Integrating Drawing Tablet and Video Capturing/Sharing to Facilitate Student Learning

ACM Global Computing Education
CompEd’19 / May 18 / Chengdu, China

Chen-Wei Wang
York University, Toronto, Canada
Challenges of Undergraduate Teaching

1. **complex computational thinking**: limited prior exposure
   - e.g., nested loops on 2D arrays [paper]
   - e.g., OOP: *aliasing*, polymorphism, dynamic binding [talk]

2. **scheduled in-class lectures**: limited comprehension
   - Large class size restricts **pauses** and **interactions**.
   - Instructor’s **verbal remarks** and **written notes** reflect their **insights into the taught subjects**, but . . .
   - it’s difficult to **copy** and **understand** them simultaneously.
How Would You Help this Upset Student?

Student:
I *did attend* classes
but *failed to follow completely*.
Motivating Question

How can we make the in-depth *illustrations* in class *accessible* to students for their *self-paced study* outside the classroom?
Contribution: An Approach for Effective After-Class Learning

A technique for

- **In-class illustrations** of complex ideas on a **drawing tablet**.
  - Pre-class preparation of **starter artifacts** (e.g., code fragments)
  - Frequent and heavyweight **annotations**

- Allowing students to **review** taught contents outside class

Let’s illustrate the technique using a short **review** lecture on OOP.

At the end of the lecture, ask me a question (**as a student**)!
Consider the following **model** of a person:

```java
public class Person {
    /* Attributes */
    double weight; /* kilograms */
    double height; /* meters */
    /* Constructor */
    Person(double weight, double height) {
        this.weight = weight;
        this.height = height;
    }
    /* Accessor/Getter: Body Mass Index */
    double getBMI() {
        double bmi = this.weight / (this.height * this.height);
        return bmi;
    }
    /* Mutator/Setter: Change of Weight */
    void gainWeight(double amount) {
        this.weight = this.weight + amount;
    }
}
```
What are the **console outputs** produced by the following test?

```java
public class PersonTester {
    public static void main(String[] args) {
        Person jim = new Person(72, 1.72);
        Person jonathan = new Person(65, 1.81);
        System.out.print("Jim’s BMI: ");
        System.out.printf("%.2f\n", jim.getBMI());
        System.out.print("Jonathan’s BMI: ");
        System.out.printf("%.2f\n", jonathan.getBMI());
        jim = jonathan;
        jim.gainWeight(3);
        System.out.println("===== After Jim gained 3 kgs =====");
        System.out.print("Jim’s BMI: ");
        System.out.printf("%.2f\n", jim.getBMI());
        System.out.print("Jonathan’s BMI: ");
        System.out.printf("%.2f\n", jonathan.getBMI());
    }
}
```
Example Lecture: Console Output

- Let’s first verify this in Eclipse!

<table>
<thead>
<tr>
<th>Jim’s BMI: 24.34</th>
<th>Jonathan’s BMI: 19.84</th>
</tr>
</thead>
<tbody>
<tr>
<td>===== After Jim gained 3 kgs =====</td>
<td></td>
</tr>
<tr>
<td>Jim’s BMI: 20.76</td>
<td>Jonathan’s BMI: 20.76</td>
</tr>
</tbody>
</table>

- After Jim gained weight:
  
  Q: Why was Jim’s BMI decreased?  
  [ Didn’t Jim gain weight? ]

  Q: Why was Jonathan’s BMI increased?  
  [ Wasn’t it Jim who gained weight? ]

- Let’s illustrate how this happened!
Example Lecture: Q & A

Questions about the OOP lecture?
A Pattern for Teaching Complex Ideas

- I just demonstrated a teaching pattern, choreographing:
  - Slide Show: Specify Problem.
  - Programming IDE: Illustrate Solution.
  - Drawing Tablet: Annotate on starter pages to gradually build towards the solutions or conclusions. e.g., starter page vs. annotated page in the example lecture
  - Drawing Tablet: Answer students’ questions.

- More examples:
  - Paper: teaching computations on 2-dimensional arrays
  - My lectures page: https://www.eecs.yorku.ca/~jackie/teaching/lectures/index.html
Contribution:
An Approach for Effective After-Class Learning

- Slide Show
- Code Demos on Programming IDE
- Illustrations on Drawing Pad
- Recording
- Notes
- Source Code

recorded & uploaded
re-iterated on demand

Projector Screen

information flow

instructor

students

students

in-class

after-class

ask

answer

present
Study Resources for Students (1)

York Lassonde EECS2030 Fall 2017

27 videos • 6,470 views • Last updated on Nov 30, 2017

Jackie Wang
### Study Resources for Students (2)

<table>
<thead>
<tr>
<th>Topics</th>
<th>iPad Notes (PDF)</th>
<th>Slides</th>
</tr>
</thead>
<tbody>
<tr>
<td>• Singly-Linked Lists</td>
<td>• Stacks and Queues</td>
<td>• 06-Stacks-and-Queues</td>
</tr>
<tr>
<td>• Stacks and Queues</td>
<td></td>
<td>• 06-Stacks-and-Queues (4-up)</td>
</tr>
</tbody>
</table>

**Recording**

**Example Source Code**
Teaching Context

Proposed approach adopted in **undergraduate teaching**: 7 iterations of four courses [1st-, 2nd-, 3rd-year]

Taught **1,295 students**

- **Procedural Programming**
  - variables, assignments
  - if-statements, loops
  - arrays, linked lists, trees

- **Object-Oriented Programming**
  - classes, attributes, methods, objects, aliasing
  - inheritance, polymorphism, dynamic binding

- **Software Design**
  - design by contract, program correctness
  - design patterns

- Nonetheless, the proposed approach is sufficiently general for teaching any complex idea.
Reflections

- **Instructor’s Efforts**
  
  **Starter Pages**: What concepts/examples should be illustrated?

- **Drawing Tablet vs. Blackboard/Whiteboard**
  
  - **Time Effectiveness**: Pre-set starter pages save time on copying.
  - **Reusability**: Starter pages may be elaborated and reused.

- **Drawing Tablet vs. Slide Animations**
  
  **Flexibility**: *Dynamic* control of the pace and level of details w.r.t. the *comprehension level*.
  
  e.g., *starter* page vs. *annotated* page in the example lecture

- **Review of Lectures**
  
  **Repetition**: Even effective in-class illustrations take repetitions to achieve *full comprehension*. 

15 of 24
Beyond this talk... 

- Read my paper!
  - Adopting the Approach
  - Evaluation: Students’ Perception
  - Evaluation: Improvement on Students’ Performance
  - Comparison with Related Works

- Similar approach adopted for creating tutorial materials:


Questions?
Teaching Challenge: Big Classes
Adopting the Approach

- **Personal Computer**
- **Online Sharing Platform**
- **Screen Recording**
- **Tablet Projection**
- **Presentation**
- **Programming IDE**

**software**

- **installed**
- **connected to**
- **uploaded to**

**hardware**

- **Wireless Microphone**
- **Wireless Control/Pointing Device**
- **Drawing Tablet**
Evaluation: Student Perception (1)

Students answered anonymously on a 7-point scale:

1. The course helped me grow intellectually.
2. The course learning outcomes were clearly stated and achieved in the course.
3. The instructor conveyed the subject matter in a clear and well-organized manner.
4. The instructor helped me understand the importance and significance of the course content.
5. Overall, the instructor was an effective teacher in this course.
## Evaluation: Student Perception (2)

<table>
<thead>
<tr>
<th>COURSE</th>
<th>CS1 Response</th>
<th>CS2 Response</th>
<th>CS3 Response</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>58.09% (219/377)</td>
<td>58.42% (59/101)</td>
<td>85.73% (70/82)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>COURSE</th>
<th>Q1</th>
<th>Q2</th>
<th>Q3</th>
<th>Q4</th>
<th>Q5</th>
</tr>
</thead>
<tbody>
<tr>
<td>CS1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>agree</td>
<td>82.33</td>
<td>90.6</td>
<td>not available</td>
<td></td>
<td></td>
</tr>
<tr>
<td>neutral</td>
<td>9.02</td>
<td>4.51</td>
<td>not available</td>
<td></td>
<td></td>
</tr>
<tr>
<td>disagree</td>
<td>7.15</td>
<td>4.14</td>
<td>not available</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CS2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>agree</td>
<td>91.53</td>
<td>98.3</td>
<td>100</td>
<td>98.3</td>
<td>96.61</td>
</tr>
<tr>
<td>neutral</td>
<td>6.78</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1.69</td>
</tr>
<tr>
<td>disagree</td>
<td>1.69</td>
<td>1.69</td>
<td>0</td>
<td>1.69</td>
<td>1.69</td>
</tr>
<tr>
<td>CS3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>agree</td>
<td>80</td>
<td>80</td>
<td>94.28</td>
<td>98.3</td>
<td>90</td>
</tr>
<tr>
<td>neutral</td>
<td>1.43</td>
<td>11.43</td>
<td>2.86</td>
<td>0</td>
<td>2.86</td>
</tr>
<tr>
<td>disagree</td>
<td>18.57</td>
<td>8.58</td>
<td>2.86</td>
<td>10.0</td>
<td>7.25</td>
</tr>
</tbody>
</table>
Evaluation: Improvement on Performance (1)

Student Performance Measure in Various *Complex Ideas*:
1. Subcontracting (Contracts in Descendant Classes)
2. The Visitor Design Pattern
3. Genericity
4. Formal Verification (Proving Loop Correctness and Termination)
5. OOP (Inferring Classes/Attributes/Methods from a Tester)
## Evaluation: Improvement on Performance (2)

<table>
<thead>
<tr>
<th>COURSE</th>
<th>CS3 (SU15)</th>
<th>CS3 (F17)</th>
</tr>
</thead>
<tbody>
<tr>
<td>PROPOSED TECHNIQUE ADOPTED?</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>CLASS SIZE</td>
<td>49</td>
<td>80</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>TOPIC</th>
<th>STUDENT AVERAGE SCORES</th>
</tr>
</thead>
<tbody>
<tr>
<td>Subcontracting</td>
<td>51.63%</td>
</tr>
<tr>
<td>Visitor Pattern</td>
<td>51.33%</td>
</tr>
<tr>
<td>Genericity</td>
<td>63.27%</td>
</tr>
<tr>
<td>Formal Verification of Software</td>
<td>63.62%</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>COURSE</th>
<th>CS1 (SP17)</th>
<th>CS1 (W18)</th>
</tr>
</thead>
<tbody>
<tr>
<td>PROPOSED TECHNIQUE ADOPTED?</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>CLASS SIZE</td>
<td>38</td>
<td>190</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>TOPIC</th>
<th>STUDENT AVERAGE SCORES</th>
</tr>
</thead>
<tbody>
<tr>
<td>Object-Oriented Programming</td>
<td>42.97%</td>
</tr>
<tr>
<td></td>
<td>56.4%</td>
</tr>
</tbody>
</table>
Teaching Context

Reflections

Beyond this talk…

Teaching Challenge: Big Classes

Adopting the Approach

Evaluation: Student Perception (1)

Evaluation: Student Perception (2)

Evaluation: Improvement on Performance (1)

Evaluation: Improvement on Performance (2)