

Modifying All Samples in a Sound

Barb Ericson
Sept 2010

08-ModifyingAllSamplesInASound

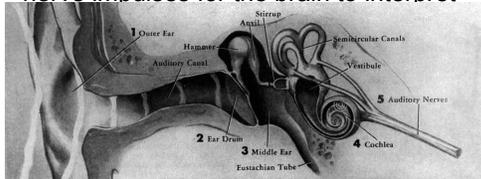
Learning Goals

- How is sound digitized?
- Figure out how many bits are need to store sound values.
- Understand and use a one-dimensional array
- Use iteration (for-each, while, and for loops) for manipulating sounds
- Use conditionals (if-else) when manipulating sounds

08-ModifyingAllSamplesInASound

How does Hearing Work?

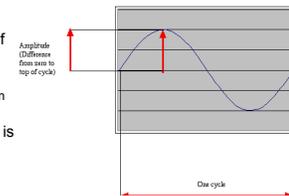
- The outer ear “catches” sounds
- The eardrum vibrates
- The inner ear translates the vibrations to nerve impulses for the brain to interpret



08-ModifyingAllSamplesInASound

Acoustics, the physics of sound

- Sounds are waves of air pressure
 - Sound comes in cycles of compressions and rarefactions
 - Increases and decreases in air pressure
 - The *frequency* of a wave is the number of cycles per second (cps), or *Hertz*
 - The *amplitude* is the maximum height of the wave



08-ModifyingAllSamplesInASound

Volume and Pitch

- Our perception of volume is related (logarithmically) to changes in amplitude
 - If the amplitude doubles, it's about a 3 decibel (dB) change.
 - A *decibel* is a ratio between two intensities: $10 * \log_{10}(I_1/I_2)$
 - As an absolute measure, it's in comparison to threshold of audibility
 - 0 dB can't be heard.
 - Normal speech is 60 dB.
 - A shout is about 80 dB
- Our perception of pitch is related (logarithmically) to changes in frequency
 - Higher frequencies are perceived as higher pitches
 - We can hear between 5 Hz and 20,000 Hz (20 kHz)
 - A above middle C is 440 Hz

08-ModifyingAllSamplesInASound

Try It

- Download MediaTools
 - <http://home.cc.gatech.edu/TeaParty/43>
- Double click Squeak.exe on windows or mediatools-v4-sa.image on Apple machines
 - Then click on SoundTools
- Click on Record Viewer and Record
 - And do high and low sounds in the microphone
 - Notice that high sounds have more cycles per second
- Click on Quit to exit Squeak

08-ModifyingAllSamplesInASound

Sound Tools in Squeak

08-ModifyingAllSamplesInASound

Digitizing Sound

- In calculus you learn to estimate a curve by creating rectangles
- We can do the same to estimate the sound curve
 - Analog-to-digital conversion (ADC) will give us the amplitude at an instant as a number: a *sample*
 - How many samples do we need?

08-ModifyingAllSamplesInASound

Nyquist Theorem

- We need twice as many samples as the maximum frequency in order to represent (and recreate, later) the original sound.
- The number of samples recorded per second is the *sampling rate*
 - If we capture 8000 samples per second, the highest frequency we can capture is 4000 Hz
 - That's how phones work
 - If we capture more than 44,000 samples per second, we capture everything that we can hear (max 22,000 Hz)
 - CD quality is 44,100 samples per second

08-ModifyingAllSamplesInASound

Digitizing Sound in the Computer

- Each sample is stored as a number (two bytes)
- What's the range of available combinations?
 - 16 bits, $2^{16} = 65,536$
 - But we want both positive and negative values
 - To indicate compressions and rarefactions.
 - What if we use one bit to indicate positive (0) or negative (1)?
 - That leaves us with 15 bits
 - 15 bits, $2^{15} = 32,768$
 - One of those combinations will stand for zero
 - We'll use a "positive" one, so that's one less pattern for positives so the range is from -32,768 to 32,767

08-ModifyingAllSamplesInASound

Storing Sound Values

- If we capture 44,100 elements for every second of sound
 - How will we keep track of all of these?
 - We wouldn't want to name them all
- We use an array
 - Stores contiguous data of the same type
 - Use an index to access array data
 - Indices start at 0, last is length - 1

59	39	16	10	-1	...
0	1	2	3	4	

08-ModifyingAllSamplesInASound

Playing a Sound

- We can create a Sound object just like we created a Picture object
 - Get a file name and save a reference to it
 - String fileName = FileChooser.pickAFile();
 - Pick a file that ends in .wav
 - Create the sound object by asking the class to create a new Sound object and initialize it by reading data from the given file name
 - Sound sound1 = new Sound(fileName);
 - Play the Sound
 - sound1.play();

08-ModifyingAllSamplesInASound

Working with a SoundSample array

- You can get the length of the array

```
System.out.println(sampleArray.length);
```
- You can get back a SoundSample object

```
SoundSample sample = sampleArray[0]; // first
SoundSample sample2 = sampleArray
[sampleArray.length-1]; // last
```
- You can get and set the value

```
System.out.println(sample.getValue());
sample.setValue(0);
```

08-ModifyingAllSamplesInASound

Modifying Sounds

- To modify a sound
 - Create a Sound object from a file that holds data for a sound
 - Get the array of SoundSample objects from the sound
 - Loop through the SoundSample objects
 - Modify the value at each sample
 - Optionally write the modified sound to a file
 - Using the write(String name) method

08-ModifyingAllSamplesInASound

Doubling all the Sound Values

- You could change each SoundSample by hand
 - There are 8808 SoundSamples in croak.wav
 - Do you really want to do that?
 - How long would it take you?
- Let's let the computer do it in a loop
 - For-each
 - While
 - For

Intro-Sound-part2

21

For-Each Loop (Java 5.0+)

- For each of the elements in a collection of objects do the body of the loop
 - Each time through the loop the variableName will refer to a different object in the collection

```
for (type variableName : collection)
{
    // statement to repeat
}
```

Intro-Sound-part2

22

Comparison to Alice

- In Alice you had each rockette kick up her leg one at a time by having a list of rockettes and telling each one in the list to do the action
 - This is similar to a Java for-each loop



```
For all World.dancers, one item_from_dancers at a time {
    rockette.kickUpRightLeg ( whichRockette = item_from_dancers );
}
```

Intro-Sound-part2

23

Increase Volume with For-Each Loop

```
public void increaseVolume()
{
    SoundSample[] sampleArray = this.getSamples();
    int value = 0;

    // loop through SoundSample objects
    for (SoundSample sample : sampleArray)
    {
        value = sample.getValue(); // get the value
        sample.setValue(value * 2); // set the value
    }
}
```

Intro-Sound-part2

24

Testing increaseVolume

```
String file =
    FileChooser.getMediaPath("gettysburg10.wav");
Sound soundObj = new Sound(file);
soundObj.explore();
soundObj.increaseVolume();
soundObj.explore();
```

Intro-Sound-part2

25

Decrease Volume Exercise

- Write a method to decrease the volume of the sound
 - decreaseVolume()
 - Divide each value by 2
- What parts need to change from the last method?
 - Only the calculation of the new value
- Try it:


```
Sound s = new Sound(
    FileChooser.getMediaPath("gettysburg10.wav"));
s.explore();
s.decreaseVolume();
s.explore();
```

Intro-Sound-part2

26

What Does For-Each Do?

- It uses each element in the array one and only one time
 - So it needs to keep track of the current index
 - It needs to check that the current index is less than the length of the array
 - And stop when they are equal
 - It needs to pull out the item at the index
 - It needs to increment the index after the loop body has executed
- This is explicit in a while loop

Intro-Sound-part2

27

While Loop

- Loops while a boolean (true or false) test is true
 - When the test is false execution continues with the first statement after the loop
- ```
while(test)
{
 // statements to be done while the test is true
}
```
- To use a while loop you may need to declare variables before the loop and change them in the loop

Intro-Sound-part2

28

## Alice While Loop

- In Alice we used a while loop to have a shark chase a goldfish



Intro-Sound-part2

29

## While Loop to Process Sound Samples

```
int index = 0; // starting index
SoundSample sample = null; // current sample obj
int value = 0; // value at sample
while (index < sampleArray.length)
{
 sample = sampleArray[index]; // get current obj
 value = sample.getValue(); // get the value
 sample.setValue(value * 2); // set the value
 index++; // increment index
}
```

Intro-Sound-part2

30

## Shortcuts for Common Operations

- In programming you often need to add one to a value
  - index = index + 1;
- You may use the shortcut
  - index++;
- If you wanted to subtract 1 instead
  - index = index - 1;
  - index--;

Intro-Sound-part2

31

## Increase Volume with While Loop

```
public void increaseVolumeWhile()
{
 SoundSample[] sampleArray = this.getSamples(); // get array
 int index = 0; // starting index
 SoundSample sample = null; // current sample obj
 int value = 0; // value at sample

 // loop through SoundSample objects
 while (index < sampleArray.length)
 {
 sample = sampleArray[index]; // get current obj
 value = sample.getValue(); // get the value
 sample.setValue(value * 2); // set the value
 index++; // increment index
 }
}
```

Intro-Sound-part2

32

## Testing increaseVolume

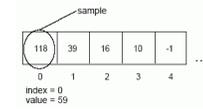
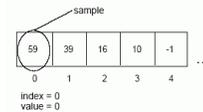
```
String file =
 FileChooser.getMediaPath("gettysburg10.wav");
Sound soundObj = new Sound(file);
soundObj.play();
soundObj.explore();
soundObj.increaseVolume();
soundObj.play();
soundObj.explore();
```

Intro-Sound-part2

33

## Tracing Execution

- The index is set to 0
- The value is set to the value in the array at that index (59)
- The sample value at the current index is set to 2 \* value
- The index changes to the next index (1)
- We check if the index is less than the length of the array and
  - If so do the loop again
  - Else jump to the first statement after the loop



Intro-Sound-part2

34

## Memory versus Disk

- When we read from a file we read from disk into memory
  - Computers only do calculations in memory
- We change the values in memory
- The file on the disk hasn't changed
- To save our new sound we need to write a file to the disk
  - soundObj.write(fileName);

Intro-Sound-part2

35

## Decrease Volume Exercise

- Copy decreaseVolume and call the new method decreaseVolumeWhile
  - Modify it to use a while loop
- Try it:
 

```
Sound s = new Sound(
 FileChooser.getMediaPath(
 "gettysburg10.wav"));
s.explore();
s.decreaseVolumeWhile();
s.explore();
```

Intro-Sound-part2

36

### While Loop versus For Loop

- It is easy to make mistakes when you use a while loop for looping a set number of times
  - Forget to declare variables before the loop
  - Forget to increment the variables in the loop before the next test
- Programmers use a For loop when the number of times to loop is known
  - And a while loop when you don't know

Intro-Sound-part2

37

### For Loop

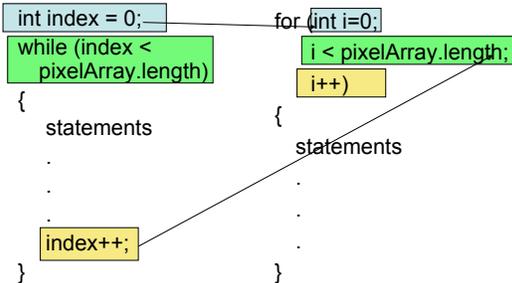
- A for loop allows you to declare and initialize variables, specify the test, and specify the way the variables change
  - All in one place
  - But, they still happen in the usual place

```
for(int index = 0;
 index < sampleArray.length;
 index++)
{
}
```

Intro-Sound-part2

38

### Comparison of While and For Loops



Intro-Sound-part2

39

### Alice For Loop

- In Alice we used a general for loop to cause a bunny to hop a set number of times

```
for (int index=0; index< 8 times ; index++) {
 bunny.hop ();
}
```

Annotations: initialize index to 0, test: index < 8?, increment index by 1

Intro-Sound-part2

40

### Increase Volume with a For Loop

```
public void increaseVolumeFor()
{
 SoundSample[] sampleArray = this.getSamples();
 SoundSample sample = null;
 int value = 0;

 // loop through all the samples in the array
 for (int index = 0; index < sampleArray.length; index++)
 {
 sample = sampleArray[index];
 value = sample.getValue();
 sample.setValue(value * 2);
 }
}
```

Intro-Sound-part2

41

### Modify decreaseVolume Exercise

- Copy decreaseVolume and create a decreaseVolumeFor
  - Modify it to use a for loop
    - Comment out declaration of the index
    - Comment out the increment of the index at the end of the loop
    - Comment out the while and put in a for loop
- Test it to make sure it still works
  - String file = FileChooser.getMediaPath("gettysburg10.wav");
  - Sound soundObj = new Sound(file);
  - soundObj.explore();
  - soundObj.decreaseVolumeFor();
  - soundObj.explore();

Intro-Sound-part2

42

## General Change Volume Method

- The methods `increaseVolume` and `decreaseVolume` are very similar
  - They multiply the current sound values by a given amount
    - To change this you would need to modify the method and compile
  - The methods would be more reusable if we pass in the amount to multiply the current sound values by
    - As a parameter to the method

Intro-Sound-part2

43

## What makes a good method?

- A good method should do one and only one thing
  - But be reusable
  - And other people should be able to understand it
- `increaseVolume` and `decreaseVolume` are very similar
  - Only one line is different
- We could write a more general `changeVolume` method
  - That takes a double factor to multiply by as a parameter

08-ModifyingAllSamplesInASound

## Using parameters in Alice methods

- In Alice we specified the number of times to spin with a parameter
  - `howManySpins`

```
public void spin (Number howManySpins) {
 // skater spins around
 // howManySpins specifies the number of revolutions for the spin
 iceSkater.prepareToSpin ();
 iceSkater.turn (LEFT, howManySpins.revolutions); more...
 iceSkater.finishSpin ();
}
```

08-ModifyingAllSamplesInASound

## General changeVolume method

```
public void changeVolume(double factor)
{
 SoundSample[] sampleArray = this.getSamples();
 SoundSample sample = null;
 int value = 0;

 // loop through all the samples in the array
 for (int i = 0; i < sampleArray.length; i++)
 {
 sample = sampleArray[i];
 value = sample.getValue();
 sample.setValue((int) (value * factor));
 }
}
```

Intro-Sound-part2

46

## Casting from a double to an int

- In the method `changeVolume` the factor is of type `double`
- The result of an integer times a double is a double
  - But a sound value must be an integer
- We need to cast from double to integer using `(int) (value * factor)`
- This will throw away any fractional part

08-ModifyingAllSamplesInASound

## Methods calling Methods

- One method can call another
  - Change `decreaseVolume` and `increaseVolume` to call `changeVolume`
- You can use
  - `this.changeVolume`
    - To invoke the method on the current object
- Or you can use
  - `changeVolume`
    - The "this" is implicit and will be added by the compiler

Intro-Sound-part2

48

### Force to Extremes

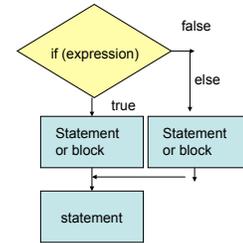
- What if we want to make all values in a sound the maximum positive or negative value?
  - If the value is 0 or positive make it 32,767
  - If the value is negative make it -32,768
- We need a way to execute code based on if a test is true
  - We can use a conditional (if and else)

Intro-Sound-part3

49

### Conditionals

- Allow you to only execute statements if an expression is true
  - Or (optionally) only if it is false
- If you clean your room
  - You can go out
- Else
  - You must stay home



Intro-Sound-part3

50

### Conditionals in Alice

- We used an if/else statement in Alice to return one value if the Boolean expression was true and another if it was false

```

 if (Math.abs(biphone.distanceAbove(otherPlane)) < minimumDistance) {
 return true ;
 } else {
 return false ;
 }

```

08-ModifyingAllSamplesInASound

### Setting Values to Extremes Exercise

- Write the method forceToExtremes()
  - Change all positive values (including 0) to the max positive value 32767
  - and change all the negative values to -32768.
  - Using a conditional
    - if and else
- Test with:
 

```

 String file =
 FileChooser.getMediaPath("preamble.wav");
 Sound soundObj = new Sound(file);
 soundObj.explore();
 soundObj.forceToExtremes();
 soundObj.explore();

```

Intro-Sound-part3

52

### Clipping

- When you force all values in a sound to the maximum or minimum values you will generate "Clipping" which sounds like static.
- Clipping happens when the normal curves of a sound are broken by the limitations of the sampling rate

08-ModifyingAllSamplesInASound

### Joining Boolean expressions

- You can join together two or more Boolean expressions using
  - And (&&) or Or (||)
- Try the following in the interactions pane
 

```

 > int x = 3;
 > if (x > 1 && x < 10)
 System.out.println("x is greater than 1 and less than 10");
 x is greater than 1 and less than 10
 > int y = -4;
 > if (y < 0 || y > 2) System.out.println("y is either less than 0 or
 greater than 2");
 y is either less than 0 or greater than 2

```

08-ModifyingAllSamplesInASound

### Using Sounds in Alice

---

- You can record sounds using the Sound Tools in MediaTools
- You can make sounds louder or quieter in Java
  - Then import them into Alice (File-Import)
- All Alice objects can play sounds
- If you want background music have the world play it when the world starts

08-ModifyingAllSamplesInASound

### Summary

---

- Arrays store data of the same type
  - Data can be accessed using indices
- You can use a for-each loop to loop through
  - all of the elements in an array
- You can use a while loop to repeat a statement or a block of statements
  - while a Boolean expression is true
- You can use a general for loop to repeat a statement or block of statements
  - a certain number of times
- You can conditionally execute statements

08-ModifyingAllSamplesInASound