MAGD 150 - LAB 9

WHAT WE BEEN COVERING

- Strings
- PFonts
- Processing PDF Library

FIRST STEPS

1. Open Processing
2. Go to Sketch> Import Library > Add Library
3. Search "Video"
4. Install the Video Library by The Processing Foundation
5. Search Sound
6. Install the Sound Library by The Processing Foundation

IMAGES

The PImage class is the datatype for storing images. Processing can display .gif, .jpg, .tga, and .png images.

Before an image is used, it must be loaded with the loadImage() function.

The PImage class contains fields for the width and height of the image, as well as an array called pixels[] that contains the values for every pixel in the image.

IMAGE FIELDS

pixels[]: Array containing the color of every pixel in the image

width: Image width

height: Image height

IMAGE METHODS

loadPixels() : Loads the pixel data for the image into its pixels[] array

updatePixels(): Updates the image with the data in its pixels[] array

resize(): Changes the size of an image to a new width and height

get(): Reads the color of any pixel or grabs a rectangle of pixels

set(): Writes a color to any pixel or writes an image into another

IMAGE METHODS

mask(): Masks part of an image with another image as an alpha channel

filter(): Converts the image to grayscale or black and white

copy(): Copies the entire image

blend(): Copies a pixel or rectangle of pixels using different blending modes

save(): Saves the image to a TIFF, TARGA, PNG, or JPEG file

save()

save() allows you to save a sketch as an image file.

Image File Types supported:

- .jpg
- .png
- .tiff
- .tga

EXAMPLE

Open Canvas

example_15_01_drawimage

EXAMPLE

Open Canvas

example_15_02_ImageSprite

EXAMPLE

Open Canvas

example_15_03_ImageArray

PIXEL ARRAYS

A PImage can be represented as an array of pixels. These values are of the color datatype.

This array is the size of the image, meaning if the image is 100 x 100 pixels, there will be 10000 values and if the window is 200 x 300 pixels, there will be 60000 values.

EXAMPLE:

- example_15_05_PixelArray

EXAMPLE:

- example_15_07_PixelArrayImage

VIDEO

The Video library plays movie files and captures video data from a camera.

Video can be captured from cameras connected to the computer.

Movies can be loaded from files located on your computer or anywhere on the Internet.

LIVE CAPTURE

Capture(this, requestWidth, requestHeight, frameRate)

LIVE CAPTURE METHODS

available(): Returns "true" when a new video frame is available to read

start(): Starts capturing frames from the selected device

stop(): Stops capturing frames from an attached device

read(): Reads the current video frame

list(): Gets a list of all available capture devices such as a camera

EXAMPLE

- 16_02_ManipulateCapture

PRE-RECORDED VIDEO

The Movie class is the datatype for storing and playing movies. Movies must be located in the sketch's data folder or an accessible place on the network to load without an error.

MOVIE METHODS

frameRate(): Sets the target frame rate
speed(): Sets the relative playback speed
duration(): Returns length of movie in seconds
time(): Returns location of playback head in units of seconds
jump(): Jumps to a specific location
available(): Returns "true" when a new movie frame is available to read.

MOVIE METHODS

play(): Plays movie one time and stops at the last frame
loop(): Plays a movie continuously, restarting it when it's over.
noLoop(): Stops the movie from looping
pause(): Pauses the movie
stop(): Stops the movie
read(): Reads the current frame

EXAMPLE

- example_16_04_MoviePlayback
- example_16_05_MovieScrub
- Exercise
  - Using example_16_04_MoviePlayback
    1. Change the speed or starting time of the movie
       - speed(), jump()
    2. Using transformations, change the look of the movie
       - rotate(), translate(), scale(), tint()

SOUND

The Sound library for Processing provides a simple way to work with audio. It can play, analyze, and synthesize sound.

NOTE: For best compatibility, use lossless **.wav** or **.aiff** files.

The library comes with a collection of oscillators for basic wave forms, a variety of noise generators, and effects and filters to alter sound files and other generated sounds.

LIVE SOUND

Similar to Capture, AudioIn takes in input from selected microphone or line-in device.

LIVE SOUND METHODS

- start(): Starts the input stream
- play(): Start the Input Stream and route it to the Audio Output
- stop(): Stop the input stream
- amp(): change the amplitude (from 0.0 to 1.0)
- add(): offset value for modulating audio signals
- pan():	Pan in a stereo panorama. -1.0 pans to the left channel and 1.0 to the right channel.
- set(): set amp, add, and pan simultaneously

EXAMPLE

- example_20_mic_input

SOUNDFILE METHODS

- **play():** Starts the playback of a sound file. Only plays the sound file once.
- **loop():** Starts the playback of a sound file to loop.
- **jump():** Jump to a specific position in the file while continuing to play.
- **cue()**: Cues the playhead to a fixed position in the soundfile. Note that the time parameter supports only integer values.
- **stop():** Stops the player

SOUNDFILE METHODS

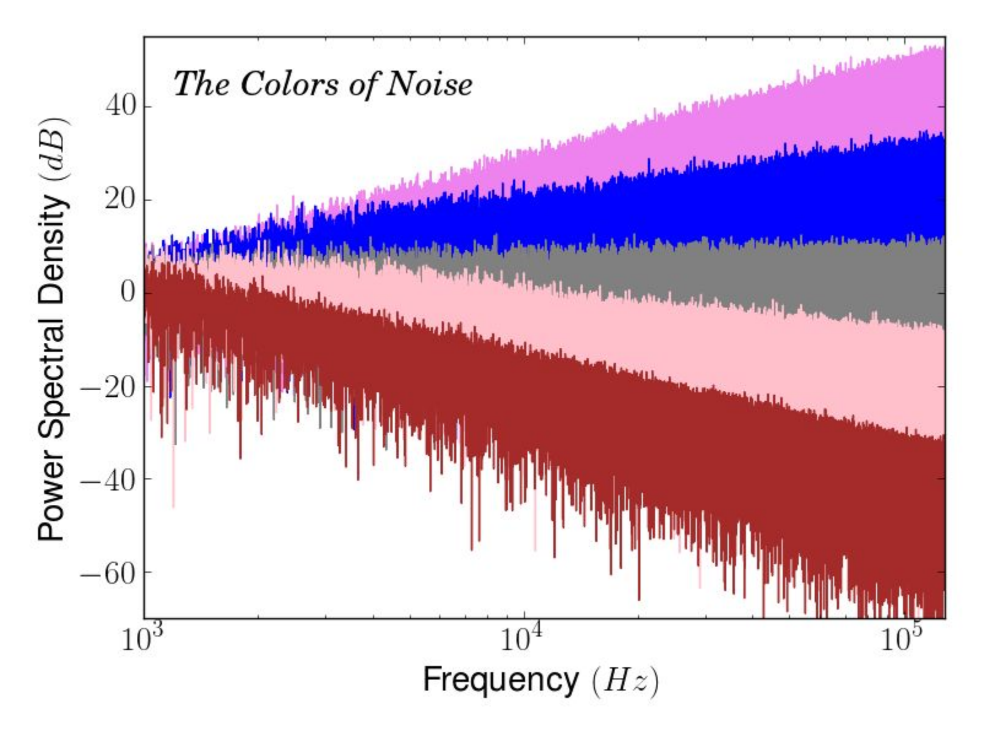
- frames() Returns the number of frames/samples of the sound file.
- sampleRate() Returns the sample rate of the soundfile.
- channels() Returns the number of channels in the soundfile.
- duration() Returns the duration of the the soundfile.
- set() Set multiple parameters at once
    - rate() Change the playback rate of the soundfile.
    - pan() Move the sound in a stereo panorama, only supports Mono Files
    - amp() Changes the amplitude/volume of the player.
    - add() Offset the output of the player by given value

EXAMPLE

- example_20_03_manipulatesound

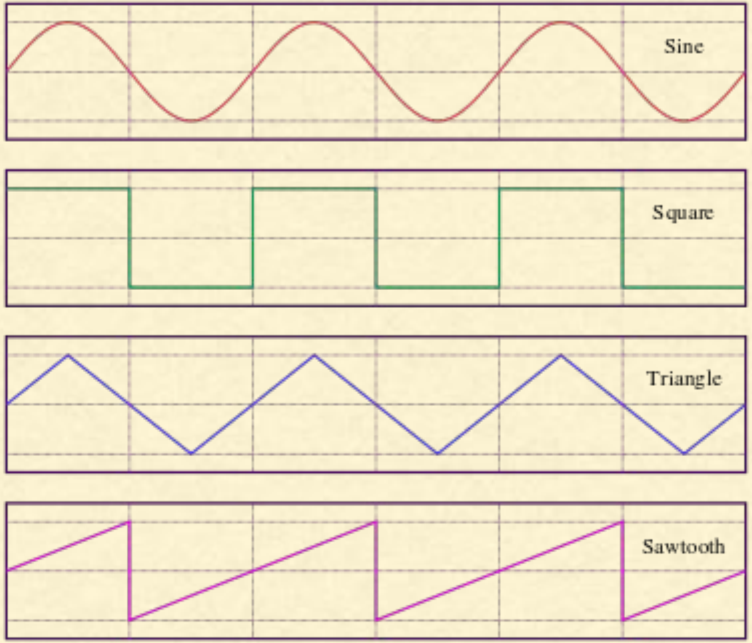
NOISE GENERATION

- Noise Generation
  - WhiteNoise
  - PinkNoise
  - BrownNoise



SIGNAL GENERATION

- Signal Generation
  - SinOsc //Sine Wave
  - SqrOsc // Square Wave
  - TriOsc // Triangle Wave
  - SawOsc // Sawtooth Wave



EXAMPLES

- example_20_08_noise
- example_20_06_oscillator_frequency

FFT(Fast Fourier Transforms)

The Fast Fourier Transform (FFT) analyzer calculates the normalized power spectrum of an audio stream the moment it is queried with the analyze() method.

example_20_13_soundfile_FFT