Slide 1

Administrivia

- Reminder: Homework 2 code due today.
- Homework 3 due dates on Web (design next Tuesday, code Thursday).
- Reminder: Midterm a week from today. There will be a short review sheet on the Web soon, and we can spend part of Thursday reviewing.

Slide 2

Homework 3

- In this homework you start writing code for your player, to replace the stick figure in the starter game.
- Key parts of this assignment are making the player
  - interact with different kinds of blocks.
  - move in response to keyboard or mouse input from human player.

  (If these don't apply to your game, talk to me about whether there are reasonable substitutes.)

  For design phase, you just need to describe this interaction.

Slide 3

Homework 3, Continued

- `Player` defines some constants you should use.
- You will implement `KeyListener` or one/both of the mouse-listener interfaces. When you do this, the framework will deliver key and/or mouse "events" to you.
- Most logic will go in `update`, `getUpdateTime`, and the listener methods.

Slide 4

Multithreading in Java

- Much interest recently in "multithreaded" programming, because of hardware changes — having more than one CPU/core not just for high-end systems.
- Interestingly enough, Java has included support for multiple threads from the beginning. Interaction among Java threads based on "monitors" (see textbooks on operating systems, parallel programming — idea goes back to 1975 papers by Hoare and Brinch Hansen). Java leaves out some aspects of full-fledged idea, but keeps enough to be useful.

Slide 5

Threads in Java

- `Thread` class provides basic functionality. To start a new thread, make a `Thread` object and call its `start` method. Two choices:
  - Create a `Thread` with an object that implements `Runnable` — `run` method has code to execute.
  - Define a subclass of `Thread` that has a `run` method with code to execute.
- Interthread interaction based on (implicit) locks:
  - Every object (and every class) has a lock.
  - `synchronized` methods must acquire lock — so only one at a time can run.
  - `wait` gives up the lock and sleeps; `notify` and `notifyAll` wake up one/all sleeping thread(s).

Slide 6

Threads in Java, Continued

- Other useful methods:
  - `Thread.sleep` makes current thread sleep for some interval.
  - `t.join` wait for `Thread t` to finish.
  - `t.interrupt` interrupts `Thread t` (which can check whether it has been interrupted with `isInterrupted` — safe/approved way for one thread to stop another.
- Can set thread priorities — sometimes useful, but not a substitute for proper synchronization.
- Lots of new threads-related stuff in Java 1.5 / 5.0 (`java.util.concurrent` package).

Slide 7

Uses for Threads in Java

- Formerly many uses for multithreading in GUIs (e.g., animation), but now most can be accomplished with new features of GUI classes (e.g., timers). Still useful, however, if you want something that might take a while to execute in the background.
- Multithreading also potentially useful for improving performance of computationally intensive code.
- Examples as time permits . . . .

Slide 8

Minute Essay

- None — quiz.