

types of games

• puzzle-based
  (e.g., Scrabble, TextTwist, Hangman, TicTacToe, etc)
• plot-based
  (e.g., Rogue, Zelda, etc)
• simulation-based
  (e.g., SimCity, SimAnt, etc)
• performance-based
  (e.g., sports games, first-person shooter games, etc)
• some games are educational
• some games are purely for entertainment
• others combine the two: edutainment

players

• human player (“self”)
  – what role will the user play in the game?
  – will the user be an observer?
  – will the user be a controller?
  – will the user cooperate with other players, either human or agent (“bot”)?
  – will the user compete with other players, either human or agent (“bot”)?
  – how much information will the user have? i.e., how much can the user “see”?
  – what can the user control? other agents? the environment?
• single-player versus multi-player
computer science versus art

• different perspectives
• computer science focuses on technical aspects of the game:
  – behavior, intelligence of the characters
  – activity in the game
  – mechanics of how the pieces interact and function
  – dynamic aspects (how does the game respond to changes?)
  – hardware platform (where is the game played? how is the game delivered?)
• art focuses on visual and audio aspects of the game:
  – aesthetics
  – music
  – sound fx (effects)

other aspects of game design and game programming

• learning
  – does the game adapt or change as the user learns to play it better?
• action
  – “real-time” (dynamic) versus static
• game play
  – synchronous (players take turns playing) versus asynchronous (everyone plays at once)
• environment
  – is there a physical counterpart? does physics matter?
• data collection
  – does the game collect information about users while/after they play? e.g., high scores table
  – game logs keep track of all the actions the user takes and how the game responds
  – user profiling: categorizing the user’s actions based on his/her performance

getting started

• we will use game programming as a context for the course
• many of the examples we discuss will be based in games
• you will write simple games for many of your assignments
• mostly we will explore simple, familiar games such as hangman or tic-tac-toe
• toward the end of the term, we may get more adventurous

how to learn a programming language

• YOU are responsible for your own learning!!!
• I will point you in the right direction...
• but YOU must PRACTICE, PRACTICE, PRACTICE...
• and PRACTICE some more!!!
• It’s like learning to play the piano—you have to put your fingers on the keyboard!
• If you don’t understand, then ASK for help!
• You need to be ready to THINK and learn how to figure things out.
• Think ‘outside the box’!
Scratch is a narrative and game programming environment created for young students. There are many different ways to write computer programs. They all involve writing instructions that tell the computer what to do. In Scratch, these instructions are given sequentially.

Each instruction is represented by a sort of puzzle-piece-shaped block. The blocks are designed to fit together in pre-defined ways, to make sure that you (the programmer) follow the rules of programming.

Sometimes you want the computer to follow a set of instructions multiple times. "Looping" or "Repetition" means doing something more than once. Sometimes you want the computer to make a choice about which instructions to follow. Branching or Selection means making a choice about what to do next.

Key is to know when to STOP looping (i.e., repeat a fixed number of times, repeat forever, repeat if a condition is true, or repeat until a condition becomes false).
scratch: controlling sprite motion

• In Scratch, you can control the “motion” of the characters (sprites).
  – Notice that the movement of the sprite is measured in **steps**.
  – Notice that the heading of the sprite is measured in **degrees**.
  – Notice that the position of the sprite is represented in (x, y) **coordinates**.

scratch: editing sprites

• You can change the “look” of a sprite. In Scratch, Sprites “wear costumes”.
• You can also create new sprites.

scratch: sounds

• You can control the sounds that your Scratch program makes.
• Sounds can be associated with sprites and/or with certain things happening in the game or narrative.

scratch: sprite communication

• You can have sprites send each other messages using the “broadcast” feature.
**scratch: sample games**

- Scratch is available on-line for FREE! [http://scratch.mit.edu](http://scratch.mit.edu)
- Try some sample games from the on-line gallery:

  - Nuclear Fallers
  - Traffic Jam
  - Space Beans
  - Super Mario
  - Zelda
  - Donkey Kong

---

**to do**

- **in class:**
  - fill out pre-semester survey and give it to me before you leave today
  - work on today’s lab
- **at home:**
  - check out the class web page: [http://www.sci.brooklyn.cuny.edu/~sklar/cisc1110](http://www.sci.brooklyn.cuny.edu/~sklar/cisc1110)
  - get a USB flash drive if you don’t already have one
  - get the textbook: