Lecture 11: Web Usability

Today’s lecture is about designing your web sites for maximum usability. I’ll also cover some object oriented design concepts. Did you miss the lecture? I suggest you ask a colleague what was covered, as I have listed points here that I expanded in person.

Today we’ll cover:

- Pretty pages versus powerful pages.
- Screen real estate
- Browser compatibility
- Response times
- Navigational issues
- Search
- Writing for the Web
- Object oriented design

Pretty vs. powerful pages

In his book “Designing Web Usability”, Jakob Neilsen discusses the idea of Art versus Engineering. He states that designing usable web pages is more a matter of engineering than it is art. Of course, completely artless pages aren’t a good idea either, but a nice arty page that isn’t usable is… useless. That’s why in general, functional prototypes are developed when designing web sites. Don’t ignore the role of good graphics design though.

Screen Real Estate

Recall your tutorial exercise here, in which you were asked to identify and classify the various areas on screen for a web page. The area occupied on screen by an element should be appropriate for its usage. Sometimes blank space, which you might consider to useless, has its purpose- if a page’s design lends itself to directing the user’s eye to the right location, it’s done its job. Notice how I provide lots of examples of correct punctuation use, so there’s no excuse not to get it right in the second assignment!

Btw:
Loud!
Louder!!
Crazy!!!

Browser compatibility

A site’s usability is governed by the situations in which it is usable. If a number of browsers aren’t supported correctly, then your site is useless on those browsers. Here’s a worthwhile exercise-install as many browsers as you can, and look at your page in those browsers. Snapshot your
screen so you can easily compare (for example in Photoshop) what’s going on in terms of differences between them.

Ideally you would design a single page that renders correctly on all browsers, in practice you will need to make allowances for individual browsers and platforms. Have a look at the source for TiddlyWiki for examples of this.

**Load and Response times**

Here’s a slice of media pizza- it’s heavy at the bottom, and light at the top. Bear this in mind when you think of load times. Response time is the time it takes for a web page to react to some input. Remember the rule of thumb:
.1 sec feels like realtime
< 1 sec feels responsive
<10 for a new page to load.

Navigational issues
http://www.useit.com/alertbox/ia.html

Search
A nice shortcut to usability is to provide a search function within your site. Users know that (if the search is implemented correctly) that their search will only be within your site, and this assures them that their search was both exhaustive and efficient.

Writing for the Web
http://www.useit.com/alertbox/search-keywords.html

Web references for web usability
6 Ways to Fix a Confused Information Architecture
http://www.useit.com/alertbox/ia.html

Use Old Words When Writing for Findability
http://www.useit.com/alertbox/search-keywords.html

Object Oriented concepts

Methods and Attributes
Methods are things that an object can do, and attributes are things that the object is (or has). Another term for method is ‘function’, and another term for attribute is ‘variable’. A class has methods and attributes.

Objects are Instances of Classes
Once you’ve defined a class, you can make one or more instances of that class. Useful when you’re animating a large number of objects that share similar behaviour. An object has instances of the attributes of its class. For example, a ‘Ball’ class might have the attributes ‘position’ and ‘bounciness’, and each instance of the ‘Ball’ class would have different values for those attributes.

Classes may inherit from other Classes
A class may inherit the methods and attributes of another class (generally only one class) and may extend or inhibit
those attributes and methods.

**Classes should be encapsulated**

It’s perfectly possible to access the attributes of an object, but that doesn’t mean you should. A better idea, where possible, is to define methods that access those attributes, and ensure that the attributes possess valid values.

**Containment**

If an object A has as one of its attributes an object B, then A is said to contain B. An example of this might be a ‘World’ object that contains a single ‘Player’ object, and several ‘Enemy’ objects. Another word for ‘contains’ is ‘manages’. One rule of style is that a contained object shouldn’t be ‘aware’ of its containing object, although this is tough to achieve in practice.

**Dot Syntax**

One means of accessing the attributes and methods of an object that is used in many languages, including Flash and JavaScript is Dot Syntax. For example:

Object.attribute

Or

Object.method(parameters)

---

**Seamless Node Animation**

**The Idea**

The idea behind S.N.A. is this- whenever the User chooses to navigate through the nodes of the project (In multimedia that’s changing screens, and in character animation that’s the character changing poses), they should see a seamless transition between these nodes. In the character example, it shouldn’t jerk between poses, and in the multimedia example, there shouldn’t be a flicker between screens. This becomes more challenging to achieve the more involved our animation becomes. Here’s an easy example:
Figure 1: Five simple poses in a nodal animation.

In the situation depicted above, a central ‘idle’ pose is connected to four other possible poses, which are in turn connected to the idle pose. If the drawings are aligned correctly (a big ‘if’ in my case) then we can be confident that we’re able move the character into these poses and back again without a jump. Here’s a quick Flash movie of this in action. I’ll show you the object oriented aspects of this in a minute. It might not be quite what you expect...

Now it gets complicated

The situation gets more challenging when instead of still frame nodes, we instead have cycling animations for each node. Consider an idle animation that shows our character performing some action over and over again. For example, he could bop up and down, click his fingers, or look around the room. When this character is told to assume a new pose, an issue arises- do we really want to have to produce as many interconnecting frames as there are frames in each animation? Here’s a schematic of what I mean:
In this schematic, I’ve drawn a connection between every frame, and only the first frame of another node sequence. I’ve only used four node sequences, for clarity. If each of these lines represents a connecting frame of animation, then you can see that there are now a great deal of new frames we have to come up. If any frame in a given node sequence could connect to any frame in another sequence, then we’d have even more. If you’re scratching your head at this point, it might be nits, and if you’re confused, then just imagine that each of these four clusters of nodes is an animation, and that each connection is what happens when we ‘ask’ the character to start animating a new pose. There has to be a better way.

And There Is

A solution to this problem (which gets a lot worse as we add poses and frames per pose) is to limit the frames that are permitted to transition to other frames. This wouldn’t be very good for a game, where the User says “go right” and the character doesn’t respond immediately, but for a scripted animation, or a multimedia project where our character is assisting in a transition, it works just fine. Here’s another schematic:

In this situation, regardless of which frame is currently being played, the animation will continue until it reaches the frame with the connection, ensuring that the transition will always occur on the frame that you’ve designed it to. But how does this relate to Flash?
**Backgrounds and Characters as Objects**

**Telling Flash where to go**

Let’s get a little practical for a moment. I’m hoping at this point that you’ve got your head (and mouse) around the idea of using simple actions and events to control the flow of playback of your Flash movies. Now it gets just a little more advanced. All we’re going to do here (and I sincerely believe that this is an appropriate level of programming for you guys) is to direct the flow of movieclips within our scene.

I’ll show you an example now of how you can define a character as a movieclip, and control its timeline with simple “gotoAndPlay” commands. The ideas presented here apply equally to projects that are more of a standard multimedia nature as well- just think of backgrounds as screens, and characters as foreground elements. The idea is to “tell” an object what to do. In most OO design, a programmer would define a variety of methods that the object could perform. Feel free to think of things you might like the object to do, such as change its position, its alpha value, etc, but all we’ll cover today is telling the movieclip to go and play a particular frame. Here are the basic steps:

1. In a movie clip, define (by adding a Keyframe) and then name your labels.
2. Add appropriate scripts to control the timeline- that is, either stopping on a frame, or jumping back to another label.
3. Place media in those frames to achieve the effect you want- I’ll show you the character example, and the bouncing ball example.
4. Embed this movieclip within another movieclip, and name your instances.
5. Use Actionscript to ‘tell’ the movieclips which frames to play.

You can tell the instance “i_object” to “gotoAndPlay” frame “f_Idle” in at least two ways:

- `i_object.gotoAndPlay("f_Idle");`
- `with (i_object) { gotoAndPlay("f_Idle");`)

The first method won’t complain if such an instance doesn’t exist, the second method will. I’d recommend the second method because it will highlight problems immediately.

**Watch Out!**

I had some difficulties with preparing the material for today’s lecture- the Flash component, no surprise, but the good news is that my frustration is your liberation, because I learned something today. It has to do with instance naming and referring to those instances within a script. Knowing which object you’re referring to in a project is absolutely vital. You’ve seen this before in JavaScript’s Document Object Model. You see it at uni where we use student ID’s to refer to you! In Flash, it can be problematic. The issue occurs when you place Instances of Movieclips within other Instances of Movieclips. Here’s an example that works:

```javascript
www.arch.usyd.edu.au/~jcla8108/deco2102/mc_in_mc.zip
```

However, in the dancing dude example I showed you, I didn’t seem to be able to refer to the instance of my character by its instance name.

The same happened in the Bill, Bob and Bert example:
I think I’ve found out why, initially he didn’t exist in the first frame of the movieclip, or at some point the instance wasn’t named. Once I did that, and gave it the name as before, it works fine. You can verify this by setting a breakpoint and looking at the instances you see in the debugger.
Suggested Websites:

www.flashkit.com
www.friendsofed.com
http://www.devaricles.com/c/a/Flash/Basic-Flash-ActionScript-for-Designers/
http://www.senocular.com/flash/tutorials/faq/

Sams Teach Yourself Flash™ MX ActionScript in 24 Hours
By Gary Rosenzweig

Object-Oriented Programming with ActionScript - New Riders