

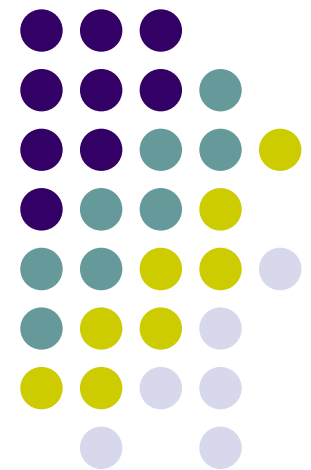
Fast Forward, Rewind, Replay: A Low-Cost Method for Digital Course Capture



Gabriel Hugh Elkaim, *Assistant Professor*
Computer Engineering, UC Santa Cruz

UC2 1st Century.

Teaching, Learning and Technology: Past, present and future
UC Davis, 21 June 2008





Course Recording Pedagogy (1.2)

- Breaks the requirement for students to either:
 - pay attention
 - take notes
- Allows the students the luxury of revisiting lectures to reinforce:
 - During homeworks
 - Reviewing for exams
 - Refreshing knowledge in preparation for other classes
- Breaks the physical constraint for students
 - When classes are scheduled on top of each other
 - Cannot come to class due to illness

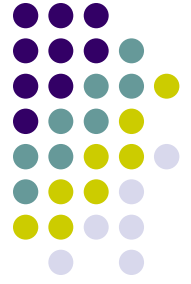


UC21st. 21.Jun.2008

UC SANTA CRUZ, AUTONOMOUS SYSTEMS LAB



Course Recording Pedagogy (2.2)



- For instructors:
 - Allows for pre-recording of lectures
 - Supplementary material
 - Conference travel
 - Illness
 - Domestic Emergencies
 - Allows for review of lectures
 - Self-monitoring for:
 - Content
 - Style
 - Interaction
 - Further reach of lecturing ability
 - Students can take the class when (very) far away
 - Issues/concerns need to be explored



UC21st. 21.Jun.2008

UC SANTA CRUZ, AUTONOMOUS SYSTEMS LAB





The Double Edged Sword

- Course Recording enables:
 - A drop in student attendance
 - Students are more likely to stay away when they know they will not suffer consequences
 - Minor illness and inconvenience become reasons to stay home
 - Attendant loss of interaction in class
 - Laziness in note taking
 - Longevity of lectures
 - Permanent recording of your class
 - Loss of control over content
- Many of these issues can be addressed through the use of a few ground rules
 - Make lecture videos available, but not slides
 - Control video access with secure download areas



UC21st. 21.Jun.2008

UC SANTA CRUZ, AUTONOMOUS SYSTEMS LAB





The Equipment (1.2)

- Tablet PC – “Digital Chalkboard”
- Bluetooth Headset – audio capture
- Data Projector
- Camtasia™ Software – video capture
- Classroom Presenter Software – pen-based PowerPoint

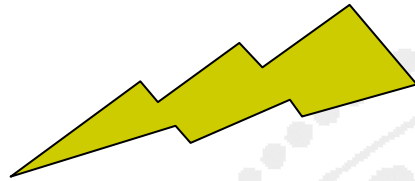
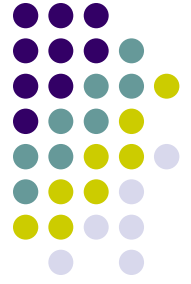


UC21st. 21.Jun.2008

UC SANTA CRUZ, AUTONOMOUS SYSTEMS LAB



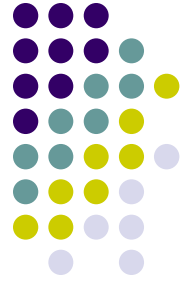
The Equipment (2.2)



UC21st. 21.Jun.2008

UC SANTA CRUZ, AUTONOMOUS SYSTEMS LAB





The Process

- Create slides in PowerPoint
- Export them to Classroom Presenter “CSD” format
- Start up the Camtasia Recording application
- Give your lecture
- Stop the recording
- Upload the file to website



UC21st. 21.Jun.2008

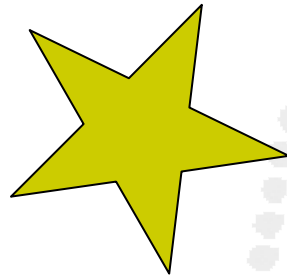
UC SANTA CRUZ, AUTONOMOUS SYSTEMS LAB



Classroom Presenter Capabilities



- Works closely with PowerPoint
- Allows for split views
 - What the classroom sees:



UC21st. 21.Jun.2008

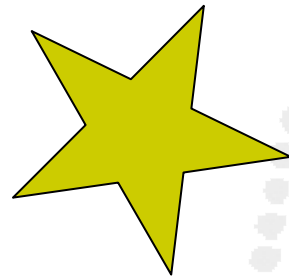
UC SANTA CRUZ, AUTONOMOUS SYSTEMS LAB



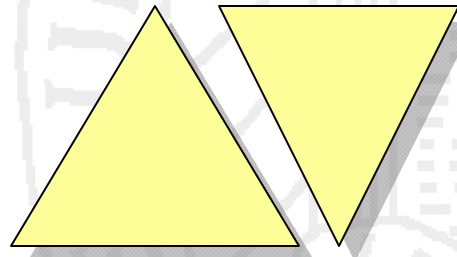


Classroom Presenter Capabilities

- Works closely with PowerPoint
- Allows for split views
 - What I see:



Call out that the star
can be a symbol for
something else



UC21st. 21.Jun.2008

UC SANTA CRUZ, AUTONOMOUS SYSTEMS LAB



Classroom Presenter



Presenter Administrator@TOSHIBA-TAB1.ConferenceXP.Net Classroom 1

File Connect Role View Help

- Classic topic in compiler courses: implementing a hash-based symbol table
- These days, use the Java collection classes (or equivalent in C#, C++, etc.)
- Map (HashMap) will solve most of the problems
- List (LinkedList) for ordered lists (parameters, etc.)

Symbol Tables for JFlat (1)

- Global
- 1 symbol table per class
- 2 entries for each method: class name, class name
- In final form, multiple symbol tables per method
- Each entry has a pointer to the class symbol table
- Reading: Stack & ??
- Done: TMU

Symbol Tables for JFlat (2)

- Global (cont)
- Single global table to map class names to class symbol tables
- Created in pass over class definitions
- Used in remaining parts of compiler to check field/method names and extract information
- All global tables persist throughout the compilation
- And beyond in a real Java or C# compiler...

```
class {  
  M1() {  
  }  
  M2() {  
  }  
  this: x;  
}
```

```
if () {  
  int x;  
}
```

```
void f(int n) {  
  int n;  
}
```

```
class C {  
  int v;  
  public C(int x) {  
    super();  
    this.x = x;  
  }  
}
```

© 2002 Hal Perkins & UW CSE

P 0 V 0 Current: 35 / 53 Slides: 53 |-semantics.csd Connected 234.5.5.5 port 5004



UC 21st. 21.Jun.2008

UC SANTA CRUZ, AUTONOMOUS SYSTEMS LAB





Storage Requirements

- Typical lecture video
 - 1 hour 45 minute lecture ~ 50 Mbytes
 - 45 Marked slides ~ 5 Mb
- Lossless CODEC's from Techsmith
 - Available for Windows XP
 - Available for Mac OSX
 - Available for Linux

- Can output to other formats



UC21st. 21.Jun.2008

UC SANTA CRUZ, AUTONOMOUS SYSTEMS LAB





The Results

- Feedback based on classroom surveys:

| Class | Years | Rate the Utility (1-5) | How Often Used (1-5) |
|-------------|---------------------|------------------------|----------------------|
| CMPE12 | 2004/2005 | 4.29/4.2 | |
| EE154/CE241 | 2005/2006/2007/2008 | 4.57/4.68/4.73/4.29 | 3.03/3.63/4.07/3.43 |
| CMPE240 | 2005/2006/2007 | 4.71/4.82/5 | 4.09/3.81/3.67 |
| CMPE118 | 2005/2006/2007/2008 | 4.2/4.4/4.57/4.29 | 2.7/3.2/3.57/2.85 |

- Feedback based on comments:
“This is just awesome!”



UC21st. 21.Jun.2008

UC SANTA CRUZ, AUTONOMOUS SYSTEMS LAB





Let's see the results

Dual System (3.3)

$$\begin{bmatrix} \dot{x} \\ y \end{bmatrix} = \begin{bmatrix} A & B \\ C & D \end{bmatrix} \begin{bmatrix} x \\ u \end{bmatrix}$$
$$\begin{bmatrix} \dot{z} \\ w \end{bmatrix} = \begin{bmatrix} A^T & C^T \\ B^T & D^T \end{bmatrix} \begin{bmatrix} z \\ v \end{bmatrix}$$



Gabriel Hugh Elkaim – Fall 2005



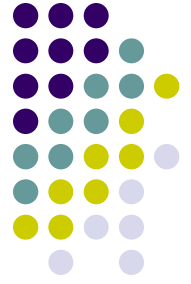
CMPE 240 – Intro. to Linear Dynamical Systems



UC21st. 21.Jun.2008

UC SANTA CRUZ, AUTONOMOUS SYSTEMS LAB





Replicating the process

- Engineering at UCSC
 - Five faculty recording their lectures
- Contacts with faculty at:
 - University of Colorado
 - University of Massachusetts
 - University of Minnesota
- Relatively easy process to replicate

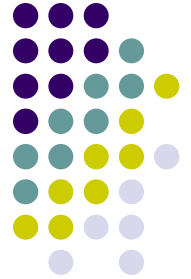


UC21st. 21.Jun.2008

UC SANTA CRUZ, AUTONOMOUS SYSTEMS LAB



Questions?



Gabriel Hugh Elkaim
Autonomous Systems Lab
Engineering 2 – 337B
UC Santa Cruz

elkaim@soe.ucsc.edu

(831) 459-3054



UC21st. 21.Jun.2008

UC SANTA CRUZ, AUTONOMOUS SYSTEMS LAB

