Using Flash to develop interactive listening guides for online distribution

Scott D. Lipscomb, *Northwestern University*, lipscomb@northwestern.edu
Marc Jacoby, *VanderCook College of Music*, mjacoby@vandercook.edu

Presented at the ATMI Conference in San Francisco, CA
November 5, 2004

BubbleMachine available for free download from:
http://faculty-web.at.northwestern.edu/music/lipscomb/bubblemachine/

Navigating through music: The bubble graph

- Voyager series – "guided tours" through musical masterpieces
- Rolling your own
  - Clip Creator (Mac)
  - TimeSketch Editor (recent version now available for Mac & PC)
- QuickTime
- Director (accessing cuePoints embedded in sound files)
- Flash
  - using *Stream* sync mode
  - Sound object and ActionScript
- BubbleMachine™ (Lipscomb & Jacoby)

[brief demos]

BubbleMachine: Our goals

- Flash-based, distributed in SWF format
  - completely cross-platform & cross-browser compatible
- limit file size for online dissemination
  - utilizing streaming MP3 files, rather than embedded
- interactive tools for the creation of bubble graphs
  - fine resolution for creating markers (to the millisecond)
- reusable player with varying content
- two modes
  - play only
  - create/edit

BubbleMachine – Issues we faced

- accessing the sound file and use of Sound object methods (stop, start, & volume)
- creating markers
  - array manipulation (add, delete, edit, and sort items)
  - visual representation and editing access via a listbox
  - fine resolution for marker placement (to the millisecond)

- navigate through sound file
  - interactive slider bar
  - updating `currentLocation` textbox
- realtime bubble graph generation
- saving data (security issues)
  - write to text file (user must copy-and-paste) and save manually

Code examples (ActionScript)

creating a Sound object:

```
// Create a Sound object
var mySound = new Sound();
// Load an external sound file, reduces swf size and rendering time.
mySound.loadSound(soundChoice, false);
```

initiate playback

```
function playMe() {
   mySound.stop();
   playFrom = _global.currLocation;
   playTempLoc = _global.currLocation;
   //Start sound file **must be in seconds**
   mySound.start(_global.currLocation);
}
```

adding a marker

```
//CONTROLS THE LIST OF MARKERS (Time and form label)
//  AND OUTPUT TO TEXTFIELD
function setMarkerList() {
   if (_global.currLocation>0) {
         listMarkers.removeAll();
         locationList.push(_global.currLocation);
         locationList.sort(sortAscendingNumbers);
         upDateList();
   }
}
```

update listbox

```
// This does the updating of the list box component
function upDateList() {
   for (x=0; x<locationList.length; x++) {
         listMarkers.addItem(locationList[x]);
   }
}
```

fine tune current location (using Control/Command, Shift, and Alt/Option)

```
// USES UP AND DOWN ARROW BUTTONS to fine tune the current location
// increment =
//       no modifier key   = 1 ms
//       with SHIFT        = 10 ms
//       with CTRL/Command = 100 ms
//       with ALT/Option   = 1000 ms (or 1 sec)
function stepUp() {
   switch (Key.getCode()) {
   case Key.SHIFT :
         _global.currLocation = _global.currLocation+.01;
```

```
            break;
      case Key.CONTROL :
            _global.currLocation = _global.currLocation+.1;
            break;
      case (245) :
            _global.currLocation = _global.currLocation+1;
            break;
      default :
            _global.currLocation = _global.currLocation+.001;
      }
       _global.currLocation = ↵
   _root.formatDecimals(_global.currLocation, 3);
      _root.setLocationText = _global.currLocation;
      _root.timecue.slideBar._x = ↵
   int(TOTAL_WIDTH*(_global.currLocation/_root.songDuration));
      playTempLoc = _global.currLocation;
      playFrom = _global.currLocation;
   }
```

navigate sound file using scrollbar (code placed on the scroll button of the slider movie clip)

```
   on (press) {
      _global.drag = true;
      this.startDrag(true,0,0,500,0);
   }
   on (release, releaseOutside) {
      _global.drag = false;
      this.stopDrag();
      _global.currLocation = ↵
   (this._x/_parent.track._width)*_root.songDuration;
      _root.mySound.stop();
      _root.mySound.start(_global.currLocation);
   }
```

generate bubble graphs using values in the marker array

```
   // generate bubble graph based on total width of 500 px
   //   representing entire duration of musical selection
   for (x=0; x<locationList.length - 1; x++) {
      //increment variable tracking bubbles
      numBubbles++;
      // use duplicateMovie to duplicate movie clip present on stage
      //   … including attached scripts
      bubbleTemplate.duplicateMovieClip("bubble"+numBubbles, ↵
   numBubbles);
      // calculate appropriate starting _x position and width
      //   of bubble based on difference (in ms) between current marker
      //   value and the next item in the array
      bubbleStart = int(BUBBLE_OFFSET + ↵
   TOTAL_WIDTH*(locationList[x]/_root.songDuration));
      bubbleWidth = int(TOTAL_WIDTH*((locationList[x+1]- ↵
   locationList[x])/_root.songDuration));
      _root["bubble"+numBubbles]._y = BUBBLE_BASE;
      _root["bubble"+numBubbles]._x = bubbleStart;
      _root["bubble"+numBubbles]._width = bubbleWidth;
      _root["bubble"+numBubbles]._height = LAYER1_HEIGHT;
      _root["bubble"+numBubbles].startPos = locationList[x];
   }
```

select all text contained in a dynamic textfield

```
// use Selection object to allow user access to content of textfield
//   containing all variable values, including the markerList array
Selection.addListener(textHolder);
// set focus to dynamic textfield
Selection.setFocus(textHolder);
// select entire string, so all user has to do is copy (CTL+C)
//   and paste (CTL+V) into a text editor
Selection.setSelection(0, textHolder.length);
```