Levels
Lecture 15 (7/30/2007)

Levels: Ways to Make Them

1. By-hand -- good for levels that are extremely similar to your existing game
   - For example, levels which basically just change a few existing variables (speed, number of enemies, etc.)
   - Also good as a “learning experience”...
2. FANG Engine’s GameLevel class -- good for levels that are often vastly different
   - For example, if each level is basically its own separate game, with a lot of different Sprites, a different way of handling collisions, etc.

Useless Fact of the Day

- It is estimated that the number of individual insects alive at any given time is $10^{24}$ (10,000,000,000,000,000,000)
- This is over 1 billion insects per living human being
- The New York Times estimates that the world holds 300 pounds of insects for every pound of humans

Making Levels By-Hand

- See the in-class exercise we did called “05_asteroids”
- Must add int constant(s) to store:
  - how many levels are in the game (unless this is infinite)
  - (possibly) how many points / blocks destroyed / aliens zapped before you go to the next level
- Must add int variables to store:
  - what level we're on now
  - (possibly) how many points / blocks destroyed / aliens zapped you’ve achieved so far in this level

source: http://www.si.edu/resource/faq/nmnh/buginfo/
Making Levels By-Hand

• Must add code to increase the level counter and re-initialize sprite locations, speeds, etc. when you go to a new level
• Normally this is in a method called changeLevels() or something similar
• You don’t normally want to re-create the Sprites (with “new”), as you may end up with duplicate copies... instead, you just want to re-initialize their characteristics

• Must test for when we try to go past the last level, and end the game victoriously (in changeLevels())
• Must test for the condition under which you want to change levels, and run your level-changing code when the condition is true (call changeLevels())
• (Normally) should have a message showing the player the current level

Making Levels By-Hand

• (Normally) should change game variables based on what level the player’s on
• Often in the re-initialization of Sprites’ characteristics (for example, setting the initial speed of the ball to be level-dependent)
• Sometimes in other portions of code (for example, code to set the number of points earned when a block is destroyed to be level-dependent might be in “handleCollisions()”)

Making Levels with FANG’s GameLevel

• The FANG engine has a class called GameLevel
• This class is very similar to GameLoop, which is the main class in the game engine
• Virtually anything that is a GameLoop can be turned into a GameLevel
• See the in-class example “05_pongLander” -- level one is the game of Pong, level two is the game of Lunar Lander EXTREME
**GameLevel**

- **GameLevel** has these differences from GameLoop:
  - it *doesn’t* have a **main** method
  - **startGame()** is instead **startLevel()**
  - when the level is over, it should call the method **finishLevel()**

---

**Traversing GameLevels**

- There are three different ways to traverse levels. *In order from easiest to hardest to implement:*
  - **Method 1:** Add all the levels at the beginning, using only the **addLevel**(theLevel) method in **GameLoop**. When one level finishes, the next one added will run.
  - **Method 2:** At the end of each level, call the **setNextLevel**(theLevel) method in order to decide which level is next. Use only the **setNextLevel** method of **GameLoop** or **GameLevel**.
  - **Method 3:** Override the **getNextLevel** method of **GameLoop**. Don’t use **addLevel** or **setNextLevel**.

---

**Example**

- Say you’ve written a GameLevel class called Pong, and a GameLevel class called LunarLander. You could set up a game where LunarLander is the first level and Breakout is the second by putting this code in **startGame()** in your main GameLoop class:

```
addLevel(new LunarLander());
addLevel(new Breakout());
```

- Pong will run first, and when Pong calls **finishLevel()**, LunarLander will immediately begin running.
- **startGame()** still starts the entire game over (at the first level).