

Basic Behaviors in Game Maker

Prof. Jim Whitehead
CMPS 80K, Winter 2006
February 22, 2006

AI isn't really AI

- Much of what goes by the name of Artificial Intelligence in computer games does not involve the use of sophisticated AI
 - We'll use the term behavior instead
- Especially in simple games, computer behaviors are mostly quite simple
- Still, a little goes a long way

Behavior: Bullets Fire Towards Player

- A simple behavior is to have computer controlled characters fire their bullets towards the player
- Determine when the character fires
 - Periodic firing can be achieved using alarms
 - Initially set alarm in create event
 - In alarm event, create bullet, then reset alarm
- Ensure the bullet is fired towards player
 - In create event for bullet, use “move towards point” action
 - Set x to *{player object}.x* (e.g., “pacman.x”)
 - Set y to *{player object}.y* (e.g., “pacman.y”)
- Show demo “bullet-fires-towards-player”

Behavior: Homing Bullets

- A more complex behavior is to have bullets that home in on the player's position
- Use periodic firing from previous example
- In create event for bullet
 - Set speed to 10
 - Set direction to “random(360)”
 - Picks a random direction to make homing more clear
- In step event for bullet
 - Set action to “move towards point”
 - As before:
 - Set x to *{player object}.x* (e.g., “pacman.x”)
 - Set y to *{player object}.y* (e.g., “pacman.y”)
- Show demo “homing-bullets-too-good”
- Problem: Bullets are too good, the player can never avoid them

Behavior: Periodic Adjustment to Bullet Trajectory

- To make bullets less deadly, only adjust them periodically
- Key idea: use alarm as a timer, whenever alarm goes off, adjust trajectory
 - In create event for bullet
 - Set alarm to go off some time in the future
 - I used a random time from 0.5-2.5 seconds
 - Set action to “move towards point”
 - Towards player object’s position, as in previous two examples
 - In alarm event
 - Do the same thing as for the create event
 - Set alarm to go off some time in the future (reset the alarm)
 - Ensures we continually are adjusting the trajectory
 - Otherwise would only adjust once
 - Set action to “move towards point”
 - Adjusts trajectory to move towards player
- Show demo “homing-bullets-alarm-adjust”
- Better: bullets not so deadly, but bullets adjust quite dramatically

Behavior: Monsters Move Towards Player When Close

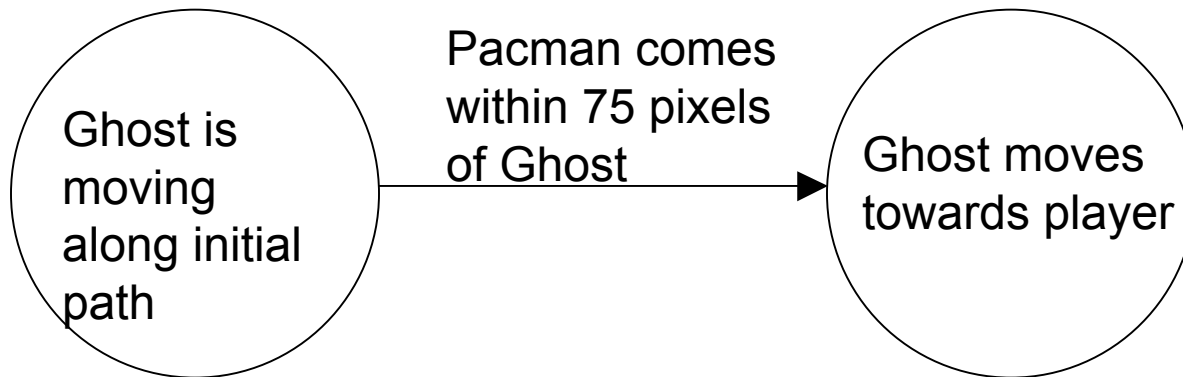
- In many games, you want the monsters to move on a path, and then move towards the player once the player gets close
- Check for proximity:
 - In the step event of the computer character, check if the distance to the player is less than some amount
 - Use “if an expression is true” action
 - Expression is “distance_to_object(*{player object}*) < *a_distance*”
 - Example: “distance_to_object(pacman) < 75”
 - If the expression is true, start moving towards player character

Behavior: Move Along a Path, then Move Towards Player When Close

- In games, often the computer characters mind their own business until the player comes along, then they react to the presence of the player
- In Game Maker:
 - Computer character object follows a path
 - Until the player gets close
 - Then the character object starts moving towards the player
 - Or starts another path

Need a State Machine

- This situation is modeled using a simple state machine:



- Each Ghost has its own state machine
 - 6 ghosts → 6 state machines

Represent State Machine

- The current state of the machine is represented using the variable “moving”
 - Moving = 0 :: moving on initial path
 - Moving = 1 :: moving towards player
- The state transition is checked for in a step event
 - Use “if an expression is true” action
 - Expression is “distance_to_object(*{player object}*) < *a_distance*”

Logic of State Transition

If (the player gets close to the character) Then

 If (moving is equal to 0) Then

 moving is set to 1 // *advances the state*

 end the character's path

 move towards the player

 End if

End if

Example

- Show example “monsters-run-toward-player”
- Can make more complex behaviors by adding additional states to the state machine