Overview of Programming Techniques Covered so Far

Tim Capes

February 11, 2011

Defining a function

1. def is used to begin the definition
2. immediately after def is the name of the function
3. after that is () if the function has no input variables or a list of input variables inside the same type of brackets if it does.
4. lastly the function has its own block of code following which we start by using :

Function Definitions Example

def aPixelFunction(pixel):

1. def is used to begin the definition
2. this function is called aPixelFunction
3. it can take any variable from the command area in and will use the name pixel for it.
4. by using this name we are indicating we assume this variable will be a pixel and if its not then the program may have problems running.

Function Calls

1. First part, the name of the function you want to call
2. Second part, passing it any relevant variables
3. What it does is run whatever code is in the function

Function Calls Example function

We will call the function setRed, which takes a pixel and a number between 0 and 255 and sets the pixels red component to that value.

1. The first part of the call is the name of the function: setRed
2. The second part of the call is any relevant variables these are:
   - 2.1 the pixel: I will assume we will have a pixel called myPixel available
   - 2.2 the number: I want to set the red value to the constant 30 so I'll put that directly into the call
3. the call ends up being setRed(myPixel,30)

JES pixel functions

Here is a list of the pixel functions we can call in JES:

1. Those take a pixel as input and return something: getRed, getBlue, getGreen, getColor
2. Those that take a pixel and a value and set the component to that value: setRed, setBlue, setGreen, setColor
3. Two get functions we haven't made use of: getX, getY

JES picture functions

Here is a list of picture functions we can call in JES:

1. Those that get various information: getHeight, getWidth, , getPixels, getPixelAt = getPixel
2. Those that display the picture: show, repaint
3. We haven't used any functions that change the picture on a level above pixels.

JES data creation/manipulation

Here is a list of file manipulation functions we can call in JES:

1. Those that load files: pickAFile, getMediaPath
2. Those that configure: setMediaPath
3. Those that make objects out of files: makePicture, makeSound
4. Those that make empty objects: makeEmptyPicture, makeEmptySound

JES sample functions

Here is a list of sample functions we can call in JES:

1. to get information: getSampleValue, getSound
2. to change information: setSampleValue

JES sound functions

Here is a list of sound functions we can call in JES:

1. to get information: getSamples, getSampleObjectAt, getSampleValueAt, getDuration, getLength, getSamplingRate, getNumSamples
2. to set information we will manipulate sounds largely at the sample level.
3. to play sounds we use: play, blockingPlay

Understanding JES functions

For all the JES functions we use there are examples in the slides and text that should help make matters clear. The demo video on the course webpage as well as a demonstration in class covered how to go into the help in JES and look at these functions. That is your best resource for looking at what they do.

An example help document

getNumSamples(sound): sound: the sound you want to find the length of (how many samples it has) returns: the number of samples in sound Takes a sound as input and returns the number of samples in that sound. Example: def songLength(): sound = makeSound(r”C:
My Sounds
2secondsong.wav”) print getNumSamples(sound) This will print out the number of samples in the sound. For a 2 second song at 44kHz, it will print out 88000.

Figuring out what a function does from the help

What the help document will tell you is.. what the variables are (sound should be a sound object) and what the function will give you, in this case the number of samples you want to find. There is an example of using the function that should help you figure out how to call it.

for loops

For loops are the main tool we use for any repetitive operation for instance, changing pixels one at a time, or changing samples one at a time. A for loop consists of the following

1. the keyword for
2. the the variable which changes in each iteration of the loop
3. the keyword in
4. the array of things to iterate over
5. a : to start a new block of code
6. the contents of the actual loop

examples of empty for loops

1. for s in getSamples(sound):
2. for s in range(0,getNumSamples(sound)):

Here the block of code in the actual loop is omitted, there are a large number of examples with the loop blocks included in the slides. The first statement increments s over the contents of an array of samples(so s is a sample) while the second statement uses range to create an array of integers (so s is an integer).

range

1. the range function works by calling range with either two or three parameters
2. the first parameter is the start value
3. the second parameter is the end value
4. the last(optional) parameter is how much you increment by (if you don't say it's 1).

if statements

1. starts with the keyword if
2. then contains a conditional (true or false statement)
3. then has a : indicating the start of a block of code
4. it will execute this code only when the conditional is true

if statement examples, empty execution blocks

1. if $5 > 3$:
2. if $5 < 2$:
3. if *myInteger* > 5:

The first of these statements is always true so the code will always execute, the second if never true so the code will never execute. The third of these is a real use of an if statement, it uses a variable in the conditional, and depends on the value of that variable. If myInteger has a value which is greater than 5 then the code will execute if not it won't. To see if statements with blocks of code so the lecture slides.

max and min

The functions max and min take in a list of numbers (or some number variables) and return the maximum number or minimum number respectively.

1. max(3,5) will return the value 5
2. min(-2,4,8) will return the value -2
3. max(myVariable,myOtherVariable) will return whichever of these variables has a bigger value

print

This keyword is sometimes used in programs in class. It is used when we want to write something to the screen. I've avoided it in A1 and will avoid it in A2 because I wanted to avoid dealing with string manipulation extensively as you've already had to learn how to manipulate pictures, pixels, samples, sounds and integers.

1. print "Hello" +"World" will output HelloWorld
2. print "Hello" ,"World" will output Hello World

You won't need to use this on assignment A2 or for the tests or exams. But should be able to learn this should you need to for your own use.

Putting things together

In writing code, we want to put these various blocks together in a way that accomplishes a specific task. To manipulate a sound for instance I want to figure out what I want to do to each sample, and then set up a for loop which allows me to get at each sample and do what I want. If the choice of what to do doesn't depend on where the sample is in the file I can just use getSamples but if it does then I'm going to have to iterate over the indices so I'd instead use something along the lines of range(0,getSampleLength()) and then to get a sample in the loop I'd use getSampleAt(sound,s) where sound is the sound file I'm working on and s is my loop variable. When I choose this method of iterating over the sounds I know where in the file I am which is sometimes necessary (for instance when mirorring).

Questions

This document is intended to help clarify some issues that arose in a discussion with a student about some problems they were having with the course. It's not intended to answer all your questions, but it hopefully clarifies some. Ideally you will have more, as questions and discussion are a natural part of learning and they are encouraged in this course. Feel free to make use of office hours (mine and the TA's) setting up appointments (if you have conflicts with my office hours and would like to meet), and my e-mail (capestim@cs.toronto.edu) to ask questions.