Information Science
5: Making More Images with Arrays

Naonori Kakimura
垣村尚徳
kakimura@global.c.u-tokyo.ac.jp

PEAK
Programs in English at Komaba

Remarks on Previous Exercises

- Before submission,
  - check whether your program works
    - By executing the program with specified parameters
      - Need careful writing
  - Consider why it works or why it does not
    - Simpler logic with no redundancy is better

Remarks on Homework

- Please insert some space for visibility
  - easier to understand nesting structures

```
def sierpinski_loop(n)
c = make2d(n+1,n+1)
for i in 0..n
c[i][0] = 1
for j in 1..(i-1)
c[i][j]=(c[i-1][j-1]+c[i-1][j])%2
end
c[i][i] = 0
end
for i in 0..n
for j in 0..n
c[i][j]=1-c[i][j]
end
end
c
end
```

```
def sierpinski_loop(n)
  c = make2d(n+1,n+1)
  for i in 0..n
    c[i][0] = 1
    for j in 1..(i-1)
      c[i][j]=(c[i-1][j-1]+c[i-1][j])%2
    end
    c[i][i] = 0
  end
  for i in 0..n
    for j in 0..n
      c[i][j]=1-c[i][j]
    end
  end
  c
end
```

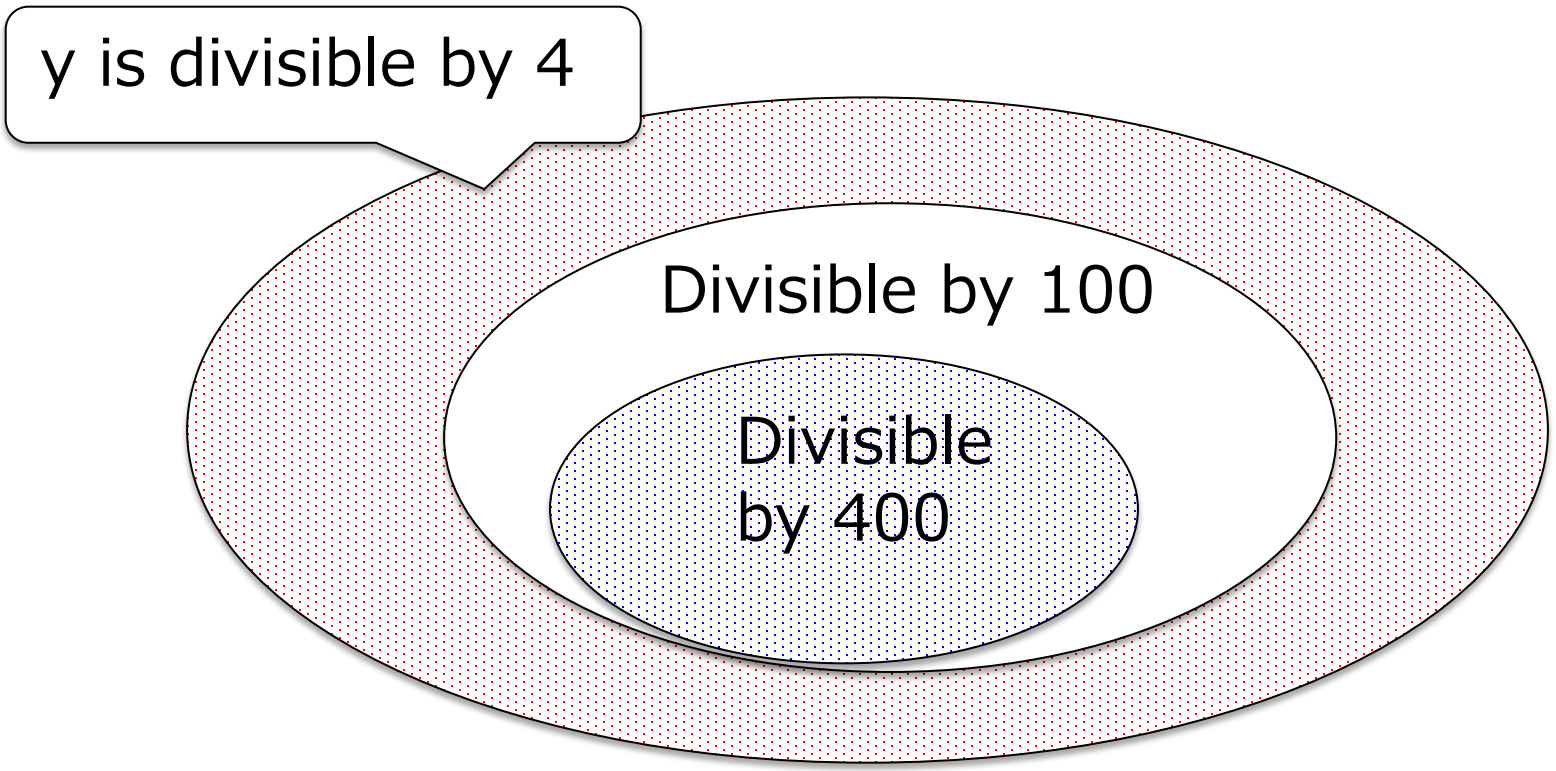
Can avoid error (lack of “end”)

Remarks on installing isrb at home

- Install isrb2
  - Not isrb

- My handouts How2Install.pdf are old
  - Visit http://prg.is.titech.ac.jp/i2cs.rb/isrb2/
    - http://prg.is.titech.ac.jp/i2cs.rb/isrb2/isrb2-20121001.zip
    - http://prg.is.titech.ac.jp/i2cs.rb/isrb2/isrb2-20121001.tar.gz

Review: leap_year(y)

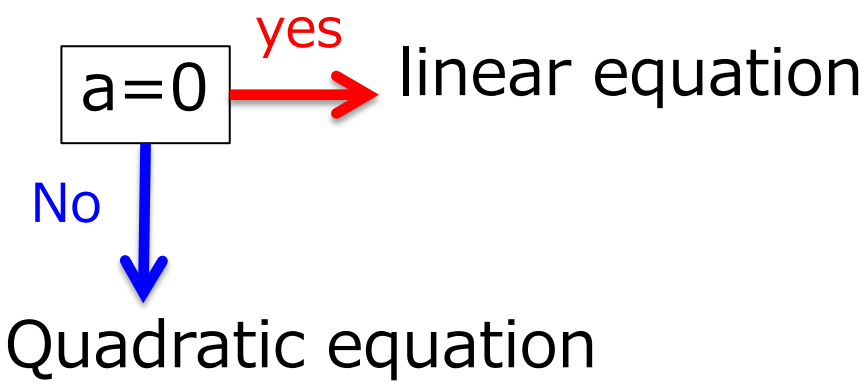



```
def leap_year(y)
      (y%4==0 && y%100!=0) || y%400==0
end
```

Many ways to describe the condition using “if”

Rev. Computing Number of Real Solutions

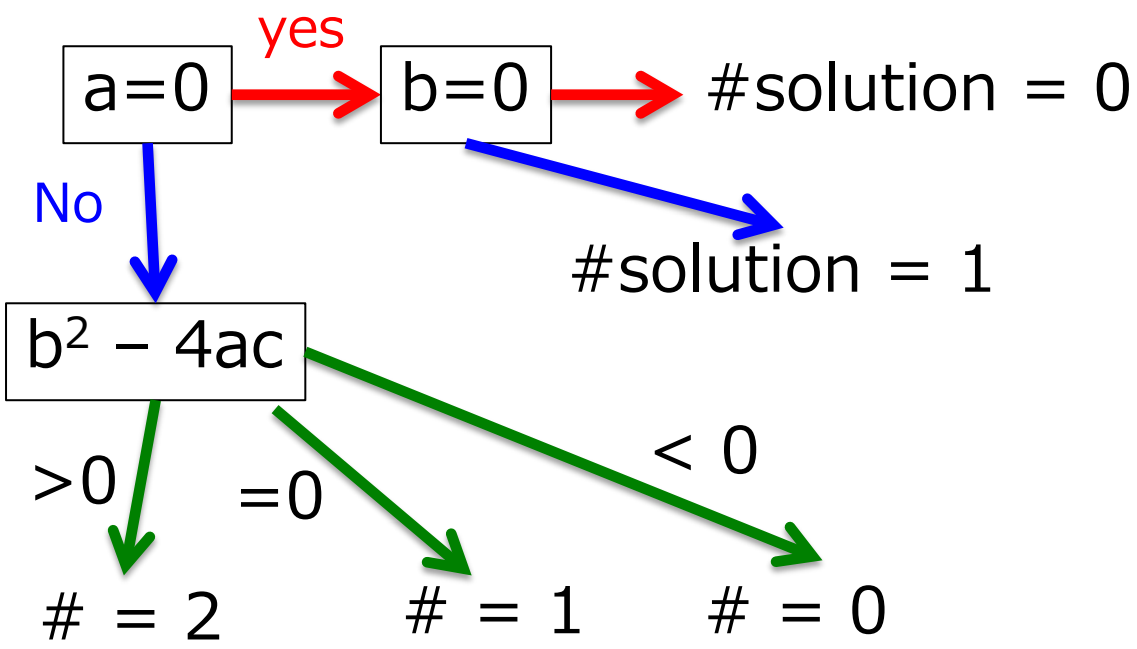
- $ax^2 + bx + c = 0$




```
def solutions(a,b,c)
  if a == 0
    (when it is linear)

  else
    (when it is quadratic)



  end
end
```

Computing Number of Real Solutions

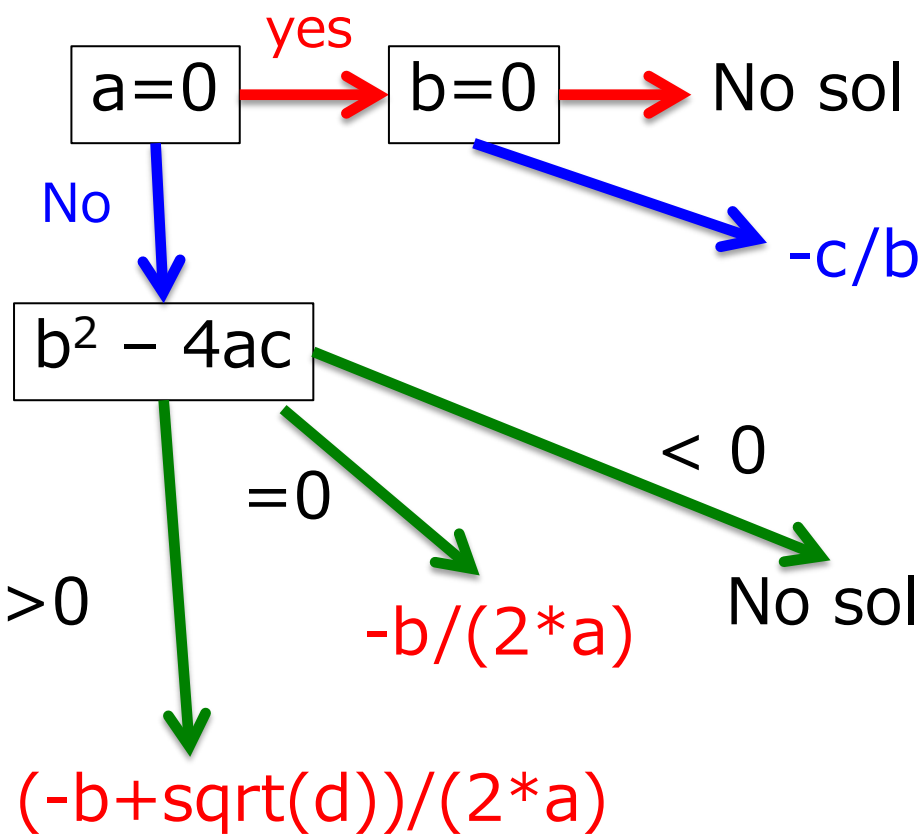
- $ax^2 + bx + c = 0$




```
def solutions(a,b,c)
  if a == 0
    if b==0
      0
    else
      1
    end
  else
    d = b**2 - 4*a*c
    if d>0
      2
    elsif d==0
      1
    else
      0
    end
  end
end
```

Computing One of Real Solutions

- $ax^2 + bx + c = 0$



Just replace w./ a solution

```
def solve1(a,b,c)
  if a == 0
    if b==0
      nil
    else
      -1.0*c/b
    end
  else
    d = b**2 - 4*a*c
    if d>0
      (-b+sqrt(d))/(2.0*a)
    elsif d==0
      -b/(2.0*a)
    else
      nil
    end
  end
end
```

Median: 6 Possibilities of Ordering

- y > x > z → x
- z > x > y → x
- x > y > z → y
- z > y > x → y
- x > z > y → z
- y > z > x → z

```
def median(x,y,z)
  if (y>x && x>z) || (z>x && x>y)
     x
  elsif (x>y && y>z) || (z>y && y>x)
     y
  elsif (x>z && z>y) || (y>z && z>x)
     z
  end
end
```

Today's Contents

- Making more images
  - Review of "gradation" and "Sphere"
  - Make a rectangle
  - Random points
  - Make a color image
  - Animation
- Exercises

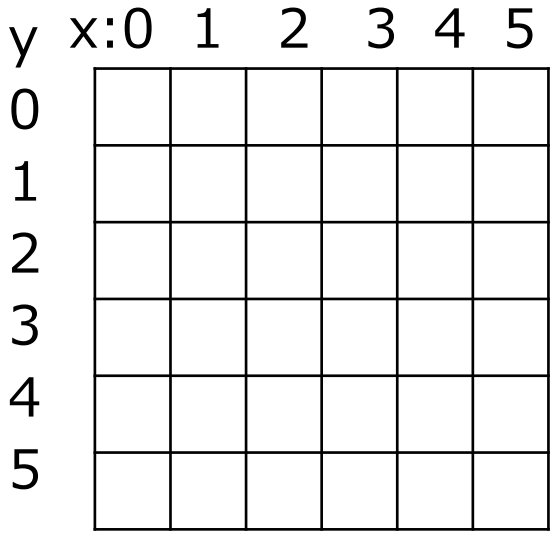
Review: Gradation


```
def gradation(s)
    image = make2d(s, s)
    for y in 0..(s-1)
        for x in 0..(s-1)
            image[y][x] = (2.0*s – x - y)/(2*s)
        end
    end
    image
end
```

Make a 2-dim array
all of whose entries are 0

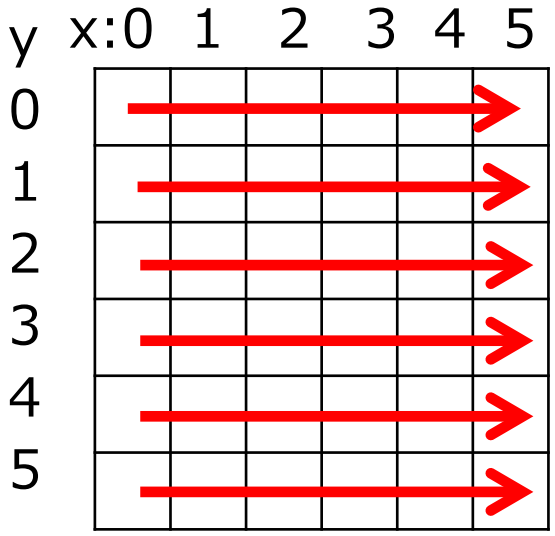
For each entry,
determine the brightness
according to some rule

```
def gradation(s)
    image = make2d(s, s)
    for y in 0..(s-1)
        for x in 0..(s-1)
            image[y][x] = (2.0*s – x - y)/(2*s)
        end
    end
    image
end
```



Create a 2-dim. array with s x s
(= a canvas with s x s)

```
def gradation(s)
    image = make2d(s, s)
    for y in 0..(s-1)
        for x in 0..(s-1)
            image[y][x] = (2.0*s – x - y)/(2*s)
        end
    end
    image
end
```



x and y are changing in the order of →
y runs from 0 to (s-1):
    for each y, x runs from 0 to (s-1)

For each pair (x,y)
  determine the brightness based on
  some function

```
def gradation(s)
    image = make2d(s, s)
    for y in 0..(s-1)
        for x in 0..(s-1)
            image[y][x] = (2.0*s – x - y)/(2*s)
        end
    end
    image
end
```

```
image[0][0]=1.0
image[0][1]=0.8333333333333334
image[0][2]=0.6666666666666666
image[1][0]=0.8333333333333334
image[1][1]=0.6666666666666666
image[1][2]=0.5
image[2][0]=0.6666666666666666
image[2][1]=0.5
image[2][2]=0.3333333333333333
```

when y=0

Change x from 0 to 2

when y=1

Change x from 0 to 2

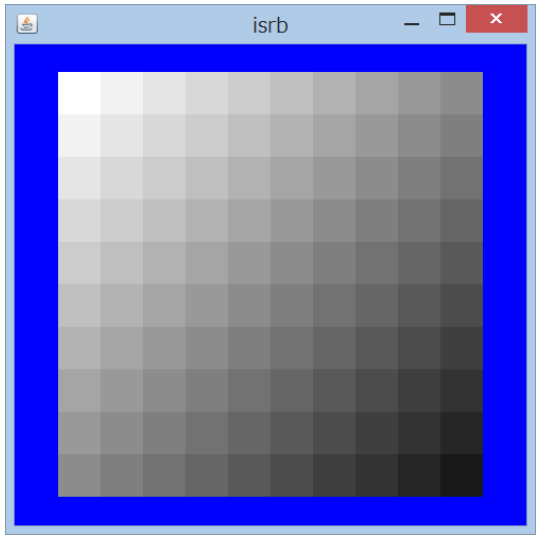
when y=2

Change x from 0 to 2

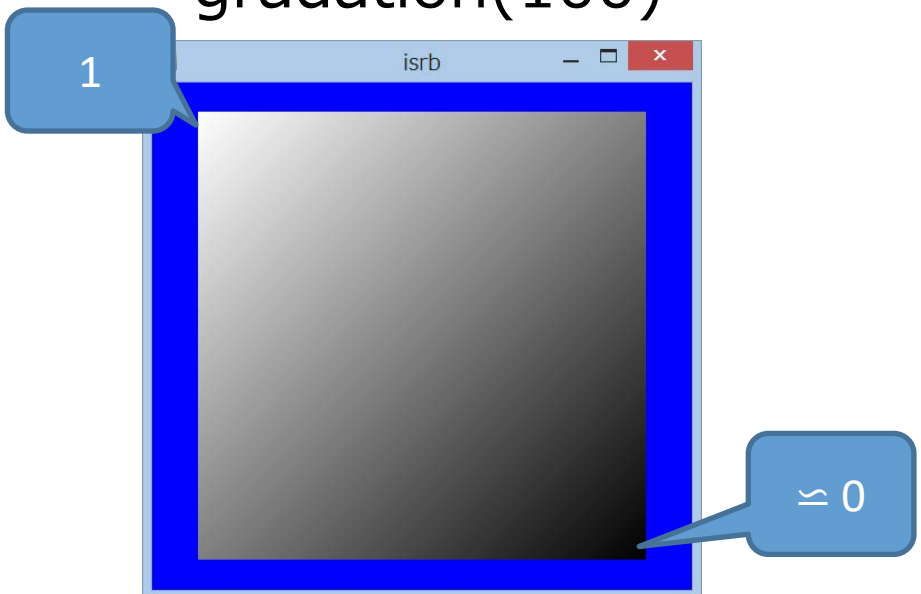
…

```
def gradation(s)
    image = make2d(s, s)
    for y in 0..(s-1)
        for x in 0..(s-1)
            image[y][x] = (2.0*s – x - y)/(2*s)
        end
    end
    image
end
```

gradation(10)



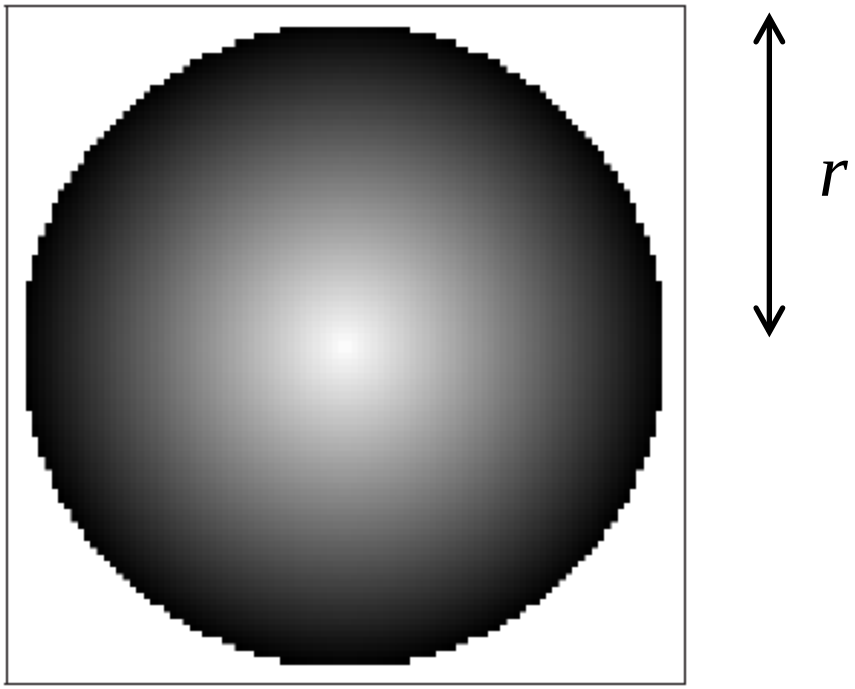
gradation(100)



Last Week's Exercise

Making an Image by using "Repetition"

radius is r

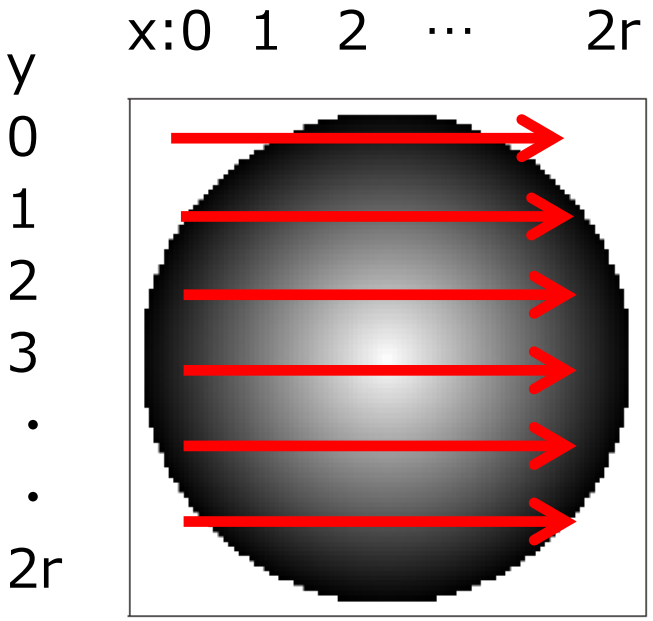


A point outside the circle(distance > r) is white

A point inside the circle becomes darker
if the distance is larger

Last Week's Exercise

Framework is same as gradation(s)



r=radius of the circle
The image size =2r+1

```
def sphere(r)
    image = make2d(2*r+1, 2*r+1)
    for y in 0..(2*r)
        for x in 0..(2*r)
            image[y][x] = XXX
        end
    end
    image
end
```

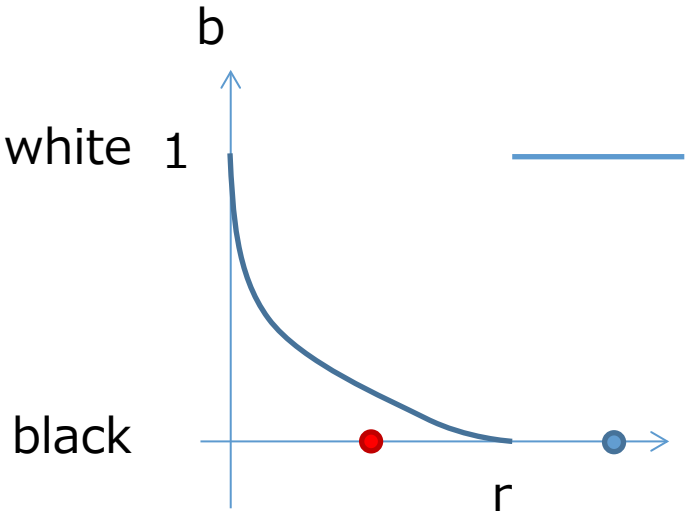
We change this function to make a sphere

Last Week's Exercises

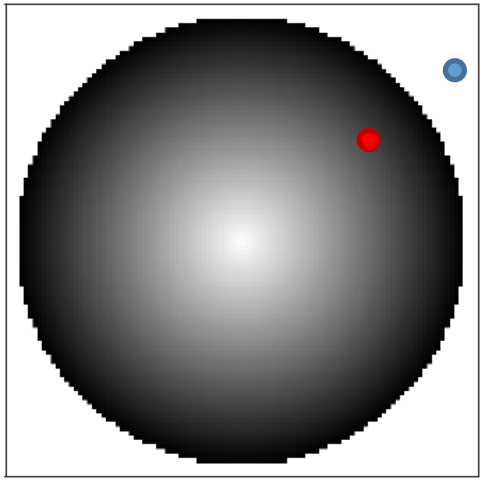
- Define the function b(r,x,y) that computes b(x,y) when r is specified.

$$b(x,y) = \begin{cases} \frac{r - d(x,y)}{r} & (d(x,y) \le r) \\ 1 & (d(x,y) > r) \end{cases}$$

Distance of (x,y) from the center



Distance from the center



(x,y)

Drawing a Sphere: Partial Program is Downloadable

Using b(r,x,y), we can define a function to draw a sphere

```
def sphere(r)
    image = make2d(2*r+1, 2*r+1)
    for y in 0..(2*r)
        for x in 0..(2*r)
            image[y][x] = b(r,x,y)
        end
    end
    image
end
```

Make a 2-dim array
all of whose entries are 0

For each entry,
determine the brightness
by the function b

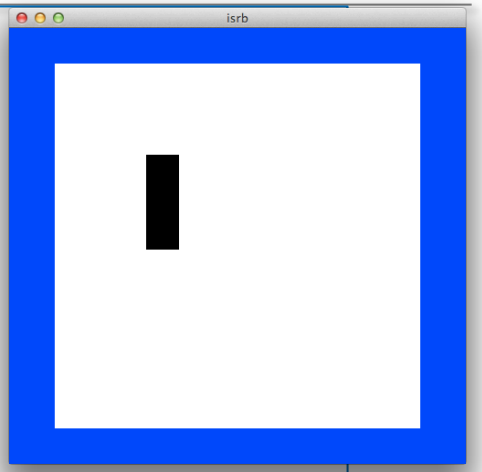
sphere.rb

Today's Contents

- Making more images
  - Review of "gradation" and "Sphere"
  - Make a rectangle
  - Random points
  - Make a color image
  - Animation
- Exercises

Drawing a Black Box: Downloadable

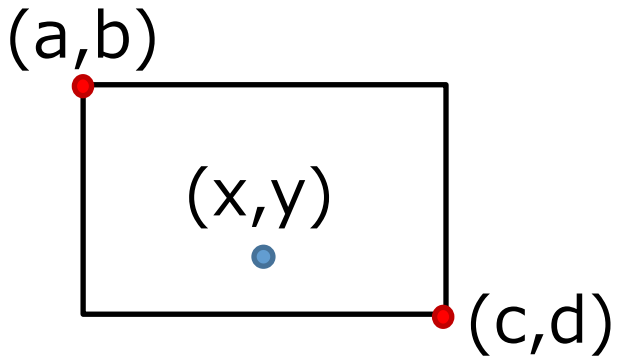

```
def BlackBox(s)
  image = make2d(s,s)
  for x in 0..(s-1)
     for y in 0..(s-1)
        image[y][x]=1 # paint white
       #If (x,y) is within some box, then paint black
        if inBox(x,y,s/4,s/4,s/3,s/2)
           image[y][x]=0
        end
     end
   end
   image
end
```




```
def inBox(x,y,a,b,c,d)
   a<=x && x<=c && b<=y && y<=d
end
```

Testing whether a point is contained or not

- inBox(x,y,a,b,c,d)



Returns true if (x,y) is in the box, and false otherwise

```
def inBox(x,y,a,b,c,d)
   a<=x && x<=c && b<=y && y<=d
end
```

Drawing a Black Box: Downloadable


```
def BlackBox(s)
  image = make2d_color(s,s)
  for x in 0..(s-1)
     for y in 0..(s-1)
        image[y][x]=1 # paint white
       #If (x,y) is within some box, then paint black
        if inBox(x,y,s/4,s/4,s/3,s/2)
           image[y][x]=0
        end
     end
   end
   image
end
```

image[0][0]… If inBox is true, then it is 0
image[0][1]… If inBox is true, then it is 0
image[0][2] …If inBox is true, then it is 0
image[1][0]… If inBox is true, then it is 0
image[1][1]… If inBox is true, then it is 0
image[1][2] …If inBox is true, then it is 0
image[2][0]… If inBox is true, then it is 0
image[2][1]… If inBox is true, then it is 0
image[2][2] …If inBox is true, then it is 0

Today's Contents

- Making more images
  - Review of "gradation" and "Sphere"
  - Make a rectangle
  - Random points
  - Make a color image
  - Animation
- Exercises

Random Number

- rand():
  - return a random real number btw 0 & 1
- rand(a)
  - a: positive integer
  - return a random integer between 0 & (a-1)

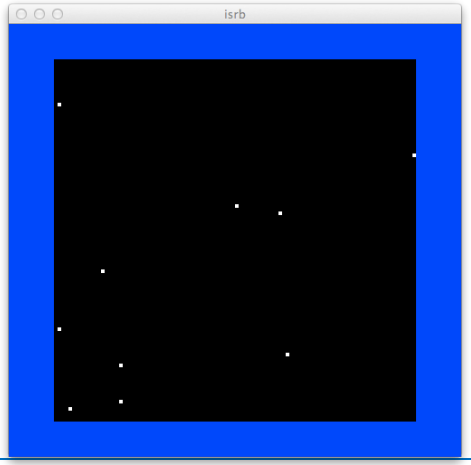
```
irb(main):012:0> rand()
0.8752890505837069
irb(main):013:0> rand(3)
1
irb(main):014:0> rand(3)
2
irb(main):015:0> rand(3)
0
```

Ex2: Random-Point Generation


```
def randomWhitePoints(s, n)
    image = make2d(s, s)
    for i in 0..(n-1)
      y = rand(s)
      x = rand(s)
      image[y][x] = 1
    end
    image
end
```

Repeat n times:
take a random point, and
paint white

s=100, n=10



Today's Contents

- Making more images
  - Review of "gradation" and "Sphere"
  - Make a rectangle
  - Random points
  - Make a color image
  - Animation
- Exercises

If You Want to Have a Color Image

- make2d_color(h,w) is available
  - Instead of make2d(height, width)
  - to make a 3-dim array for a color image

```
def make2d_color(height, width)
  a = Array.new(height)
  for i in 0..(height-1)
    a[i] = Array.new(width)
    for j in 0..(width-1)
      a[i][j] = make1d(3)
    end
  end
  a
end
```

Each entry is an array with size 3

Similar Example: Color Gradation

Each point (x,y) has three values (RGB)

```
def cgradation(s)
    image = make2d_color(s, s)
    for y in 0..(s-1)
      for x in 0..(s-1)
        image[y][x][0] = (2.0*s-x-y)/(2*s)
        image[y][x][1] = (1.0*s+x-y)/(2*s)
        image[y][x][2] = (1.0*s-x-y)/(2*s)
      end
    end
    image
end
```

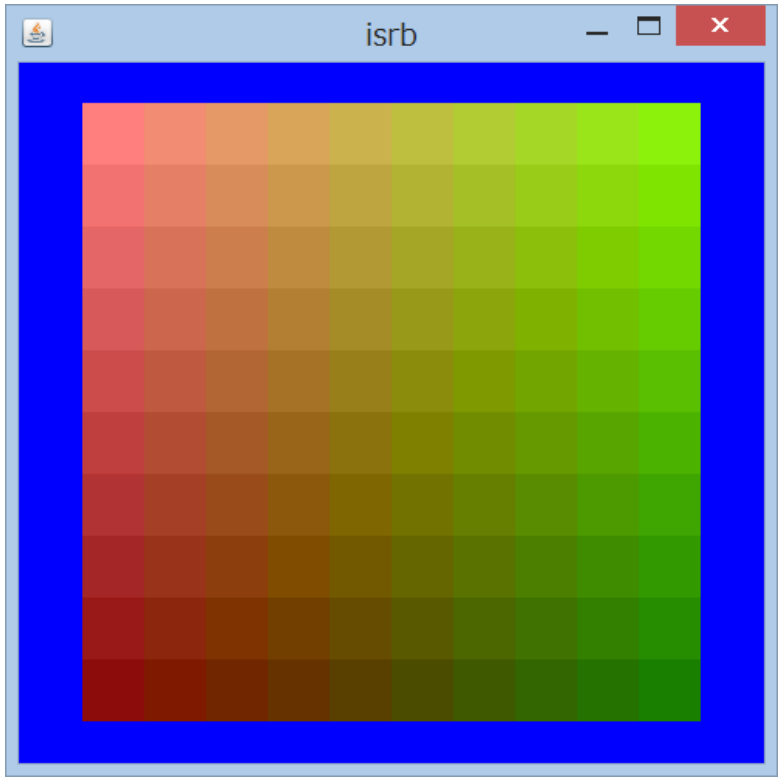
Make a 2-dim canvas
all entries are [0,0,0]

For each entry,
determine the fraction of colors(RGB)

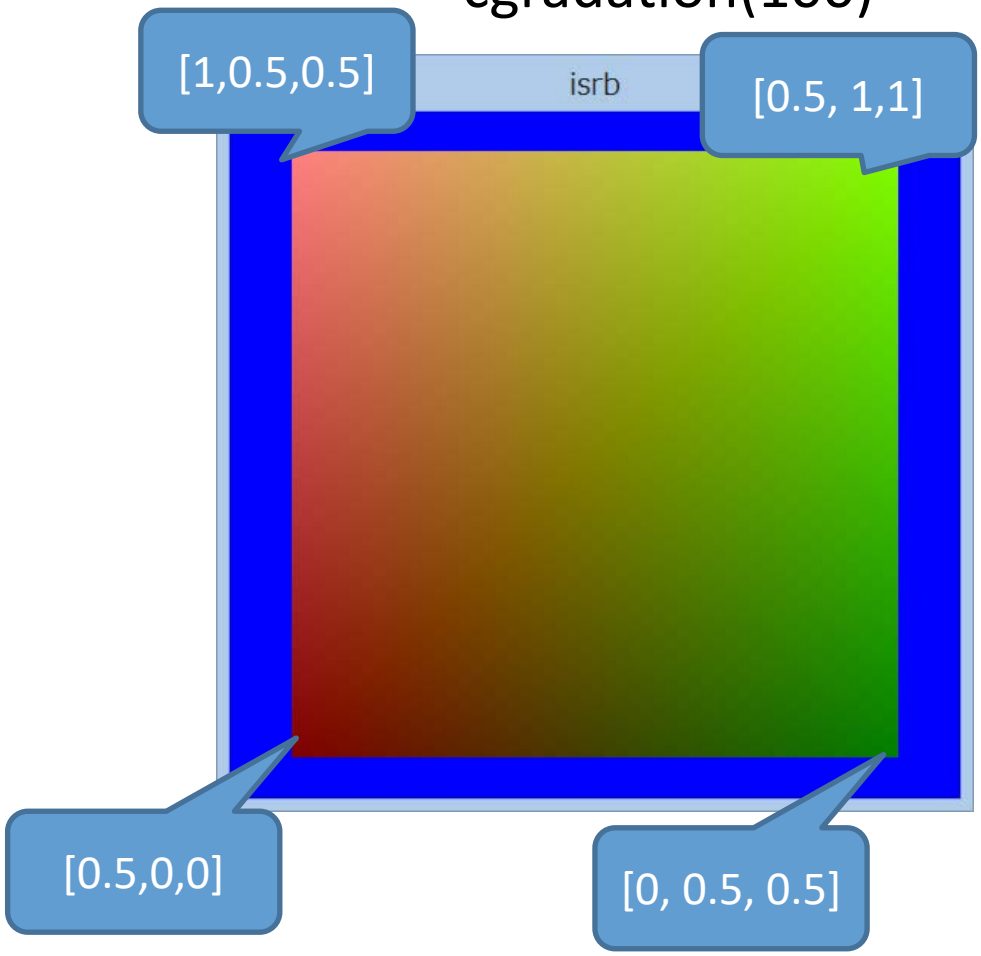
gradation.rb

Demonstration

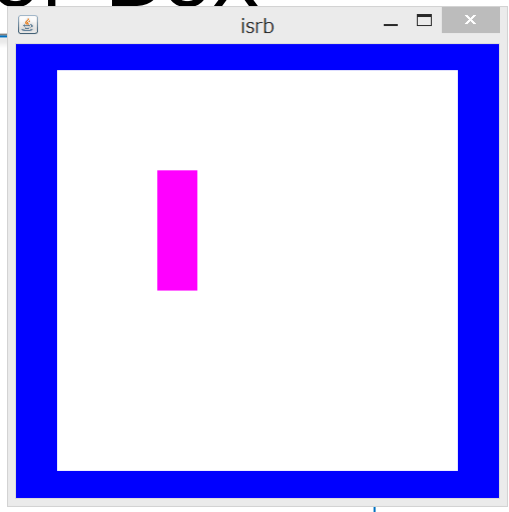
- cgradation(10)



cgradation(100)



Other Examples: Drawing a Color Box




```
def BoxImage(s)
  image = make2d_color(s,s)
  for x in 0..(s-1)
     for y in 0..(s-1)
        image[y][x]=[1,1,1] # paint white
       #If (x,y) is within some box, then paint purple
        if inBox(x,y,s/4,s/4,s/3,s/2)
           image[y][x]=[1,0,1]
        end
     end
   end
   image
end
```

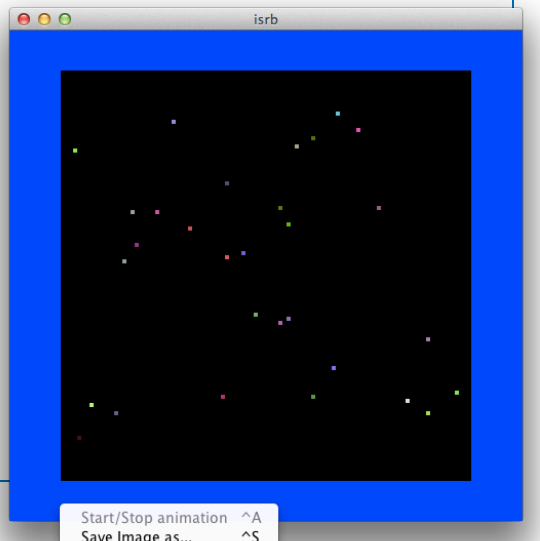
```
def inBox(x,y,a,b,c,d)
   a<=x && x<=c && b<=y && y<=d
end
```

Random-Point Generation


```
def randompoints(s, n)
    image = make2d_color(s, s)
    for i in 0..(n-1)
      y = rand(s)
      x = rand(s)
      image[y][x] = [rand(),rand(),rand()]
    end
    image
end
```

Repeat n times:
take a random point, and
paint with random color

randompoints(100, 30)



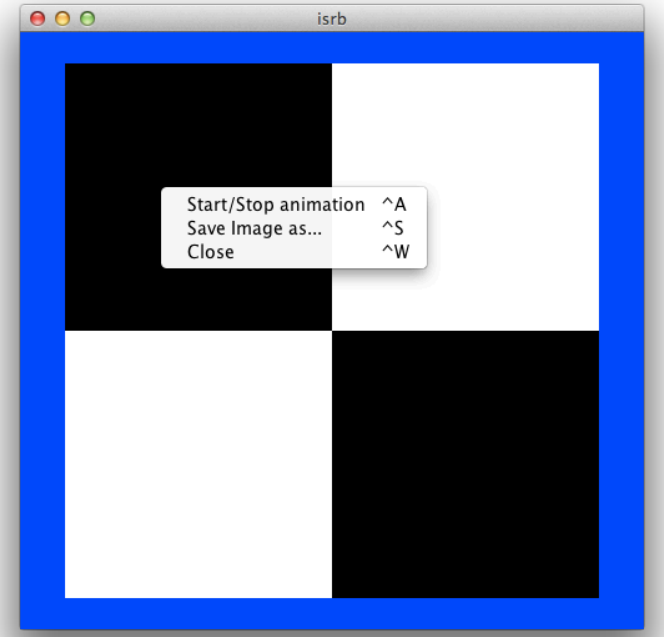
Today's Contents

- Making more images
  - Review of "gradation" and "Sphere"
  - Make a rectangle
  - Random points
  - Make a color image
  - Animation
- Exercises

Animation (available on isrb2 (not isrb))

- Repeating show(image) makes animation

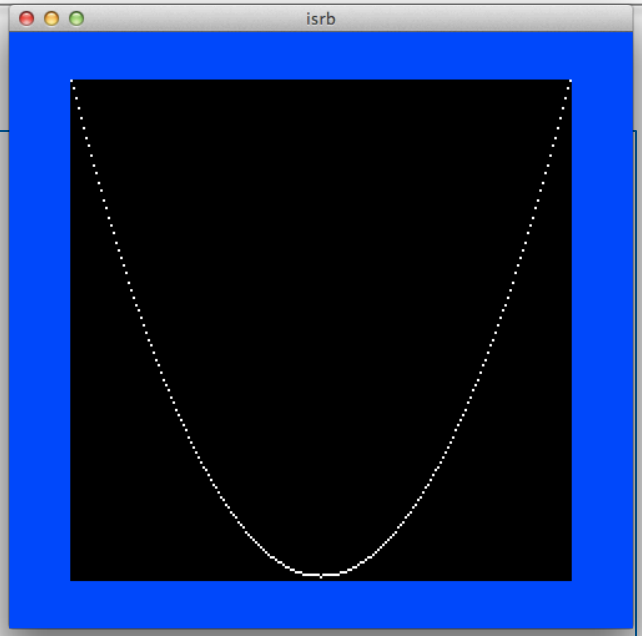
```
isrb(main):090:0> a=[[1,0],[0,1]]
isrb(main):091:0> show(a)
isrb(main):095:0> a[0][0]=0
isrb(main):096:0> a[0][1]=1
isrb(main):097:0> a[1][0]=1
isrb(main):098:0> a[1][1]=0
isrb(main):099:0> show(a)
```



The image is redrawn when you change the array a

If you click the image,
you have menu “start/stop animation”

Sample Animation

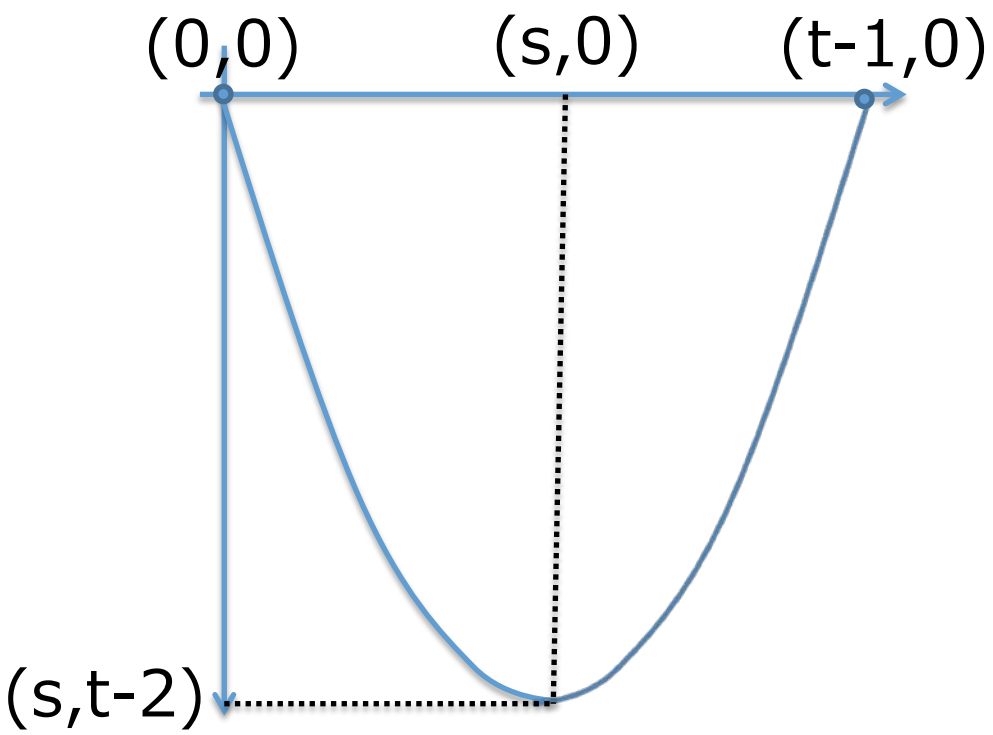


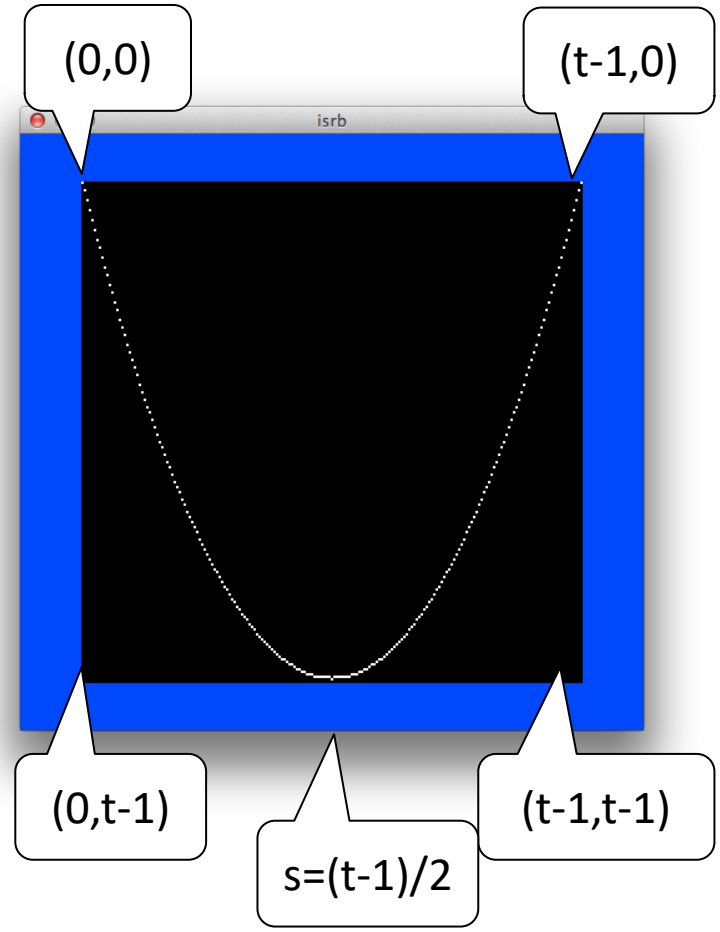

```
def quadratic(s)
    t = 2*s+1
    image = make2d(t, t)
    u = 1.0*(t-2)/s**2
    for x in 0..(t-1)
        y = -u*(x-s)**2 + t-2
        image[y][x] = 1
        show(image)
    end
    image
end
```

animation.rb

Animation of Quadratic Function

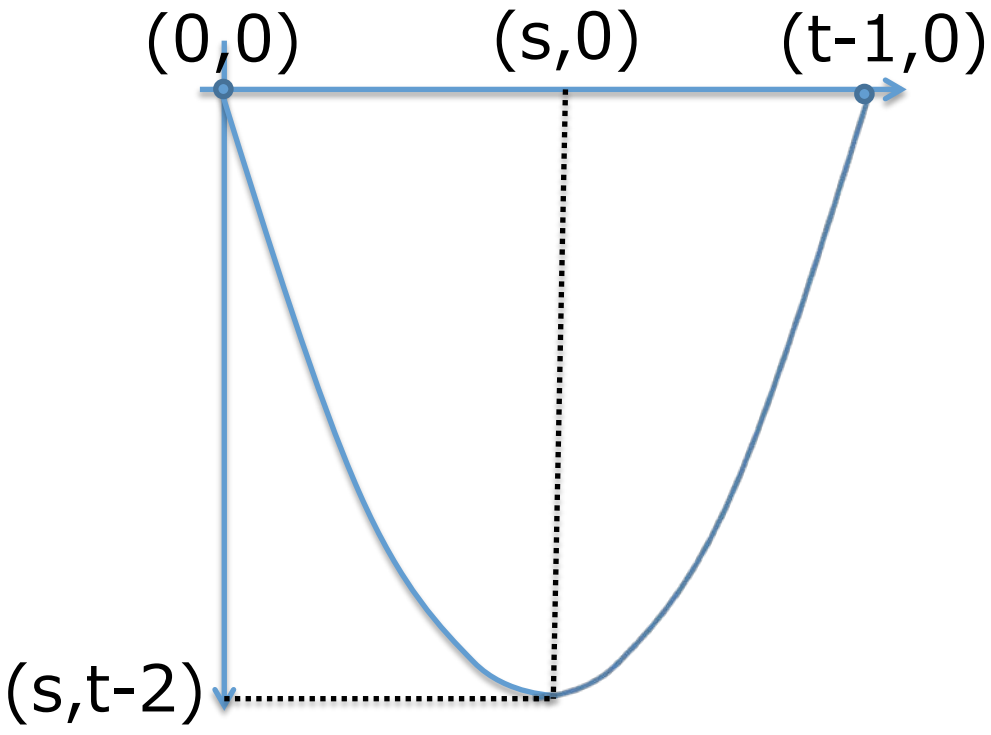
- Aim is to make the quadratic function below
  - Going through (0,0), (s,0), (t-1,0)





Review: Animation of Quadratic Function

- Aim is to make the quadratic function below
  - Going through (0,0), (s,0), (t-1,0)



Since the vertex is (s,t-2)

$$y=u(x-s)^2 + (t-2)$$

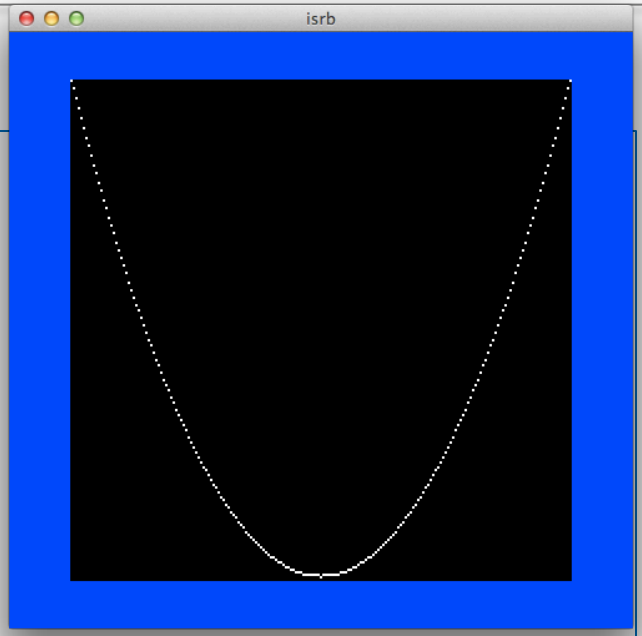
u:Defined later

Since it goes through (0,0)

$$0=u(-s)^2 + (t-2)$$

$$\Rightarrow u=-(t-2)/s^2$$

Sample Animation




```
def quadratic(s)
    t = 2*s+1
    image = make2d(t, t)
    u = 1.0*(t-2)/s**2
    for x in 0..(t-1)
        y = -u*(x-s)**2 + t-2
        image[y][x] = 1
        show(image)
    end
    image
end
```

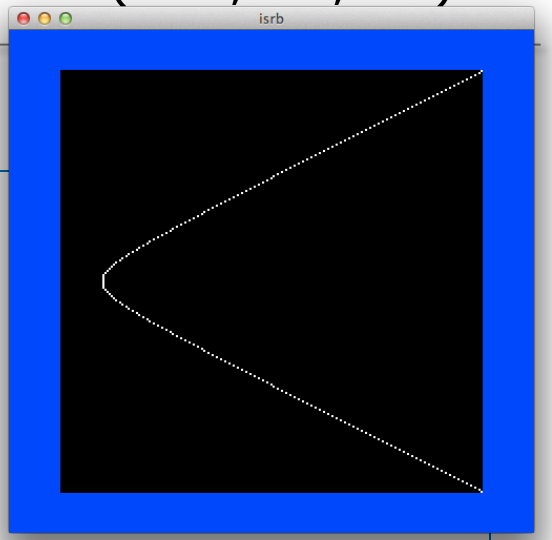
Each coordinate x determines the corresponding y
Color (x,y) white

Redraw the image each time you add a point

animation.rb

Sample Animation

Hyperbola(100, 20, 10)




```
def hyperbola(s,a,b)
    t=2*s+1
    image = make2d(t, t)
    for y in 0..(t-1)
        x = a * sqrt(1.0*(y-s)**2/b**2 + 1)
        if x < t
          image[y][x.to_i] = 1
        end
        show(image)
    end
    image
end
```

Coordinate has to be integer
"to_i" rounds down

Similar to quadratic(s)

animation.rb

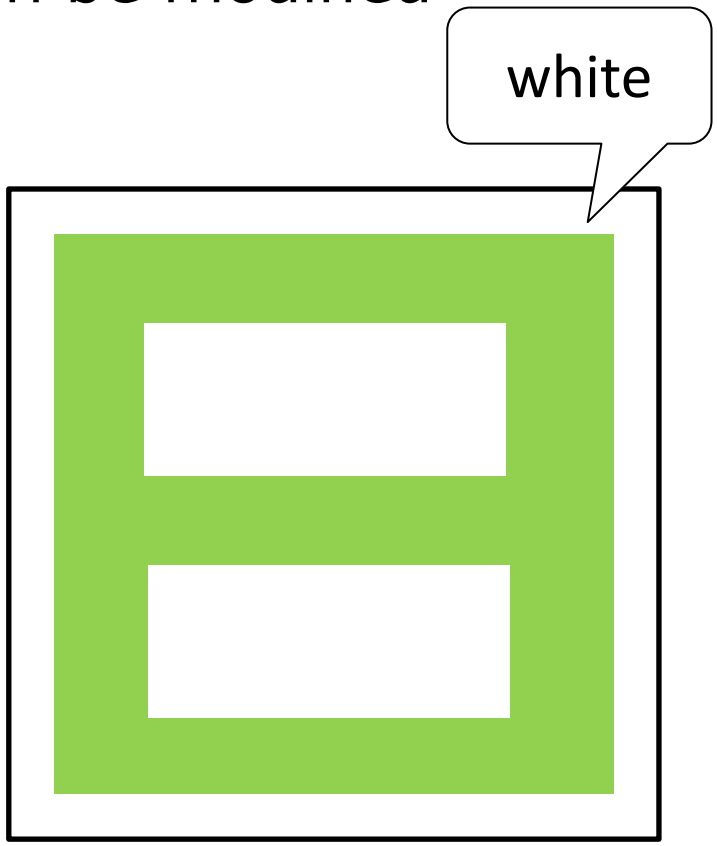
Today's Contents

- Making more images
  - Review of "gradation" and "Sphere"
  - Make a rectangle
  - Random points
  - Make a color image
  - Animation

- Exercises

Exercise 1

- Make a function to draw a picture as below
  - You can use function inBox
  - You can change the color of the object
  - The size is a parameter which can be modified

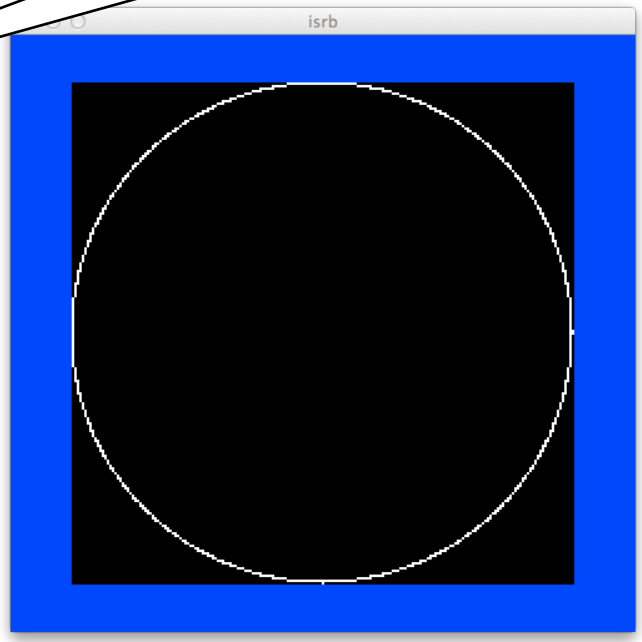


Exercise 2: Animation of Circle


```
def circle(s, n)
    t = 2*s+1
    image = make2d(t, t)
    for p in 0..(n-1)
        theta = p*2*PI/n
        y = (Fill in this part)
        x = (Fill in this part)
        image[y.to_i][x.to_i] = 1
        show(image)
    end
    image
end
```

n points on the circle

".to_i" makes the value integer

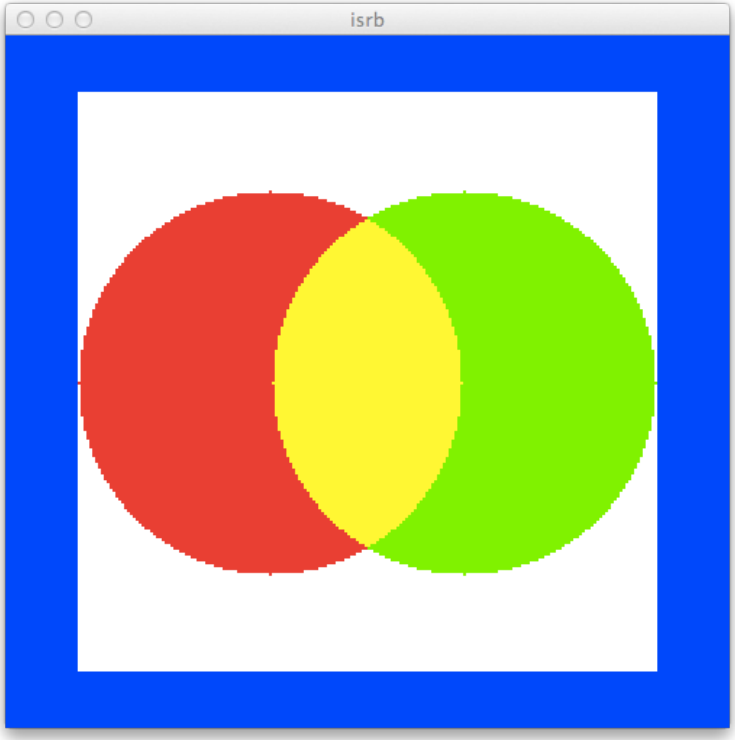


Hint: use sin &cos

Exercise 3(optional): Make the Image


```
def VennDiagram(s)
 image = make2d_color(s, s)
 for x in 0..(s-1)
   for y in 0..(s-1)
      # paint white
      image[y][x] = [1,1,1]
     (Fill in this part using inCircle)
  end
 end
 image
end

 def inCircle(x,y,a,b,r)
   (x-a)**2+(y-b)**2 <= r**2
 end
```

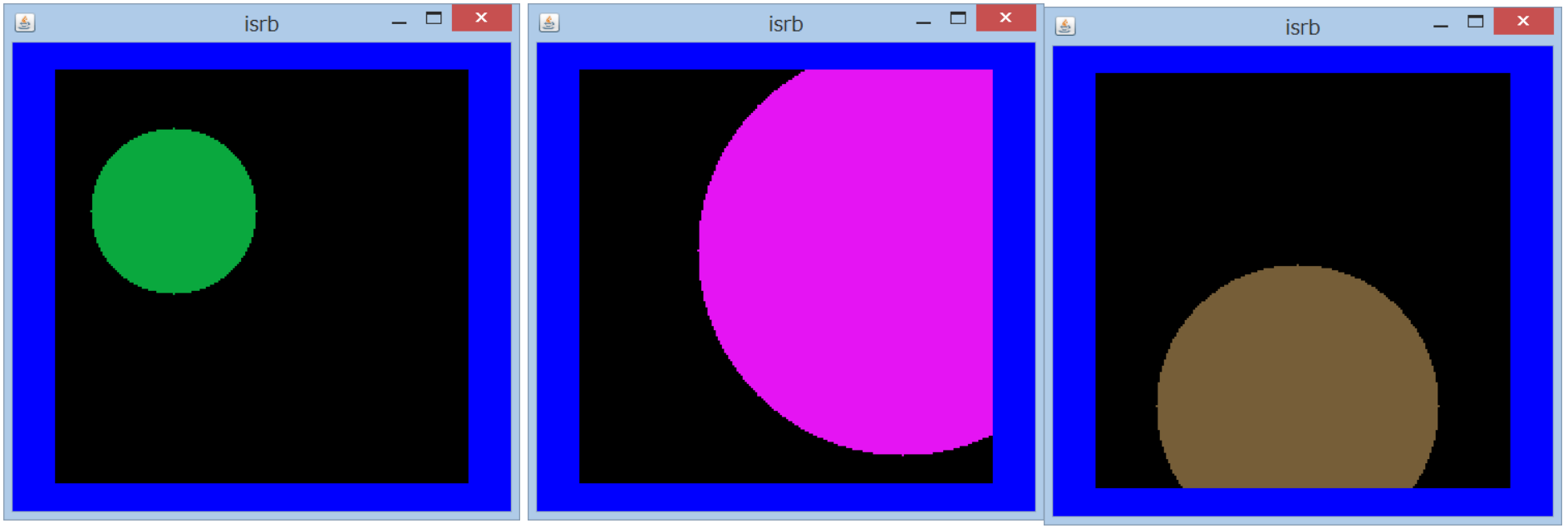
s=200



Exercise 4

- Draw one random circle using rand()
  - Center, radius, and color are determined randomly
    - Different results are obtained each time
- You can use inCircle

```
def inCircle(x,y,a,b,r)
  (x-a)**2+(y-b)**2 <= r**2
end
```



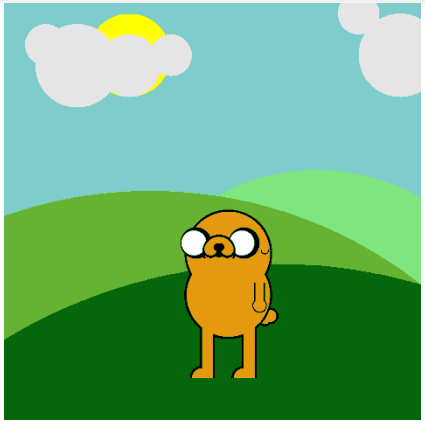
BIG Home Assignment

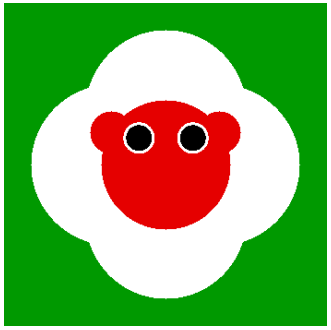
- Make your original image by using array and repetition
  - Your function has to have a parameter to specify the size of an image
  - Your program has to use “if” and “while/for”
  - You can modify example functions in slides

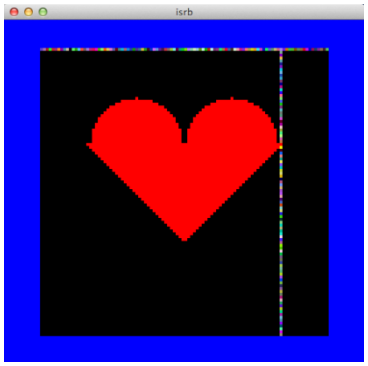
- Cf) Images created by other students
  - http://lecture.ecc.u-tokyo.ac.jp/~kuno/is13/report/gallery.html
  - http://lecture.ecc.u-tokyo.ac.jp/~kuno/is12/report/gallery.html
  - http://lecture.ecc.u-tokyo.ac.jp/~kuno/is11/report/gallery.html

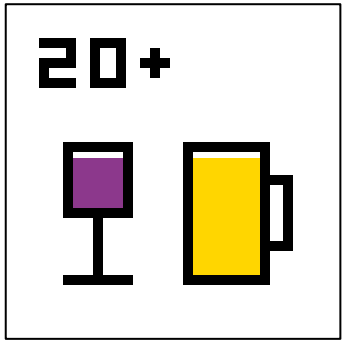
  How to save an animation image is different from ours

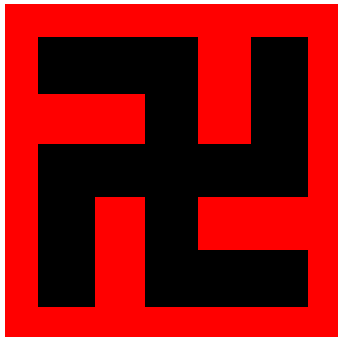
Examples from 2014

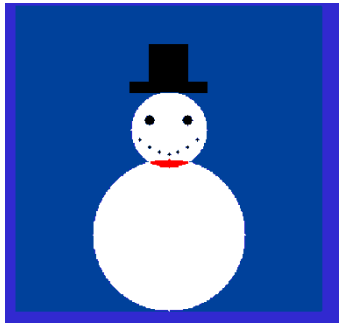


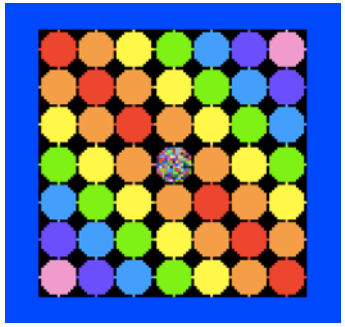












Examples from 2015

- http://www.graco.c.u-tokyo.ac.jp/labs/kakimura/Lecture/IS2015/ImagePEAK2015.html

Deadline for Home Assignment

- Deadline for Today's Two Exercises
  - **Nov. 9 (Wed) 23:59**
- Deadline for Home Assignment (1 month away)
  - **Nov. 30 (Wed) 23:59**
  - How to submit
    - Save it as an image file/get a screenshot of image
    - Submit your program with an image through ITC-LMS
- Rem.
  - Your images will be public in the class

Next Week (Nov. 7,Mon)

- Concept of Recursion
  - contrasting repetition