Creating Games with Game Maker: Inheritance, Variables, Conditionals

Prof. Jim Whitehead
CMPS 80K, Winter 2006
Feb. 8, 2006
Similar Behavior

• When creating a game, you often have a situation where you have the same object, but with slightly different behavior
  – Example: two monsters that look the same, have same collision behavior, but have different paths
  – Example: several monster types may have the same collision behavior, but have different kinds of movement, and look different

• Ideally want to specify the behavior that is the same just once
  – Can be really boring to repeat the same collision behavior for 20-30 slightly different monster types
    • Plus, if you ever need to make a change to this behavior, you need to make the change in 20-30 places. Yikes!
Inheriting Behavior

• Game Maker provides the ability to:
  – State behavior common to many objects in one place
  – Indicate which objects use this common behavior

• General idea:
  – Objects can optionally have a “parent”
  – If an object has a parent, the parent’s behavior is used by the child
    • So, if my parent says “collisions with a bullet destroy me (the parent)” then it is also the case that “collisions with a bullet destroy me (the child).”
    • The technical term for this is “inheritance” – you inherit the behavior from your parent.
Inheritance Example

Mother-of-monsters

Event: collide with bullet has
Action: destroy instance

Big-hairy-monster

Event: collide with bullet has
Action: destroy instance

Tiny-smelly-monster

Event: collide with bullet has
Action: destroy instance

This behavior is inherited from “Mother-of-monsters” – it is not specified in Big-hairy-monster.
Inheritance Example

Mother-of-monsters
Event: collide with bullet has
Action: destroy instance
**Action: play sound “argh!”**

A single change in the parent is inherited by all children

Big-hairy-monster
Event: collide with bullet has
Action: destroy instance
**Action: play sound “argh!”**

Tiny-smelly-monster
Event: collide with bullet has
Action: destroy instance
**Action: play sound “argh!”**
Overriding Behavior

• It is usually the case that children want some, but not all of their parent’s behavior
  – Example: want parent’s collision behavior, but don’t want parent’s creation behavior
    • For example, want to set a different path than the parent in the create event handler

• A child’s specification of behavior “overrides” that of its parent
  – Example:
    • parent says “on create event, do action set path to ‘big-path’”
    • child says “on create event, do action set path to ‘small-path’”
  – In this case, the child’s specification is used
    • That is, the path is set to ‘small-path’ for all child objects
    • The child’s specification ‘overrides’ the parent’s specification
Example of Overriding Behavior

<table>
<thead>
<tr>
<th></th>
<th>Mother-of-monsters</th>
</tr>
</thead>
<tbody>
<tr>
<td>Event: collide with bullet has</td>
<td>Action: destroy instance</td>
</tr>
<tr>
<td>Event: creation</td>
<td>Action: set path to ‘big-path’</td>
</tr>
</tbody>
</table>

Parent-child relationship

<table>
<thead>
<tr>
<th>Big-hairy-monster</th>
<th>Tiny-smelly-monster</th>
</tr>
</thead>
<tbody>
<tr>
<td>Event: collide with bullet has</td>
<td>Action: destroy instance</td>
</tr>
<tr>
<td>Event: creation</td>
<td>Action: set path to ‘big-path’</td>
</tr>
<tr>
<td>Event: creation</td>
<td>Action: set path to ‘small-path’</td>
</tr>
</tbody>
</table>

All behaviors inherited from parent

Collide behavior inherited from parent, creation behavior is set locally, and overrides parent
Setting Parent in Game Maker

- In the object dialog box, in the leftmost column, there is a selection “Parent”
  - A pick list lets you identify the parent object
- Typically you will create a non-visible object to be the parent
  - This object will never be used directly in the game – only its children will appear
Game Maker Demo

• Switch to Game Maker, and show “Space Eggs 2006” demo
• This demo shows inheritance for the behavior of the space eggs, and the monsters
  – Two non-visible objects are created to hold behavior
    • “mother-of-all-spaceeggs”
    • “mother-of-all-monsters”
Conditionals

• In games, very typically you need to have the behavior of the game depend on the current state of the game
  – Examples:
    • Continue to the next level when there are no monsters left on this level
    • Only allow two bullets on screen at a time
    • Have a powerup appear when the player has bopped all monsters in a given wave
Conditionals

• The typical form that these take is:
  – If *(the game is in a certain state)* Then
    • Perform an action
  – Else
    • Perform some other action

• This is known as a *conditional*
  – Perform the action only *on the condition* that
    the game is in a certain state, otherwise
    perform some other action
Conditionals in Game Maker

• Game Maker has just one idiom
  – Actions are performed in reaction to Events
  – Conditionals are placed into Actions
  – This creates a slight mismatch, as typically you have one action per entry
  – Conditionals spread across multiple action entries
Variable

• A variable holds a value
  – A variable has a name and a value
  – Example: “Num_monsters” has the value of 6
• Can perform arithmetic on variables
  – Addition, subtraction, multiplication, division
• Can be very useful for representing game state in your game
  – In Space Eggs example, the number of monsters still to be destroyed before the end of the level
  – Subtract one from this every time a monster is hit
  – When it is zero, move to the next level
• Set variables in actions, and can check the value of variables using conditionals
Game Maker Demo

- Go to “Space Eggs 2006” demo and show how conditionals are used to limit the number of bullets to 2 at any one time
- Also show how conditionals are used to end one level (room), and move to the next level (room)