Outline

Review

- Create a ball game
- Modding examples

Cool media art pieces

- Project presentations

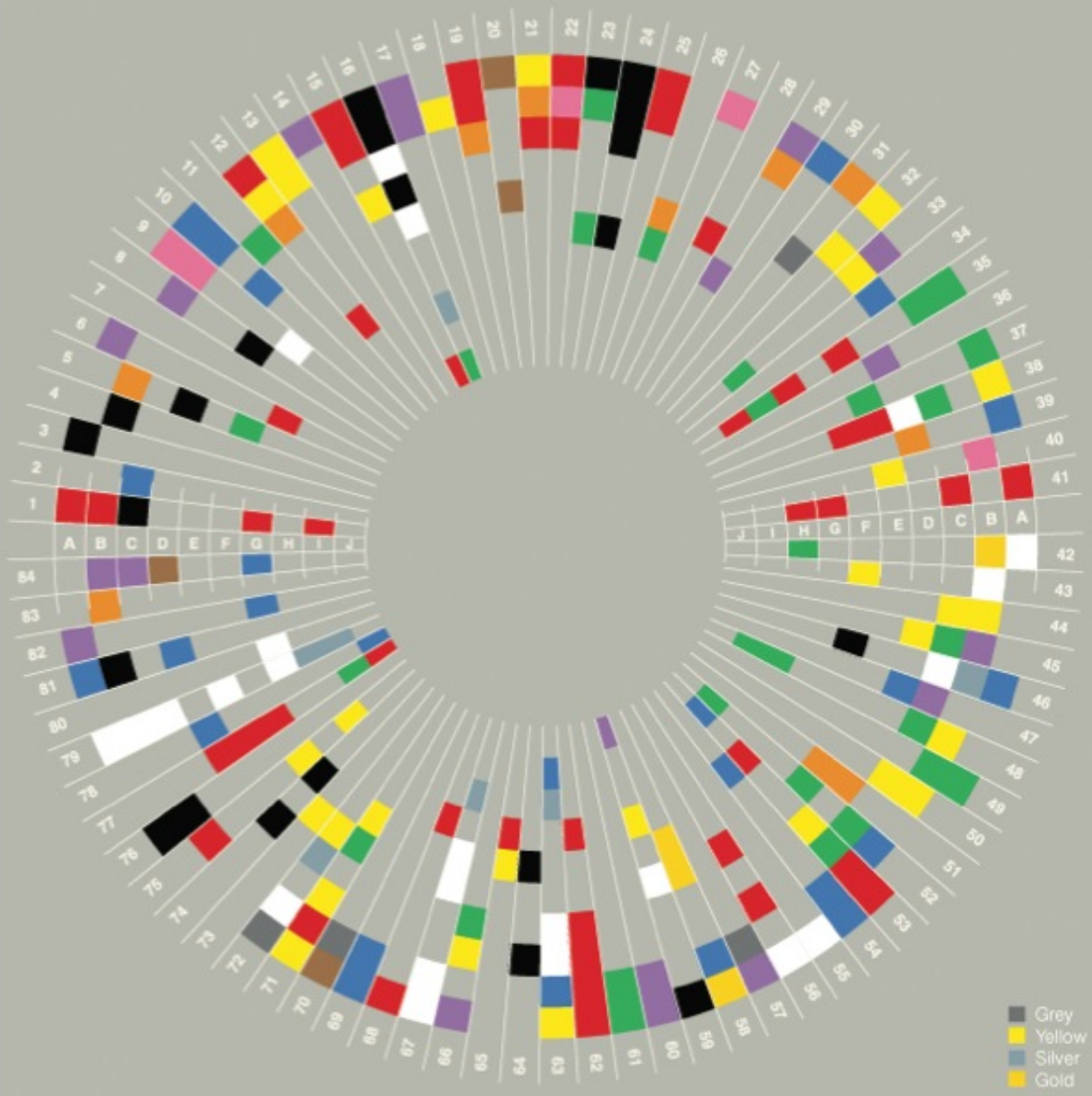
Prof. Angela Chang

Lecture 22 Modding again

Fall 2017 Nov 26

CODE, CULTURE, AND PRACTICE

Colours In Culture



A	Western / American
B	Japanese
C	Hindu
D	Native American
E	Chinese
F	Asian
G	Eastern European
H	Muslim
I	African
J	South American

1	Anger
2	Art / Creativity
3	Authority
4	Bad Luck
5	Balance
6	Beauty
7	Calm
8	Celebration
9	Children
10	Cold
11	Compassion
12	Courage
13	Cowardice
14	Cruelty
15	Danger
16	Death
17	Decadence
18	Deceit
19	Desire
20	Earthy
21	Energy
22	Erotic
23	Eternity
24	Evil
25	Excitement
26	Family
27	Femininity
28	Fertility
29	Flamboyance
30	Freedom
31	Friendly
32	Fun
33	God
34	Gods
35	Good Luck
36	Gratitude
37	Growth
38	Happiness
39	Healing
40	Healthy
41	Heat
42	Heaven
43	Holiness
44	Illness
45	Insight
46	Intelligence
47	Intuition
48	Religion
49	Jealousy
50	Joy
51	Learning
52	Life
53	Love
54	Loyalty
55	Luxury
56	Marriage
57	Modesty
58	Money
59	Mourning
60	Mystery
61	Nature
62	Passion
63	Peace
64	Penance
65	Power
66	Personal power
67	Purity
68	Radicalism
69	Rational
70	Reliable
71	Repels Evil
72	Respect
73	Royalty
74	Self-cultivation
75	Strength
76	Style
77	Success
78	Trouble
79	Truce
80	Trust
81	Unhappiness
82	Virtue
83	Warmth
84	Wisdom

David McCandless & AlwaysWithHonor.com // v1.0 // Apr 09 // InformationIsBeautiful.net

source: Pantone, ColorMatters & web sources

Sound when an event happens

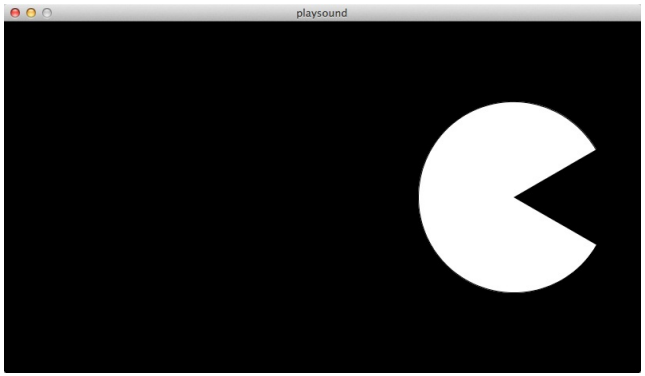
playsound

```
add_library('sound')
blip = None
radius = 120
x = 0
speed = 3.0
direction = 1
def setup():
    global blip, x
    size(800, 440)
    ellipseMode(RADIUS)
    blip = SoundFile(this, 'blip.wav')
    x = width/2 #start in the center
def draw():
    global x, direction
    background(0)
    x += speed * direction
    if x > width-radius or x < radius:
        direction = -direction #flip direction
        blip.play()
    if direction == 1:
        arc(x, 220, radius, radius, 0.52, 5.76) #Face right
    else:
        arc(x, 220, radius, radius, 3.67, 8.9) #face left
```

define the global sound variable

connect sound

play infrequently



playsound.pyde

Input a sound using AudioIn

soundInput

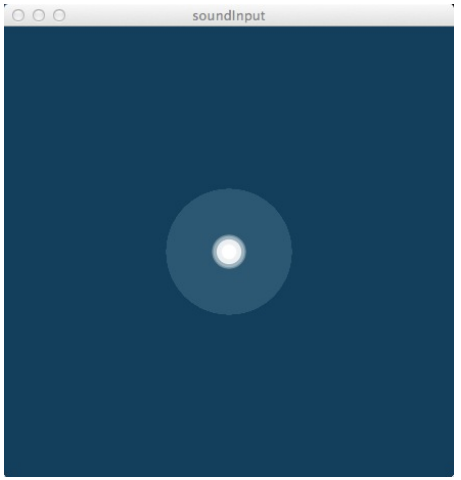
```
add_library('sound')
mic = None
amp = None
def setup():
    global mic, amp
    size(440,440)
    background(0)
    #Create an audio input and start it
    mic = AudioIn(this,0)
    mic.start()
    #Create a new amplitude analyzer and patch into input
    amp = Amplitude(this)
    amp.input(mic)

def draw():
    #draw a background that fades to black
    noStroke()
    fill(26, 76, 102, 10)
    rect(0,0, width, height)
    # Analyze() returns values between 0 and 1
    #so map() is used to convert values to larger numbers
    diameter = map(amp.analyze(),0,1, 10, width)
    # Draw circle based on the voume
    fill(255)
    ellipse(width/2, height/2, diameter, diameter)
```

define the global sound variable

open the mic

analyze the data from the mic

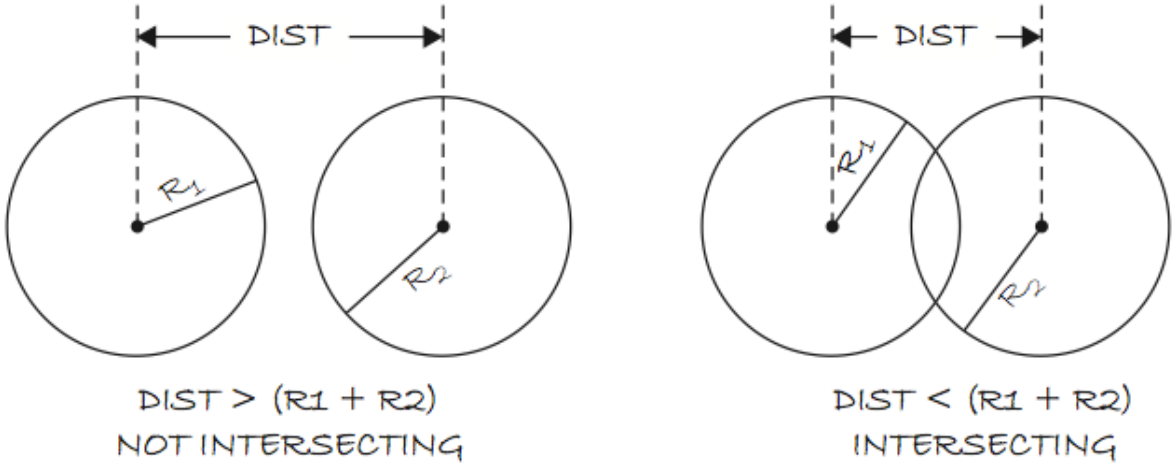
change diameter based on sound



soundinput.pyde

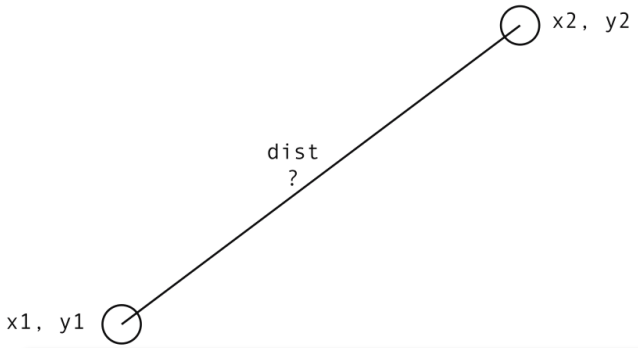
Detecting collision

If the sum of the radii is less than the distance between centers, then they are intersecting.



pseudocode:

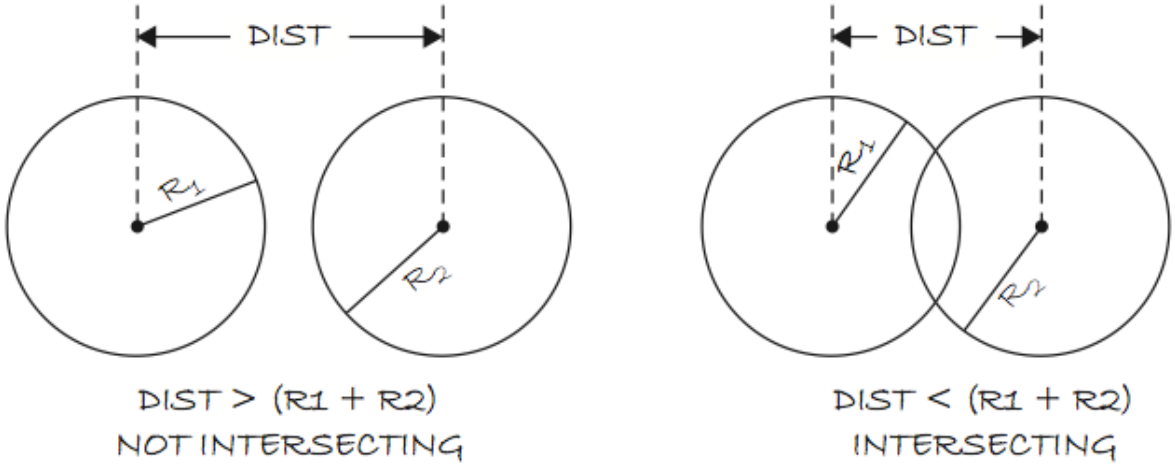
```
def intersecting():
    distance = dist(x1,y1,x2,y2)
    if (distance < r1,r2):
        return True
    else:
        return False
```



what is the distance between these two points?

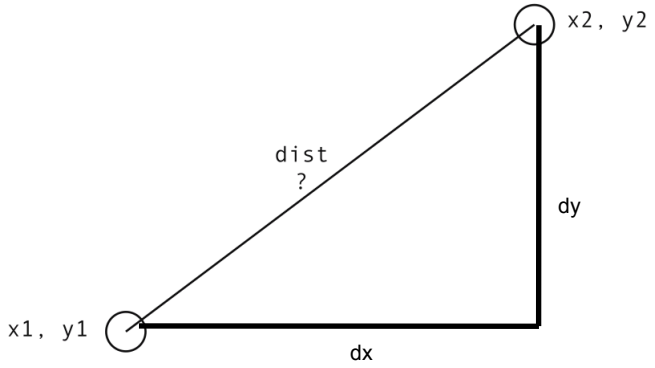
Detecting collision

If the sum of the radii is less than the distance between centers, then they are intersecting.



pseudocode:

```
def intersecting():
    distance = dist(x1,y1,x2,y2)
    if (distance < r1,r2):
        return True
    else:
        return False
```



Pythagorean theorem

$a^2+b^2 = c^2$

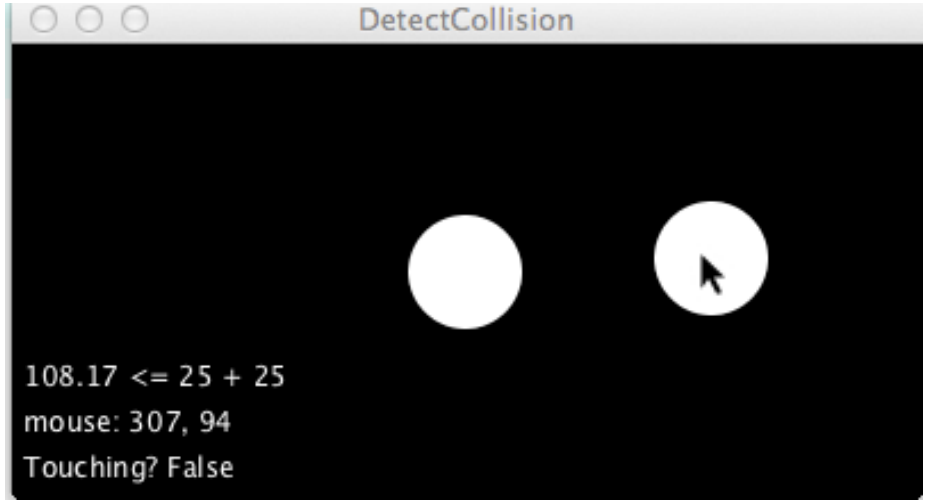
what is the distance between these two points?

turn it into a right triangle.

processing has a function for distance
https://processing.org/reference/dist_.html

Detect Collision Code

```
distance = 0.0
def setup():
    frameRate(30)
    global x1,y1,r1,r2
    size(400,400)
    noStroke()
    x1 = 45  #first ball at 45,100
    y1 = 100
    r1 = 25  #radius for two balls equal
    r2 = 25

def draw():
    background(0)
    global x1,y1,x2,y2,r1,r2
    ball1 = ellipse(x1,y1,r1*2,r1*2)
    ball2 = ellipse(mouseX,mouseY,r2*2,r2*2)
    string3 = 'Touching? '+str(intersect(x1,y1,mouseX,mouseY,r1,r2))
    string1 = str(round(distance,2)) + ' <= '+str(r1)+' + '+ str(r2)
    string2 = 'mouse: '+str(mouseX)+', '+ str(mouseY)
    text(string1,5, 150)
    text(string2,5, 170)
    text(string3,5, 190)

def intersect(p1x,p1y,p2x,p2y,ra1,ra2):
    global distance
    distance = dist(p1x,p1y,p2x,p2y)
    if (distance < ra1+ra2):
        return True
    else:
        return False
```



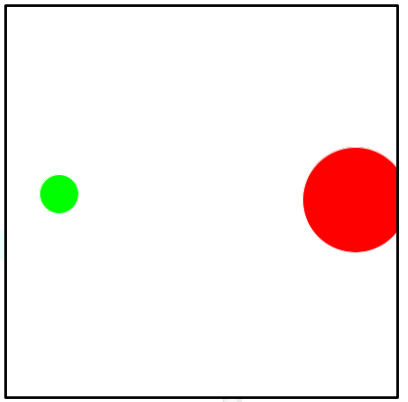
translate pseudocode:

```
def intersecting():
    distance = dist(x1,y1,x2,y2)
    if (distance < r1,r2):
        return True
    else:
        return False
```

DetectCollision.pyde

Simplify first

start by reducing the problem to just two balls colliding



Ball0_TwoBalls | Ball.py

```
from Ball import Ball

def setup():
    global ball1
    global ball2
    size(400,400)
    ball1 =  Ball(54,'#FF0000'
    ball2 =  Ball(20,'#00FF00'
    print ball1,ball2

def draw():
    global ball1,ball2
    background(255)
    ball1.move()
    ball2.move()
```

```
class Ball():

    def __init__(self,tempR,newColor):
        global r,x,y,xspeed,yspeed,myColor
        self.r = tempR
        self.x = random(width)
        self.y = random(height)
        self.xspeed = random(-5,5)
        self.yspeed = random(-5,5)
        self.myColor=newColor
        print 'ball ',self.x,self.y,self.xspeed,self.yspeed, self.myColor

    def move(self):
        #global x,y,xspeed,yspeed
        self.x += self.xspeed
        self.y += self.yspeed

        if (self.x>width or self.x<0):
            self.xspeed *= -1
        if (self.y>height or self.y<0):
            self.yspeed *= -1
        stroke(255)
        fill(self.myColor)
        ellipse(self.x,self.y,self.r*2,self.r*2)
```

Ball0_TwoBalls.pyde

Detect 2 balls intersecting

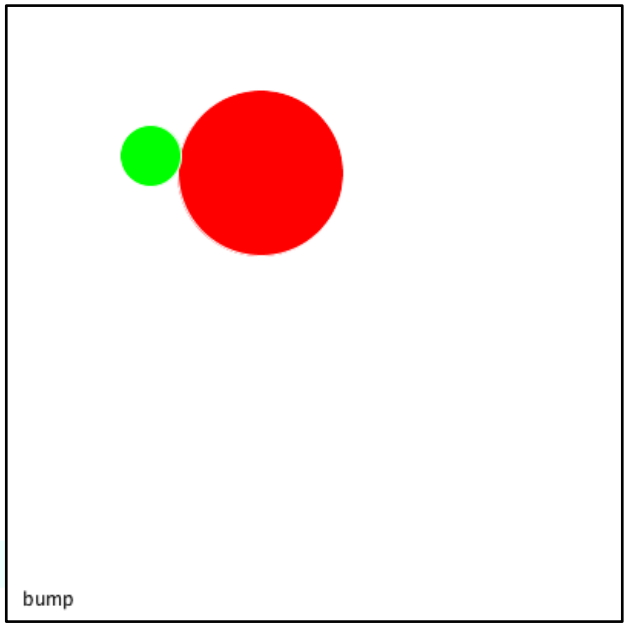
Where to put the code?

move the detection to the Ball class

```
def intersect(self, other):
    distance = dist(self.x, self.y, other.x, other.y)
    if (distance < self.r + other.r):
        return True
    else:
        return False
```

call it in draw() so ball is always checking

```
def draw():
    global ball1,ball2
    background(255)
    ball1.move()
    ball2.move()
    fill(0)
    if ball1.intersect(ball2):
        text('bump',10,390)
```



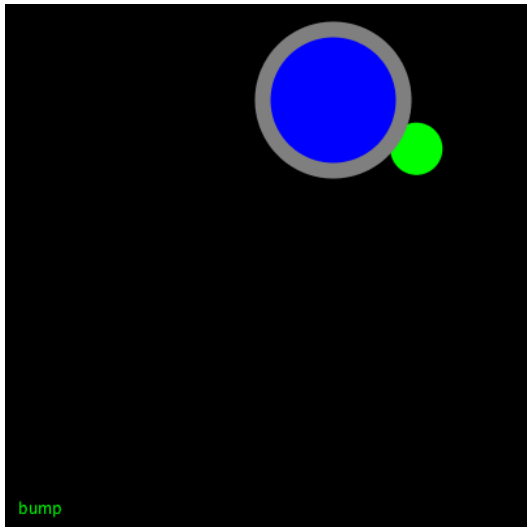
Ball0_TwoBallCollision.pyde

On collisions, do stuff

```
def move(self):
    #global x,y,xspeed,yspeed
    self.x += self.xspeed
    self.y += self.yspeed
    if (self.x>width or self.x<0):
        self.xspeed *= -1
    if (self.y>height or self.y<0):
        self.yspeed *= -1
    stroke(255)
    noStroke()
    fill(self.myColor)
    ellipse(self.x,self.y,self.r*2,self.r*2)

def highlight(self):
    strokeWeight(12)
    stroke(127)
    fill('#0000FF')
    ellipse(self.x, self.y, self.r * 2, self.r * 2)
```

remove outline, reset color

draw outline when hit color set to blue

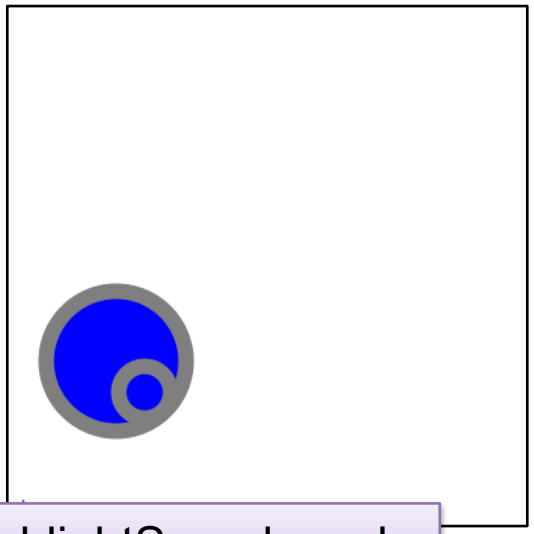


BallHighlightIntersection.pyde

```
def setup():
    blip = SoundFile(this,'blip.wav')
def draw():
    if (ball1.intersect(ball2) or ball2.intersect(ball1)):
        blip.play()
        ball1.highlight()
        ball2.highlight()
        text('bump',10,390)
    else:
        text('free',10,390)
    ball1.display()
    ball2.display()
```

set the variable to file

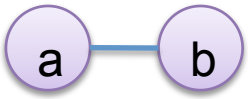
play the sound when intersecting



BallHighlightSounds.pyde

MultiBall Collisions

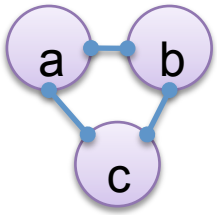
Detecting many-many collisions



1 collision

The number of collisions given *n* balls.

$$\frac{n\,(n-1)}{2}$$

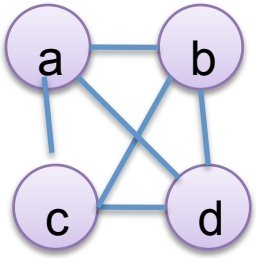


3 collisions

Checking each one against every one

is inefficient (order of *n*n)*:

a-a,a-b,a-c, b-a,b-b,b-c, c-a,c-b,c-c



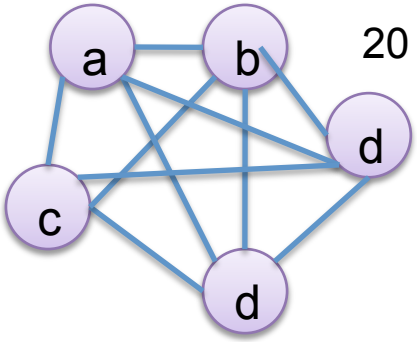
6 collisions

Don't check yourself

consider that for 3 balls, we check 6 times

order of *n*n-1:*

a-b,a-c, b-a, b-c, c-a,c-b



20 collisions

```
for i in range(len(ballList)):
    for j in range(len(ballList)):
        if ballList[i]!=ballList[j]:
            if (ballList[i].intersect(ballList[j])):
                text('bump ',10,460)
                print('bump '+str(i)+str(j))
```

no checking against itself

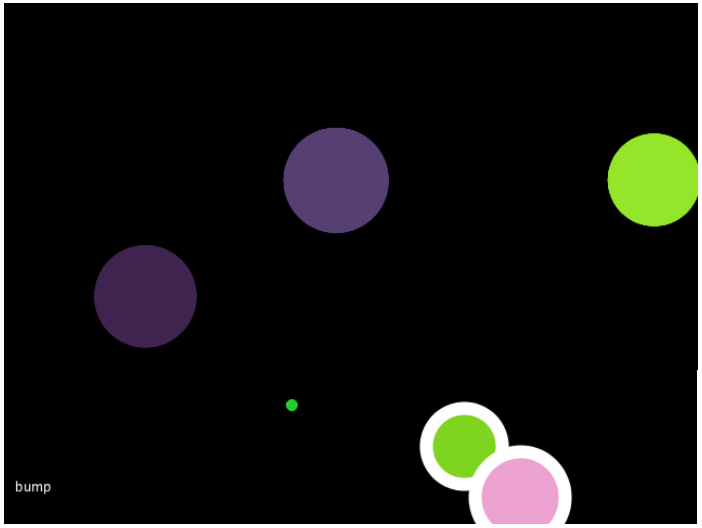
Ball1_Collisions.pyde

MultiBall Collisions

going back to the detecting many-many collisions

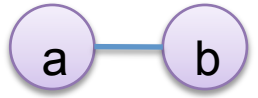
reduce duplicate checks by making list smaller

```
for i in range(len(ballList)-1):
    ballA = ballList[i]
    for j in range(i+1,len(ballList)):
        ballB = ballList[j]
        if (ballA.intersect(ballB)):
            fill(255)
            text('bump ',10,450)
            print('bump '+str(i)+str(j))
```

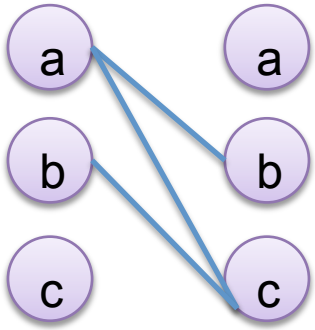


poor man's bounce

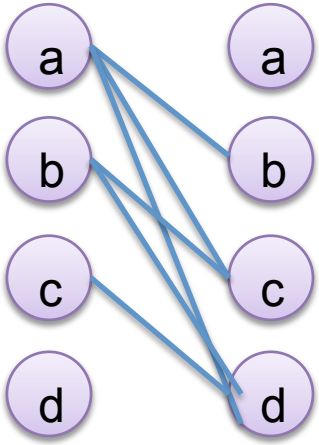
```
def reverse(self):
    self.ballVX *= -1
    self.ballVY *= -1
```



1 collision



3 collisions



6 collisions

Ball2_Collisions.pyde

PopBubbles

check each ball to see if its been clicked

```
def mouseClicked():
    clickX = mouseX
    clickY = mouseY
    for i in range(numBalls):
        if inBall(clickX,clickY,ballList[i]):
            ballList[i].disappear()
            print 'bye '+str(i)
```

clicked?

```
def inBall(tempX,tempY,tempBall):
    distance = dist(tempX,tempY,tempBall.ballX,tempBall.ballY)
    if distance < tempBall.ballDiam/2 :
        return = True #clicked the Ball
    else:
        return = False  #missed
```

add a disappear() method to Ball objects

```
def disappear(self):
    global visible
    self.ballX = -3000
    self.ballY = -3000
    self.ballVX = 0.0
    self.ballVY = 0.0
    self.Diam = 5
    self.ballCol = 255
    self.visible = False
```

embellishments:

Better movement:

- add mass to bounce
- add friction

Sound

Check for game over

Keep score

Ball3_CollisionPop

Modding examples

Ball0_Colors | Ball.py

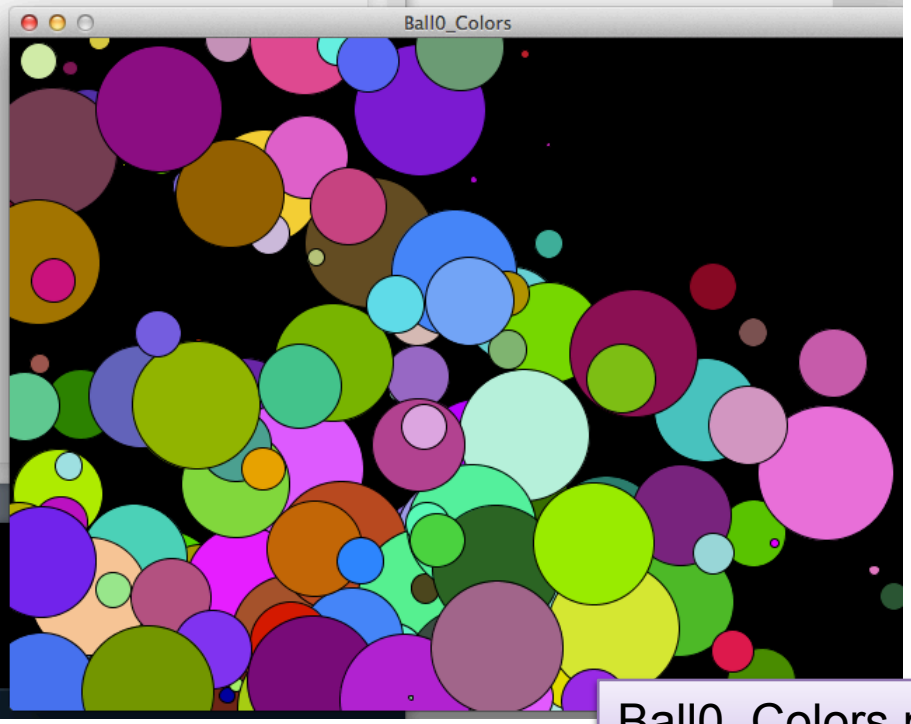
```
 #from the Ball.py file, import the Ball class
from Ball import Ball
numBalls = 200
ballList =[]
for i in range(numBalls):
    ballList.append ( Ball(random(10),random(10),\
                           random(10),random(10),\
                           random(100), random(255),\
                           random(255), random(255) ) )

def setup():
    background(0)
    size(640,480)

def draw():
    background(0)
    for i in range(len(ballList)):
        ballList[i].moveBall()
```

Console

see chu's submisision

How might you give different ways to play:
change numbers, strings
--speed, timing, colors
add variables
add functions

you can get really far!



Ball0_Colors.pyde

Homework

1

Create a game in processing, adding your own graphics

@Motivashn

2

Decide what you want to do for a final project and sign up for a group on canvas. In the class submission, describe what your project is and keep working on your projects together.

Summary of today

- Technical practice, reading code together
  - class participation: upload your ball game mods
- HW for this Wednesday:
  - create groups and descriptions on canvas for final projects http://bit.ly/CodeGroup2017
- HW for Monday (heads up):
  - Explore the Processing Examples under
  - Examples -> Processing
  - Modify and make a video using saveFrame() and one of the video examples and upload to canvas