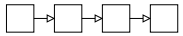
cis3.5 spring2009 lecture III.1

topics:

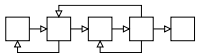
- today we will discuss game and narrative programming

story

- *storyboard*
  - an series of drawings showing how visual aspects (e.g., animation) and/or activity changes while the game or narrative is running
- 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".

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.

outcomes

- 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.

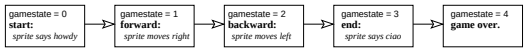
- scoring
  - most games typically have a numeric mechanism by which users are awarded *points* for accomplishing certain tasks
  - some games take points away if the user does bad things
- *intrinsic* versus *extrinsic* motivation
  - *Intrinsic* motivation means that the scoring in the game is based 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* motiviation.

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*

game state

- any game consists of a sequence of *states*
  each state is characterized by a combination of visual, audio and/or animation
- the progression of game state is typically drawn using a diagram like the one below:



- note that each state is given a number, starting with 0 and ending with 4
- in the first state, $gamestate = 0$: the game starts up and the screen shows a sprite in its starting location saying "howdy"; then the game state changes to 1
- in the second state, $gamestate = 1$: the sprite moves to the right, until it reaches the far right edge of its display window; then the game state changes to 2
- in the third state, $gamestate = 2$: the sprite moves to the left, until it reaches the far left edge of its display window; then the game state changes to 3
- in the fourth state, $gamestate = 3$: the sprite stops moving and says "ciao;" then the game state chnages to 4
- in the fifth state, $gamestate = 4$: game over!

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 behavior, activity in the game
- art focuses on visual and audio aspects of the game

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