Improving Engagement in a Traditional Programming Class through Progressive Challenge and Community Building

Amber Settle
School of Computing

DePaul Faculty Teaching and Learning Conference
April 20, 2012
Introduction

Engaging game design

Good (programming) pedagogy

Traditional (programming) classroom
Engaging game design

- Interactive
- Easy to begin but hard to master
- Progressive challenge
- Encourages risk through targeted reward
- Community development

Sources:
- *Good Video Games and Good Learning*
  - James Paul Gee
- *Rules of Play*
  - Katie Salen and Eric Zimmerman
Traditional (programming) classroom

- **CSC 241/242: Introduction to Computer Science I/II**
  - Undergraduate developers
    - Computer science (currently)
    - Game programming (beginning Fall 2012)
  - Novice programmers
  - Programming syntax
  - Problem-solving skills

- **Combination of lecture/lab**
  - Lecture twice a week
  - Lab once a week
Good (programming) pedagogy

- Easy to begin but hard to master
  - Simpler programming environments
    - Non-majors: Alice, Scratch, App Inventor
    - Simpler syntax: Python
  - Interesting problems

- Interactive
  - Live coding
  - Peer interaction
    - Peer instruction
    - Pair programming
  - Tools for immediate feedback
    - CodeLab

- Progressive challenge
  - Solutions that generalize
  - Concepts that apply across varying problems

- Community development
  - Support
  - Retention
A case study

- Curricular context
- Course/assignment structure
- Advantages
- Emergent issues
- Lessons moving forward
Context

• **CSC 241/242: Introduction to Computer Science I/II**
  ▫ *Required only for computer science majors/minors*
    • Next year: Game programmers too
  ▫ **Programming skills**
    • CSC 241: Basic types, writing functions, decision structures, looping structures, file processing, exception handling, collections
    • CSC 242: Object-oriented programming, GUI development, recursion, searching and sorting, web application development, databases
    • Language: Python
  ▫ **Computer science problems**
    • Searching
    • Text/file processing
    • Web crawling
    • Limits of computation
Course structure

- **Augmented lectures**
  - Introduction of syntax/concepts
  - Exploration through multiple examples
    - Until Spring 2012: Live coding only
    - Spring 2012 and beyond: Hands-on exercises
      - Using the interpreter
      - CodeLab
  - Highly interactive

- **Lab sessions**
  - Focused on problem solving
    - No suggestion of syntax
  - Collaboration highly encouraged
Assignments

• Lab exercises (first)
  ▫ Time-restricted and challenging
  ▫ 50% participation/50% exercise submission
  ▫ 5% of the grade

• Assignments (second)
  ▫ Individual work outside of class
  ▫ (Nearly) unlimited resources
  ▫ Strictly graded on correctness and style
  ▫ 35% of the grade

• Exams (conclusion)
  ▫ Taken in a lab
  ▫ Time restricted and limited (but generous) resources
  ▫ More generously graded on correctness only
  ▫ 60% of the grade
Advantages

• Easy syntax but hard problems
• Process of programming is exposed
  ▫ Partial solutions
  ▫ Flawed solutions
• Failure management
  ▫ Early identification of problems
  ▫ Motivation for improved effort
• Close connections with classmates
  ▫ Female students group together
Emergent issues

- Just in time preparation
  - Pace is unpredictable
  - Strengths/weaknesses vary
  - Coordination is key
- Balance in discussion
  - Attention-seeking students
  - Positive feedback
- Plagiarism
  - Lab time used to start assignments
  - Record number of violations
- Retaining attention
  - Monitors in front of them
  - Engagement with course material
Lesson moving forward

• Flexibility
  ▫ Coverage of topics
  ▫ Adjustment of assignments

• Academic Integrity policy
  ▫ Labs collaborative
  ▫ Assignments strictly individual
  ▫ Choice of grading policy

• Focus on engagement
  ▫ Is this presented in an interactive way?
Questions?

asettle@cdm.depaul.edu
http://facweb.cdm.depaul.edu/asettle/
(312) 362-5324

Thanks to Ljubomir Perkovic