WHEN OBJECTS COLLIDE: ABSTRACTIONS OVER COMMON PHYSICS PROBLEMS FOR CAPSTONE PROJECTS IN CS1

Adrienne Decker & Sara Haydanek
(University at Buffalo)

Chris Egert (Rochester Institute of Technology)

CS1-CS2 Sequence

- CS1 (First semester)
  - Small (often disjoint) programs to illustrate introductory concepts
- CS2 (Second semester)
  - “Medium-sized” piece of software, perhaps teams

CS1-CS2 Sequence

- Students often lack sense of accomplishment at the end of first semester.
- Create a capstone experience to end their first semester with this sense of accomplishment.

Our CS1

- Objects, Encapsulation, Instance Variables, Assignment
- Composition, Association, Methods and Parameters
- Inheritance, Overriding
- Interfaces, Abstract Classes, Polymorphism
- Design Patterns
- Using the graphics package (NGP – developed at Brown)
- Arithmetic and Logical Expressions
- Selection, Iteration
- Collections

The Capstone Project

- First attempt – Tetris (Spring 2000)
- Students are given nothing as a base except some helpful hints about rotating pieces and some advice about possible design patterns that could be useful in the completion of the project.

Tetris Data (SP 2004 only)

- 115 Students in class
- 65 Submitted Code for Grading (56.5%)
  - Didn't submit (Overall course grades)
    - 1 – B+
    - 1 – B
    - 1 – B-
    - 3 – C+
    - 4 – C
    - 3 – D
    - 37 – F
- 62.8 Average Grade (C)

Other Assignments

- Tron Light Cycles/Nibbles
  - Students could build:
    - Tron Light Cycles
    - Nibbles (Snake)
    - A program that dynamically switched between the two while playing
- Last time assigned (Fall 2003)

Other Assignments

- Diamond Mine
- Modified version of this game (no overall board-checking to say when game is over).
- Created and assigned in Fall 2004

Diamond Mine Data (FA 2004)

- 164 Students in class
- 115 Submitted Code for Grading (70%)
  - Didn't submit (Overall course grades)
    - 1 – B-
    - 1 – C+
    - 3 – C
    - 3 – C-
    - 8 – D
    - 33 – F
- 64.3 Average Grade (C)

Overall Issue

- We wanted a way to assign more interesting games for students to build.
- Some thoughts that had been thrown around:
  - Asteroids
  - Centipede
  - Galaga/Galaxian

Problem with our Ideas

- All require some sort of collision detection
- Students would spend a lot of time creating algorithms for collision detection and missing out on the overall game-creation experience

And in walks an eager mind...

- Student looking for an interesting independent study project to work on + need for collision detection = Physics Package is born!

Physics Package

- Collision Object
- Any object that will need to detect collisions with other objects will extend this object and override the collisionReact() method with the appropriate reaction when collisions are detected.

Physics Package

- Motion
- Collision Objects (or their subclasses) can be given a “motion”, a class that implements the IMotion interface that allows us to specify how an object should move. We can also use the motion to indicate the impact of friction and gravity on the object (if wanted).

Physics Package

- Collision Group
- Some objects on the screen need to know when they collide and others do not.
- Objects in the same collision group are checked for collisions within the group. If a collision is detected between objects, their collisionReact() methods are called.
- If the object exists, but is not in the collision group, the collision is never detected or reacted to.

Let the Fun Begin!

- Centipede!
- Students must use the Physics package and their graphical abilities to make a modified version of Centipede.

Centipede Data (SP 2005)

- 102 Students in class
- 55 Submitted Code for Grading (53.9%)
  - Didn't submit (Overall course grades)
    - 1 - B
    - 2 – B-
    - 1 – C+
    - 1 – C
    - 4 – C-
    - 4 – D
    - 34 – F
- 79 Average Grade (B)