Software Engineering
What Makes a Successful Product?

- **Class Experiences**
  - **Good**
    - o ...
    - o ...
  - **Bad**
    - o ...
    - o ...

- **Scenario: “Bad” Product looks “Good”**
- **Scenario: “Good” Product looks “Bad”**
- Which is really the good product?
- Which will succeed?
Program Life Cycle

- Define the Product
- Developing the Program Specifications
- Designing the System Structure
- Coding the System (small part!)
- Testing the Code
- Revision
- Documentation
- Delivery and Training
- Maintenance and Upgrade
Adapt to CompSci 4

Understanding Problem / Specification

- Communicating with the Customer
  - Who is this customer here?
  - Possible role of user documentation

- Specification languages
  - Can be very technical and involved
  - Not clear they apply to games
  - In some cases more trouble than they are worth
What’s Important to CompSci 4?

Design Strategies/Models

- Classical Waterfall Model
  - Everything moves forward at steady pace
  - Little customer involvement in actual design
  - Revision is possible, but
    - Very late in the game
    - At great expense
What’s Important to CompSci 4?

Design Strategies/Models

- Rapid Prototyping Model
  - Build Prototype quickly
  - Get customer (person grading?) involved?
  - Then:
    - May scrap prototype and start over
    - If prototype good enough, may build on it
  - Remember: Here time is short!
Useful Pointers

Implementation Strategies
- Top Down Implementation
  - Stubs
    - Can test many parts in the absence of other parts
  - Output First
    - Allows you to “see” what the program is doing
- Test as you go
  1. Make it Run
  2. Make it Right
  3. Make it Fast
- Always have a “running” program
Useful Pointers

- **Debugging (dealing with Defects)**
  - Testing
    - Implies a specification: do we have one?
  - Can only show presence, not absence of bugs
  - Design for Testability
    - Modular
    - Hierarchical

- **Correctness**
  - What does correctness mean for a game?
  - Sell it as a feature!
Program Life Cycle

- **Documentation**
  - Two basic kinds
    - For user
    - For heirs
  - Historically often done after coding
  - Now should be on-line
  - Java Docs
    - Available for Java API
    - Must produce for your own code
Aspects of Software Engineering

- Psychology of Programming has its effects
  - Design by Committee (good or bad?)
  - Communications problems
  - Interaction
  - Creator Independence
  - Learning to work with others!

- Organizational Schemes
  - Chief Programmer Team (Harlan Mills)
    - NY Times Morgue Project
    - Surgical Team Model
    - Assumes that some people are 100 time better than others when programming
Aspects of Software Engineering

- Nitty-Gritty Practical Problems
  - Back-ups
  - Save, Save, Save
    - Alternate name save?
  - (Revision Control System?)
- Programming Tools (CASE)
  - E.g., IDE’s such as Eclipse
  - . . .
Program Life Cycle

- *Net* Productivity on VERY LARGE systems: 15 LINES OF CODE / DAY
- Rather:
  15 LINES OF CODE / DAY

- What’s the message for CompSci 4?
  - You’d better get started !!!