Problem: Flickering Images

- When your application draws into a Graphics object, there can be flicker as the draw, update, draw, update cycle continues. Part of this is the function `update(Graphics g)` which erases the background.
  - Use *double buffering* to solve this problem
    - draw into an offscreen buffer, this can take some time
    - bitblt/blast the offscreen buffer onto the onscreen buffer
    - use this for flicker-free animation, see the Bouncy.java example
  - In JDK 1.2, double-buffering is built-in, in JDK 1.1.X you must do it yourself
    - override update to call paint, override paint to do double buffering
    - get the buffer/image from `createImage()`, peer must be set
Other AWT problems/conventions

- Add all widgets to application/applet at once
  - do this in constructor/init, but helper functions are useful
  - cannot have menus in an applet
- In big program avoid inter-widget communication
  - use controller class (see Pixmap/Anagram examples)
  - relax in smaller examples
- Use one-listener per widget, no switch/if to decide what action caused an event
  - anonymous classes useful here, see Pixmap example for creating an anonymous class from an interface
- Don’t forget about validate()
  - when widgets are invalid, call validate on the widget or the widget’s container
More AWT problems/issues

● Don’t forget about layout managers
  ➤ each container (panel, frame, …) has a layout manager --- example of the strategy pattern
    • strategy pattern encapsulates an algorithm/behavior as a class, allows algorithms to be plugged in
    • class that uses the strategy delegates responses/uses of algorithm to myStrategy
  ➤ different containers have different default layouts, but you can/should put a new layout in every container you use

● Images can be imported via URLs, so can audio
  ➤ see Java Examples in a Nutshell
  ➤ you can use images in MultiBouncy.java example
Towards a design for SCOOTER

- What are the classes in SCOOTER, what are the nouns?
  - What are responsibilities/behavior for these classes
  - Are there possibilities for combining/abstracting classes into an inheritance hierarchy?
  - How should robots communicate with factory/rocket?
- What are issues in implementation
  - Don’t even ask this question until you have a good idea about what the classes are
  - what’s harder: rocket or factory (why)?
- What about visualization?
  - where does this go, both literally and in terms of the development cycle