cisc3660
game programming
fall 2012
lecture # III.3

topics:

- game loop
- game state machines

references:

- `http://www.evl.uic.edu/spiff/class/cs426/`,
  by
  Prof Jason Leigh, University of Illinois at Chicago (`http://www.evl.uic.edu/spiff/`)
  and
  Prof Robert Kooima, Louisiana State University (`http://csc.lsu.edu/~kooima/`)

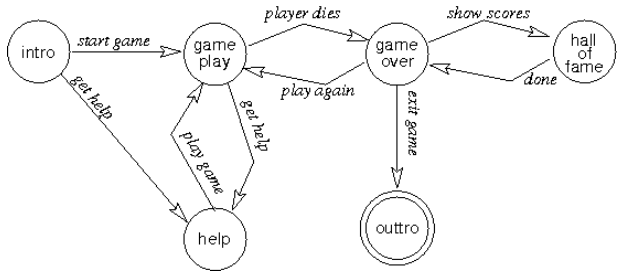
game loop

- game play is controlled by an iterative loop that cycles through multiple tasks:
  - read user input
  - calculate user parameters
  - calculate non-player character behavior/response
  - draw graphics
  - handle sound effects
- in an environment like Blender, these tasks are not necessarily handled sequentially (because multiple tasks can occur in parallel)
- but it is still a good idea to design each of these tasks and think about how they impact each other

finite state machine

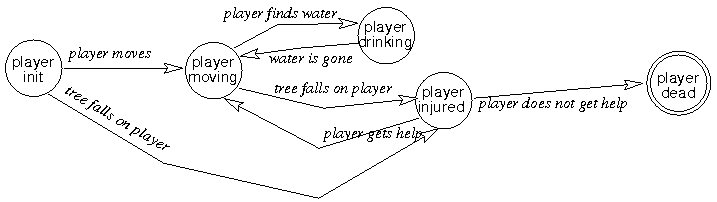
- a *finite state machine* or *FSM* is a graph that consists of nodes and directed links
- each node represents a *state* in the game
- a state is typically characterized by parameter settings for game objects
- which can be represented visually in the game environment
- for example, the scene for the game could be either "day" or "night"
- the visual representation of the "day" state could display a sun, whereas the visual representation of the "night" state could display a moon
- the directed links show *transitions* between states
- the transitions occur either as a result of *actions* carried out by the (human) player or reactions between autonomous game objects (such as collisions)
- the actions can cause parameter values to change, indicating different states

- an example finite state machine outlining the overall structure of a game is shown below:



- each of the states in this graph should correspond to one (or more) screens in your game design
- note that, by convention, the final state in an FSM is drawn with a double outline ("outtro" state in the example above)

- an example finite state machine outlining the game play is shown below:



- this graph shows multiple states for the player's avatar:
  - "init" = initial state
  - "moving" = when the player is moving around normally in the virtual world of the game
  - "injured" = when the player has been injured and is unable to move around
  - "drinking" = when the player has found water and drinks to increase her health
  - "dead" = game over!

game objects and states

- it is helpful to define game objects and states for each game object
- the state of a game object can be represented by the value(s) of one or more parameters that describe the game object
- then determine what actions can change the parameter values, such as:
  - actions taken by the (human) player
  - actions taken by an autonomous game object (e.g., artificial intelligence)
  - actions resulting from physics in the game environment
- not all changes in parameter values necessitate a change in an object's state
- a **game state** can then be defined by listing the states of all the game objects—i.e., a set of parameter values for all game objects corresponds to a game state