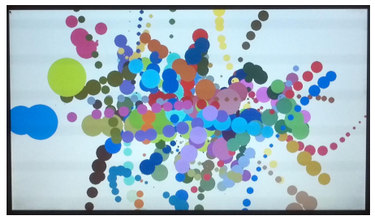
Pi Brush: Python Source Code

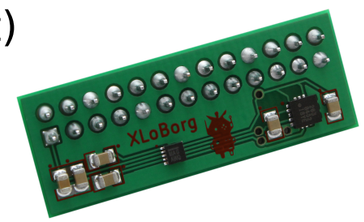
Display (Server)



mid-development photograph

```
import socket
import select
import pygame
import time
import numpy
import math
import datetime
import random

# ==============
# initialization
# ==============

# port to listen on
port = 5005

# setup networking
sock = socket.socket(socket.AF_INET,
        socket.SOCK_DGRAM) # UDP
sock.bind(("0.0.0.0", port))
sock.setblocking(0)

# screen solution
XRES = 1920
YRES = 1080

# setup display
pygame.init()
screen = pygame.display.set_mode((XRES, YRES))
# reset to white
screen.fill((255, 255, 255))
# push white
pygame.display.flip()

# length of moving average array
AL = 20

# accelerometer storage for moving average
AXa = numpy.zeros((AL, 1))
AYa = numpy.zeros((AL, 1))
AZa = numpy.ones((AL, 1))

# array index for accelerometer data
Ai = 0

# store gravity when fast is detected..
GX = 0
GY = 0
GZ = 1

# polar gravity
PGR = 0
PGA = -math.pi/2
PGB = 0

# screen gravity
PSGR = 1
PSGA = -math.pi/2
PSGB = 0

# accelerometer values
AX = 0
AY = 0
AZ = 0

# rotated for screen accelerometer values
GAX = 0
GAY = 0
GAZ = 0
last_G = 0

# timing information
last_time = time.time();

# brush info
BX = 0 # position
BY = 0
VX = 0 # velocity
VY = 0
P = 0 # amount of paint on brush
S = 0 # distance brush has traveled
last_stroke = 0

# seed random number generator
random.seed(time.time())

# =========
# functions
# =========

def polar(X, Y, Z):
    x = numpy.linalg.norm([X, Y, Z])
    if (x > 0):
        y = -math.atan2(Z, X)
        z = math.asin(Y / x)
    else:
        y = 0
        z = 0
    return (x, y, z)

def cartesian(X, A, B):
    x = 0 # don't bother - isn't used
    y = X * math.sin(B) * math.sin(A)
    z = X * math.cos(B)
    return (x, y, z)


# ============
# main program
# ============

fast = 0
notfast = 0
running = 1
while running:

    # move time forward
    dt = time.time() - last_time
    last_time = time.time()

    # no changes made to the screen so far
    draw = 0

    # check for keyboard input
    event = pygame.event.poll()
    if event.type == pygame.QUIT:
        running = 0
    elif event.type == pygame.KEYDOWN:
        if event.key == pygame.K_q:
            running = 0
        elif event.key == pygame.K_r:
            screen.fill((255, 255, 255))
            draw = 1
        elif event.key == pygame.K_s:
            filename = "%i.png" % time.time()
            pygame.image.save(screen, filename)


    # ==================
    # networking & sensor
    # ==================

    result = select.select([sock], [], [], 0)
    if len(result[0]) > 0:
        # read in data
        data = result[0][0].recvfrom(1024)
        a = data[0].split(",")
        AXa[Ai] = float(a[0])
        AYa[Ai] = float(a[1])
        AZa[Ai] = float(a[2])
        Ai = Ai + 1
        if Ai == AL:
            Ai = 0

        # moving averages for acceleration
        AX = numpy.sum(AXa) / AL
        AY = numpy.sum(AYa) / AL
        AZ = numpy.sum(AZa) / AL

        # combined acceleration for
        # working out resting gravity
        A = math.fabs(numpy.linalg.norm([AX,
                AY, AZ]) - 1)

        # in a slow moment store most recent
        # direction of the gravitational field
        if A < 0.02 and (last_time - last_G) > 0.12:
            GX = AX
            GY = AY
            GZ = AZ
            (PGR, PGA, PGB) = polar(GX, GY, GZ)
            last_G = last_time

        # rotate to screen coordinates
        # and subtract gravity
        (PAR, PAA, PAB) = polar(AX, AY, AZ)
        (GAX, GAY, GAZ) = cartesian(PAR,
                PAA - PGA + PSGA, PAB - PGB + PSGB)
        GAZ = GAZ - PGR


    # ==================
    # paintbrush physics
    # ==================

    # acceleration detection for paint strokes
    A = numpy.linalg.norm([GAY, GAZ])

    # detect moving quickly
    if A > 0.4 and fast != 1 and \
                last_time - last_stroke > 0.5:
        fast = 1
        notfast = 0
        scale = random.random() * 0.5 + 0.5
        BX = YRES * GAY * scale + XRES / 2 + \
                random.randint(-XRES/4, XRES/4)
        BY = YRES * GAZ * scale + YRES / 2 + \
                random.randint(-YRES/7, YRES/7)
        VX = 0
        VY = 0
        P = 100
        COLR = random.randint(0, 255)
        COLG = random.randint(0, 255)
        COLB = random.randint(0, 255)

    # detect stopping
    if fast == 1 and (A < 0.1 or ((BX > (XRES + 200) \
                or BX < -200) and (BY > (YRES + 200) \
                or BY < -200)) or P <= 0):
        notfast = notfast + dt
        if notfast >= 0.12:
            fast = 0
            BX = 0
            BY = 0
            last_stroke = last_time

    if fast == 1:
        # accelerate the paint brush
        VX = VX - GAY * dt * 170
        VY = VY - GAZ * dt * 170
        BX = BX + VX * dt * 120
        BY = BY + VY * dt * 120

        # add splotches.... high velocity big
        # splotches far apart, low small close
        if P > 0:
            V = numpy.linalg.norm([VX, VY])
            S = S + V
            d = A * random.randint(3, 5) * 25 + V
            if S > d:
                S = S - d
                P = P - pow(A*4, 2) * math.pi
                pygame.draw.circle(screen, (COLR, COLG,
                        COLB), (int(BX), int(BY)), int(A*45))
                draw = 1

        # push updates to the screen
        if draw == 1:
            pygame.display.flip()

pygame.quit()
```

Remote (Client)



accelerometer sensor unit

```
import socket
import XLoBorg
import time

# network stufff
server = "modelb"
port = 5005

# setup for the accelerometer
XLoBorg.printFunction = XLoBorg.NoPrint
XLoBorg.Init()

# make the socket connection
sock = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
while True:
    message = '%+01.4f,%+01.4f,%+01.4f' \
            % XLoBorg.ReadAccelerometer()
    sock.sendto(message, (server, port))
    time.sleep(0.005)
```