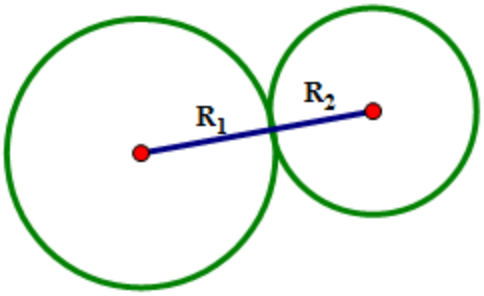
Collision Detection

Collision detection is required when one object meets another object in computer programming.

The collision can involve all kinds of shapes. The following is a discussion on a couple of the most common types.



Circle vs Circle

In order to check for a collision between two circles, the distance between the centers needs to be less than the sum of the radii, as shown in the diagram.

The following function tests for this:

```
def checkCollision(circle1, circle2, radius1, radius2):
    # check the distance between the center of the circles
    # Grade 10 math, distance between two points
    distance = (circle1[0] - circle2[0])**2 + (circle1[1] - circle2[1])**2
    distance = math.sqrt(distance)
    if distance < radius1 + radius2:  #sum of the radii's
        return True
    return False
```

Example #1

The following program uses circle collision to test when two circles meet.

```
from pygame import *
import random
import math
init()
size = width, height = 800, 600
screen = display.set_mode(size)
button = 0
```

```
BLACK = (0, 0, 0)
GREEN = (0, 255, 0)
BLUE = (0, 0, 255)

def drawScene(screen, mouseCircle, circle2):
    draw.rect(screen, BLACK, (0,0, width, height))
    draw.circle(screen, GREEN, mouseCircle, 10)
    draw.circle(screen, BLUE, circle2, 10)
    display.flip()

def initCircle():
    circle = [random.randint(0, width), 0]
    return circle

# take two circles, each a list of x and y, two radii
def checkCollision(circle1, circle2, radius1, radius2):
    # check the distance between the center of the circles
    # Grade 10 math, distance between two points
    distance = (circle1[0] - circle2[0])**2 + (circle1[1] - circle2[1])**2
    distance = math.sqrt(distance)
    if distance < radius1 + radius2:  #sum of the radii's
        return True

running = True
myClock = time.Clock()
mouse = [0, 0]
targetCircle = initCircle()
radius1 = radius2 = 10
score = 0
# Game Loop
while running:
    for evnt in event.get():            # checks all events that happen
        if evnt.type == QUIT:
            running = False
        if evnt.type == MOUSEMOTION:
            mx,my = evnt.pos
            mouseCircle = [mx, my]
```

```
        if checkCollision(mouseCircle, targetCircle, radius1, radius2) ==
True:
            targetCircle = initCircle() #generate new circle
    # move the villain target
    targetCircle[1] += 1
    drawScene(screen, mouseCircle, targetCircle)
    myClock.tick(60)                # waits long enough to have 60 fps

quit()
```

Rectangle vs Rectangle

Pygame has created an object just for this. It is called Rect.

Rect is similar to a List or a String but has some special features that we can use when creating a graphics program.

Rect(x, y, width, height)

Methods:

collidepoint(a, b) - tests if the point (a, b) is within the rectangle
- returns True if it is
- can also test a two-element list

colliderect(testRect) - tests if the two rectangles overlap
- Returns True if they overlap
- Returns False if they do not

collidelist(listRect) - listRect would be a list of rectangles
- tests if any rectangles in the list collide with the original rectangle
- very handy when there are many enemies, etc...

Example #2

The following program uses rects and colliderects similarly to example #1.

```
from pygame import *
import random
import math
init()
size = width, height = 800, 600
screen = display.set_mode(size)
button = 0

BLACK = (0, 0, 0)
GREEN = (0, 255, 0)
BLUE = (0, 0, 255)

def drawScene(screen, rectangle1, rectangle2):
  draw.rect(screen, BLACK, (0,0, width, height))
  draw.rect(screen, GREEN, rectangle1)
  draw.rect(screen, BLUE, rectangle2)
  display.flip()

def initRect():
  rectangle = Rect(random.randint(0, width), 0, random.randint(20, 100),
random.randint(20, 100))
  return rectangle

running = True
myClock = time.Clock()
mouseRect = Rect(0, 0, 1, 1)
targetRect = initRect()
# Game Loop
while running:
  for evnt in event.get():            # checks all events that happen
    if evnt.type == QUIT:
      running = False
    if evnt.type == MOUSEMOTION:
      mx,my = evnt.pos
      mouseRect = Rect(mx, my, 40, 40)
      if mouseRect.colliderect(targetRect) == True:
```

```
        targetRect = initRect() #generate new circle
    # move the villain target
    targetRect[1] += 1
    drawScene(screen, mouseRect, targetRect)
    myClock.tick(60)                    # waits long enough to have 60 fps

quit()
```

Example #3

The following program uses Rect and colliderect to determine when the user controlled rectangle intersects a randomly generated rectangle.

```
from pygame import *
import random

init()

RED = (255, 0, 0)
BLACK = (0,0,0)
BLUE = (0, 255, 0)
info = display.Info()
width = 500
height = 300

SIZE = (width, height)
screen = display.set_mode(SIZE)#,FULLSCREEN)

#some game states
KEY_RIGHT = False
KEY_LEFT = False
KEY_UP = False
KEY_DOWN = False

def drawScreen(player, food):

    draw.rect(screen, BLACK, (0, 0, width, height))
```

```
  draw.rect(screen, RED, player)
  draw.rect(screen, BLUE, food)
  display.flip()

def initFood():
  rect = Rect(random.randint(0, width), random.randint(0, height), 20, 20)
  return rect

def checkForCollision(player, food):
  if player.colliderect(food):
    return True
  return False

myClock = time.Clock()
running = True
x = width/2
playerRect = Rect(0, 0, 50, 50)
foodRect = initFood()
while running:
  for evnt in event.get():
    if evnt.type == QUIT:
      running = False

    if evnt.type == KEYDOWN:
      if evnt.key == K_LEFT:
        KEY_LEFT = True
      if evnt.key == K_RIGHT:
        KEY_RIGHT = True
      if evnt.key == K_UP:
        KEY_UP = True
      if evnt.key == K_DOWN:
        KEY_DOWN = True

    if evnt.type == KEYUP:
      if evnt.key == K_LEFT:
        KEY_LEFT = False
      if evnt.key == K_RIGHT:
        KEY_RIGHT = False
```

```
      if evnt.key == K_UP:
        KEY_UP = False
      if evnt.key == K_DOWN:
        KEY_DOWN = False

  if KEY_LEFT == True:
    playerRect[0] -= 1
  if KEY_RIGHT == True:
    playerRect[0] += 1
  if KEY_UP == True:
    playerRect[1] -= 1
  if KEY_DOWN == True:
    playerRect[1] += 1

  if checkForCollision(playerRect, foodRect) == True:
    foodRect = initFood()

  drawScreen(playerRect, foodRect)
  display.flip()
  myClock.tick(60)

quit()
```