

LCC 6310
The Computer as an
Expressive Medium

Lecture 1

Overview

Go over the syllabus

Brief introduction to me and my work

Art, programming and Java

Syllabus

Signup sheet

If you're not listed, please add your name

If you're listed, please check off your name and update your information as needed (e.g. nickname, email, etc.)

Handout

Online

<http://www.lcc.gatech.edu/~mazalek/courses/fall07/lcc6310/>

Background

MS & PhD in Media Arts and Sciences (MIT Media Lab)


Tangible Media & Interactive Cinema research groups

MS: Tangible interfaces for interactive narratives

PhD: Media Tables - architecture & applications


Some of my past work...

geniebottles (2000)



Boundary objects that exist at the intersection of the story space where the characters live and the real space where the user lives

Tangible containers for digital stories
Tight coupling of interface and story



artists / researchers all mazalek, alison wood, hiroshi ishii

flights of fantasy (2001)

Interactive installation at the Decordova Museum



- Visitors assemble story sequences using a sliding blocks interface with an iconic language
- Large database of content that tells stories about a specific set of characters and locations



researchers barbara barry, win burleson, david crow, glorianna davenport, aisling kelliher, ali mazalek, paul nemirovsky, isaac rosmarin, james seo, arnan sipitakiat

tviews table (2001-present)

Tangible computer interfaces can bring people together around story exploration in the same way that board games bring friends and families together around game-play

- Design an interface that can sit in social and living spaces rather than in offices on desks
- Multiple users around one interface
- Face-to-face social interactions







photo browsing



TViews tables at MIT Media Lab & Samsung




researchers ali mazalek, glorianna davenport



tangible viewpoints (2001-02)

Storytelling engine for multi-viewpoint stories on the TViews table




- Physical pawns used as tangible embodiments of the character viewpoints in a story
- Story segments appear as thumbnails clustered around the pawns, fading in and out as viewers move forward through the story
- A spreading activation network is combined with a rule-based storytelling engine
- Allows the system to adapt to viewer preferences while still providing a coherent overall narrative

Kids sharing personal stories
Computer Clubhouse, Boston Museum of Science

researchers ali mazalek, glorianna davenport

tangible spatial narratives (2002-present)


Creating narrative landscapes in a physical form on the TViews table



Audiences collectively reflect upon and navigate complex spatially structured and multi-viewpoint stories


- Visual landscapes on the interaction surface provide a spatial framework for the story
- Different perspectives and narrative threads emerge as the story unfolds

Digital Dialogues: Technology and the Hand
Haystack Mountain School of Crafts, 2002



Multi-viewpoint spatially distributed stories

E.g. Spatial browsing of large media collections



Location-tagged photo collections

researchers ali mazalek, glorianna davenport

Some directions

I'm someone to chat with about...

- Tangible interfaces & media arts applications
- Emerging sensing technologies
- Physical fabrication (lasercutters & other fun tools)
- Interactive narratives (particularly multiviewpoint stories)

I direct the Synaesthetic Media Lab in the GVU
<http://synlab.gatech.edu/>

Introduce yourselves

- Name & place of origin?
- Background, academic & other?
- Research & other interests?
- Programming experience?
- Something interesting about yourself?

Programming languages

An abstract "human understandable" language for telling the computer what to do

The abstract language must be translated into the low level language understood by the machine

This translation is accomplished by an **interpreter** or **compiler**

We will be learning the compiled language Java

Some definitions:

A **compiler** is a program that processes statements written in a programming language and converts them into machine language, binary code, that a computer processor uses. E.g. Java, C, C++

An **interpreter** translates code one line at time, executing each line as it is translated. Generates binary code that is never compiled into one program entity, but interpreted every time the program executes. E.g. BASIC, JavaScript, Perl (can also be compiled)

A simple Java program

Human readable?!?

```
for (int i = 0; i < 10; i++) {  
    println(i);  
}
```

Just prints the numbers 0 to 9 on the screen

"Human readable" is relative

```
for (int i = 0; i < 10; i++) {  
    println(i);  
}
```



Java compiler translates this into...

Java VM assembly code

```
public static void                24:  invokevirtual #34  
main(java.lang.String[]);        27:  iinc     1, 1  
Code:                              30:  iload_1  
  0:  iconst_0                          31:  bipush  10  
  1:  istore_1                          33:  if_icmplt 5  
  2:  goto    30                          36:  return  
  5:  getstatic  
  8:  new  
 11:  dup  
 12:  ldc  
 14:  invokespecial #23                  test.PrintLoop();  
 17:  iload_1                            Code:  
 18:  invokevirtual #27                  0:  aload_0  
 21:  invokevirtual #31                  1:  invokespecial #43;  
                                       4:  return
```

Programming paradigms

A programming paradigm is a style of programming, complete with concepts & common practices that determine how the programmer will structure the program and think about its execution

Different programming languages advocate different programming paradigms

Not necessarily a one-to-one mapping, since some programming languages can support multiple programming paradigms

Some examples

C - Procedural programming

Java - Object-oriented programming

C++ - Supports elements of both

More about these paradigms...

Object Oriented vs. Procedural Languages

Procedural (e.g. C)

We create some data representing an image (array of pixels to draw on the screen)

We write a procedure than can be passed the image and that will draw it

Encourages programmers to think in terms of data and the functions that can be used to manipulate the data

Object Oriented (e.g. Java)

We create a class that contains an image AND a routine draw it

The data and the behavior (ability to draw) are in one "container"

Encourages programmers to think of their programs as sets of interacting "objects" that can operate on each other

A couple of Java's relatives

Smalltalk

Invented by Alan Kay and a group of researchers at Xerox PARC in the 1970s

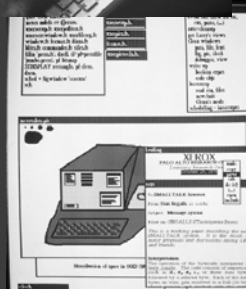
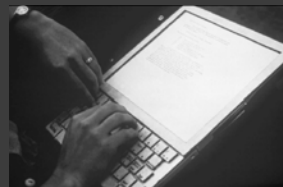
Designed for programming the Dynabook, a concept handheld computer that would enable learning, reading, and creative expression with different kinds of media (graphics, audio, etc)

Centered on the concept of objects

C++

Bjarne Stroustrup developed C++ in 1983 at Bell Labs as an enhancement to the C programming language

Used object-oriented programming to ease the management of big C programs



Java

Developed at Sun Microsystems in the early 1990s

Designers started with C++

Smaller

Simpler

Safer

Programming embedded systems

Toasters, microwave ovens, TV set top boxes

Reliability very important--avoid costly recalls

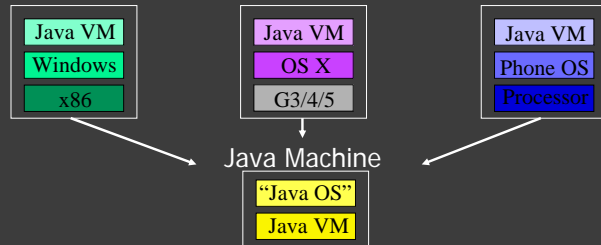
Web programming

Incorporated into web browsers at critical moment

The virtual machine

Since Java was designed to run on embedded systems, it was designed around a virtual machine

"Write once, run everywhere"



But we're using Processing

Processing is built on top of Java

Supports script-like coding

Easy to get simple programs up fast

But allows transition to full Java programming

Has built-in methods and classes to make drawing easy

Easy to export program to applet

The Processing environment



Menu
Toolbar (run, stop, new, open, save, export)
Tabs

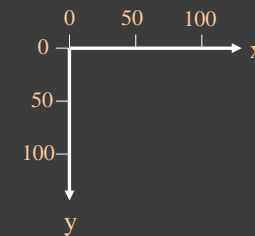
Text editor (this is where you type code)

Message area (feedback, errors)
Text output (print commands)

Drawing in Processing

Automatic creation of display window

Window has a coordinate system for drawing



Let's draw a point: point()

point(x, y) – draws a point at the location x, y

Let's try it in [Processing](#):

```
point(50, 50)
```

Unexpected token: null – what the #@#\$!?!

Compiler errors appear in the bottom pane

All lines must be terminated with a semicolon ;

Drawing several points

```
point(30, 20);  
point(85, 20);  
point(85, 75);  
point(30, 75);
```

Comments

Comments are non-program text you put in the file to describe to others (and yourself) what you're doing

Important for being able to look back at your code and understand it

Single-line comments begin with //

Multi-line comments begin with /* and end with */

Commenting and uncommenting lines useful for figuring out code

Drawing shapes: some primitives

```
line(x1, y1, x2, y2)
```

```
triangle(x1, y1, x2, y2, x3, y3)
```

```
rect(x, y, width, height)
```

rectMode() – CORNER, CENTER_DIAMETER, CORNERS

```
ellipse(x, y, width, height)
```

ellipseMode() – CORNER, CENTER_DIAMETER, CORNERS,
CENTER_RADIUS

Controlling color and line

Colors represented as Red Green Blue (RGB) values

Each one ranges from 0 to 255

Can also use Hue Saturation Value (HSV) space, but we won't worry about this for now

background(R, G, B) – set the background color

stroke(R, G, B) – set the colors of the outline (default black)

noStroke() – no outline drawing around shapes

fill(R, G, B) – set the fill color for shapes (default white)

noFill() – don't fill the shapes (background shows through)

strokeWeight(w) – line width for outlines (default 1)

Playing around

To learn how to program it is necessary to play around with code!!!

Don't just wait for me to show you things

Processing makes this easy

Use the Reference in the Help menu (local version of website [reference](#))

Play with the examples

Saving your work

You should install Processing on your own machine

Do this for Thursday!

Processing saves all projects in a default directory – no way to change it

This means that you should always copy your code to your local disk

Don't depend on your project remaining undisturbed on lab machines

Things to do for Thursday

Readings

Three students: present one reading each

Everyone else: prepare one discussion question for each reading

From Software: Exhibition at the Jewish Museum (NMR p.247)

Four Selections by Experiments in Art and Technology (NMR p.210)

Concepts, Notations, Software, Art (Linked from course website)

Processing

Download and install on your own laptops or home machines

<http://www.processing.org>

Programming concepts this week:

Processing, pp. 58-75, 109, 340-349

Processing, pp. 85-88, 482-486, 564-570, 603-605