5. Drawing in a window

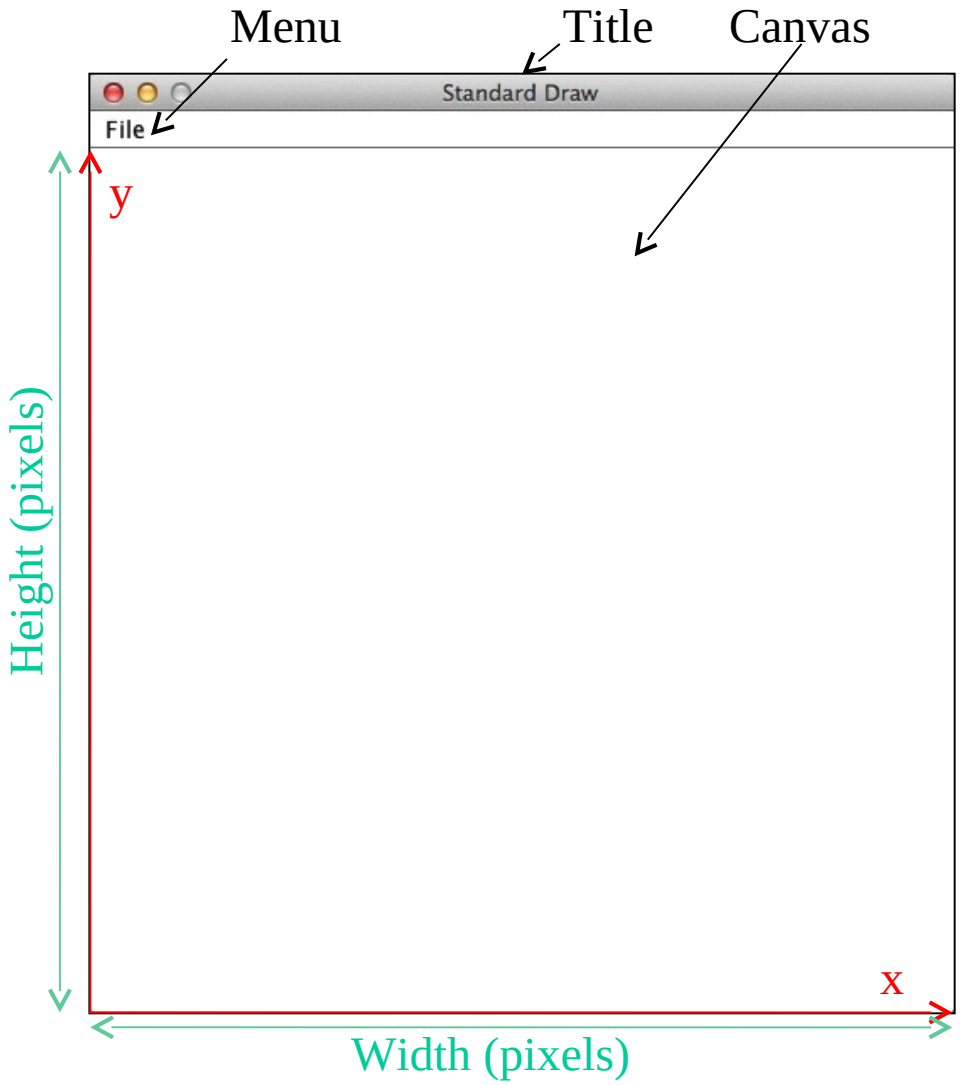
Prof. Dr. Charles Wüthrich
B.Sc. Francesco Andreussi,
CoGVis/MMC, Faculty of Media
Bauhaus-University Weimar

Introduction

- We have now “some” control on
  - Program flow
  - Easy data structures such as arrays
- Next we will jump into how to use windows and draw on it
- For this purpose, we will use the University of Princeton `StdDraw` library which provides
  - Methods for drawing points, lines
  - Methods for drawing squares, circles, polygons and fill them
  - Methods for inserting pictures and text
  - Methods for displaying animations
  - Methods for listening to keyboard and mouse input
  - Methods for saving the window we are drawing into

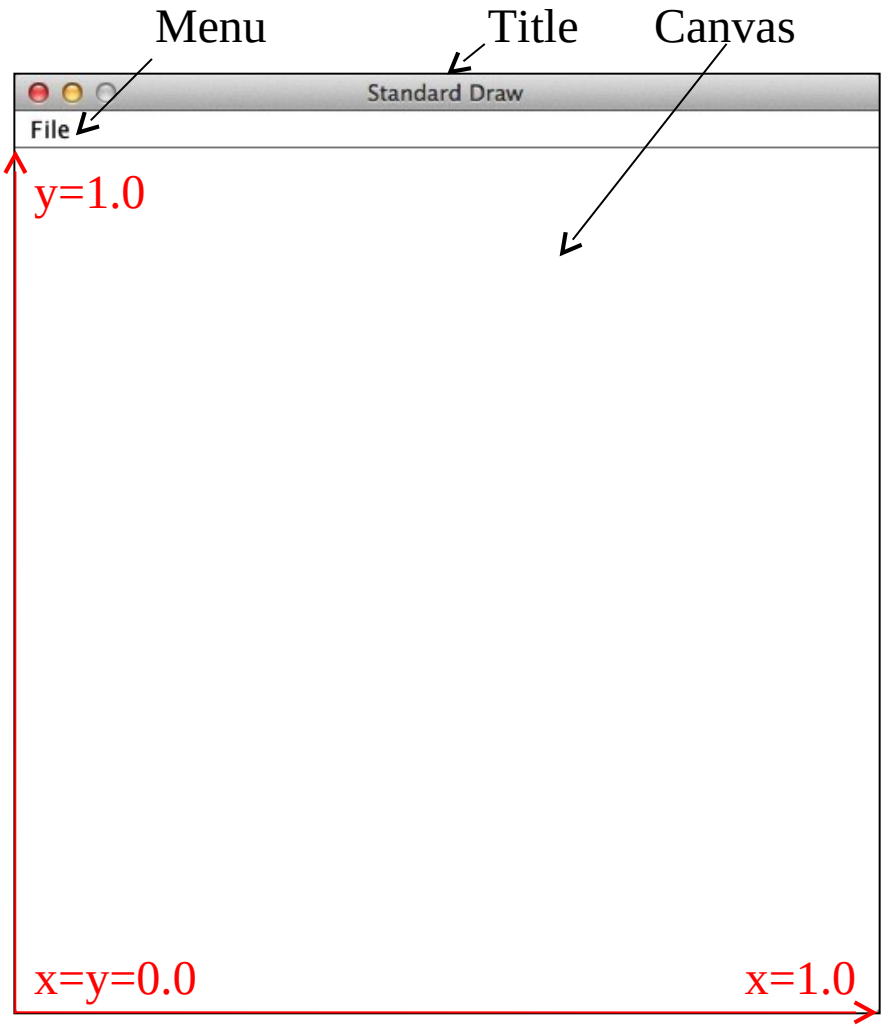
Windows

- In Modern systems, a *window* is an autonomous subsection of the computer screen.
- It can be seen roughly as a two-dimensional array of special cells called pixels.
- The window has a width and a height (number of pixels)
  - For example, a 1080p image has 1920x1080 pixels
- Two axes, x and y are used for the coordinates of the points in the window, i.e. on its canvas
- Windows “catch” mouse/keyboard input
- By default, windows are 512x512 pixels



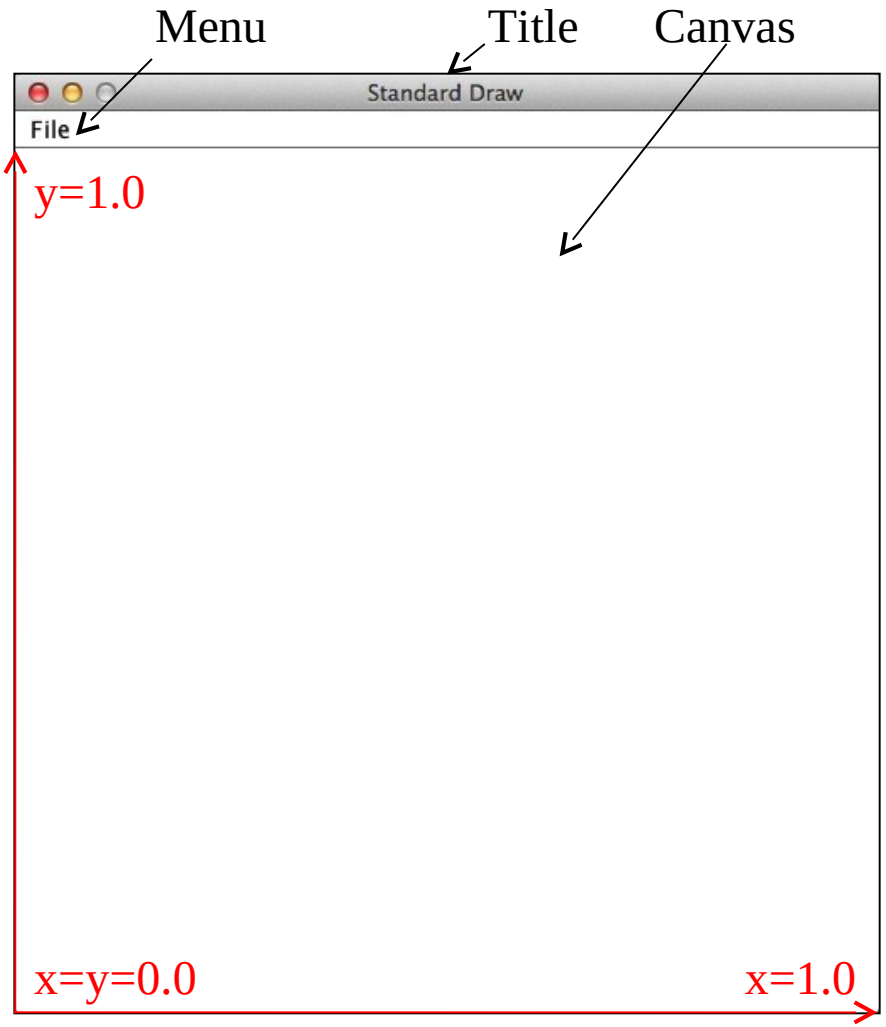
Windows in StdDraw

- In StdDraw, x and y axes have coordinates varying between 0 and 1
- Drawable canvas starts below the menu bar
- Instead of integer numbers, StdDraw uses doubles for its coordinates
- This is a bit confusing, but allows not to change things when the window is resized by the user
- How does one convert from pixels to numbers between 0 and 1?



Converting pixels to [0,1]

- If the window is in pixels Xpixels x Ypixels, then
  - Xpixels must be 1.0 as coordinate
  - Ypixels must also be 1.0
- Thus the position of the pixel with coordinates (x,y) is:
  - Xcoordinate=(1.0/Xpixels)*x
  - Ycoordinate=(1.0/Ypixels)*y
- Where Xcoordinate and Ycoordinate are the values between 0 and 1 of the point
- Notice that pixels have integer coordinates, but canvases have double coordinates, so type casting needed



Notation notice

- Notice that ALL methods of the library `StdDraw` require when programming to be called with

  `StdDraw.<themethodname>`

  as a prefix
- This means, whenever hereafter we talk about the method `themethodname`, when programming you have to use it by typing `StdDraw.themethodname`.
- So now let us write our first program drawing into a window to test if we can….

My first drawing

```
public class TestStdDraw {
   public static void main(String[] args) {
      StdDraw.setPenRadius(0.05);
      StdDraw.setPenColor(StdDraw.BLUE);
      StdDraw.point(0.5, 0.5);
      StdDraw.setPenColor(StdDraw.GREEN);
      StdDraw.line(0.2, 0.2, 0.8, 0.2);
   }
}
```

- Set the pen thickness: `setPenRadius(dpensize)`
  where pen thickness is type double
- Set pen color: `setPenColor(red,green,blue)`
  where red, green blue are integers
- Draw point: `point(xcoord,ycoord)` as doubles
- Draw line: `line(x1coord,x2coord,y1coord,y2coord)` doubles
- Notation notice: we indicated types required in a method by explicitly listing them in the round bracket, e.g for the point function:
  `point(double xcoord,double ycoord);`

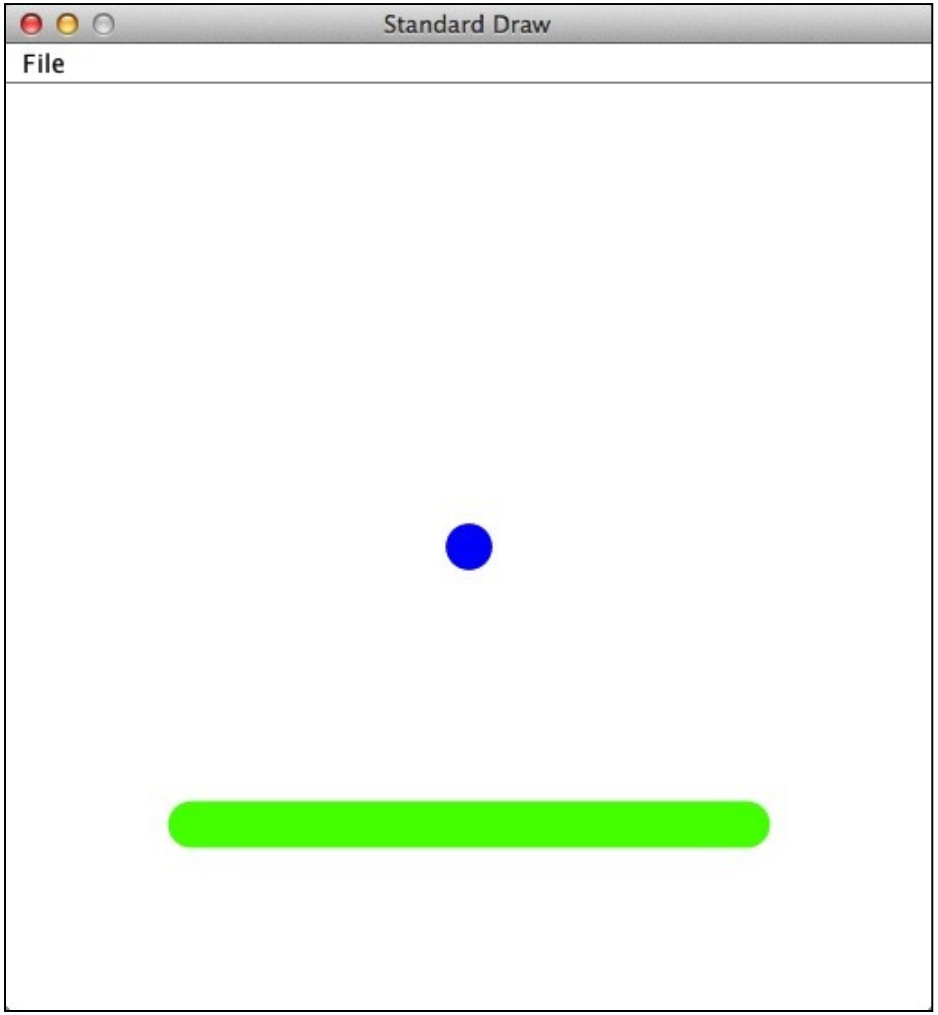
Pen is circular!

My first Drawing

- If we compile, and execute: what do we obtain?

My first Drawing

- If we compile, and execute: what do we obtain?



- Amazing!!!!!
  A WINDOW with a drawn CANVAS!!!!

My first Drawing

```
public class TestStdDraw {
   public static void main(String[] args) {
      StdDraw.setPenRadius(0.05);
      StdDraw.setPenColor(StdDraw.BLUE);
      StdDraw.point(0.5, 0.5);
      StdDraw.setPenColor(StdDraw.GREEN);
      StdDraw.line(0.2, 0.2, 0.8, 0.2);
   }
}
```

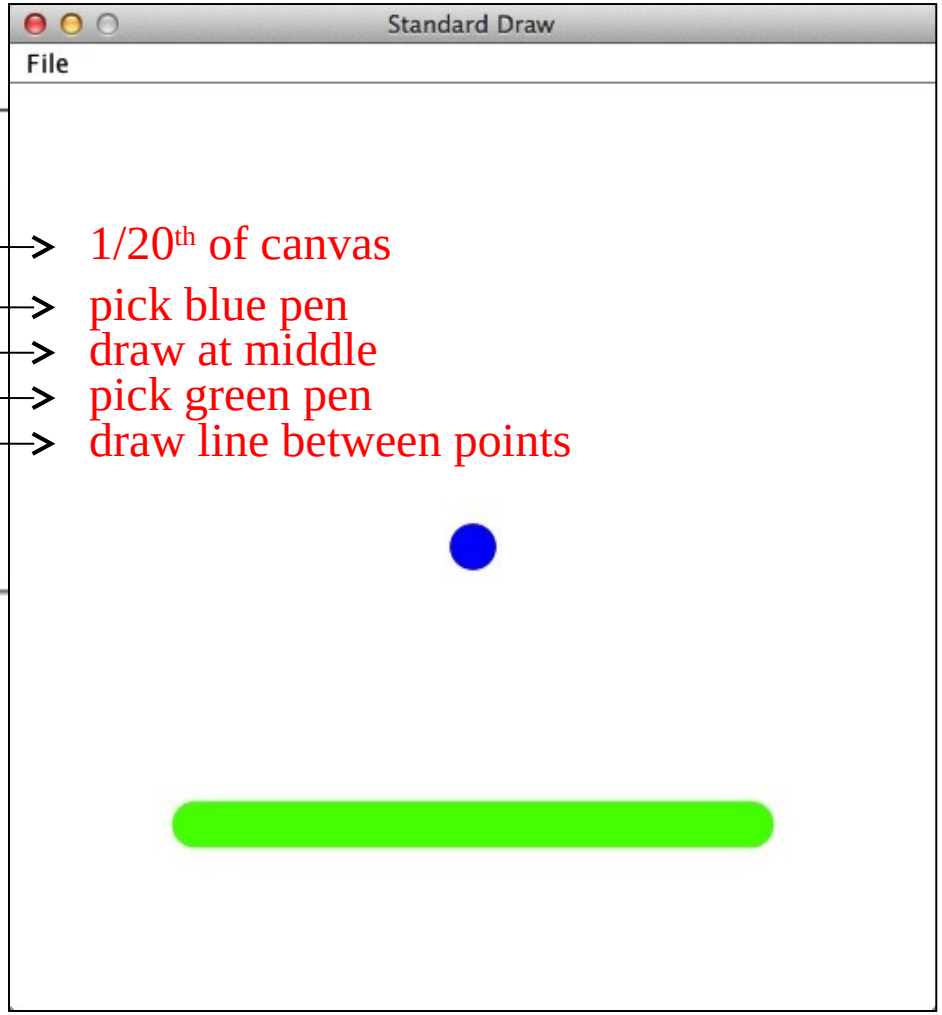
→ 1/20th of canvas

→ pick blue pen

→ draw at middle

→ pick green pen

→ draw line between points



Note: StdDraw.GREEN is (0,255,0) for RGB

Changing canvas size

- Set the canvas size:

```
setCanvasSize(int width, int height);
```

- Change coordinate scale:

```
setXscale(double xmin, double xmax);
setYscale(double ymin, double ymax);
setScale(double min, double max);
```

Changing canvas size

- Set the canvas size:

```
setCanvasSize(int width, int height);
```

- Change coordinate scale:

```
setXscale(double xmin, double xmax);   <---- for x
setYscale(double ymin, double ymax);   <---- for y
setScale(double min, double max);      <---- for both
```

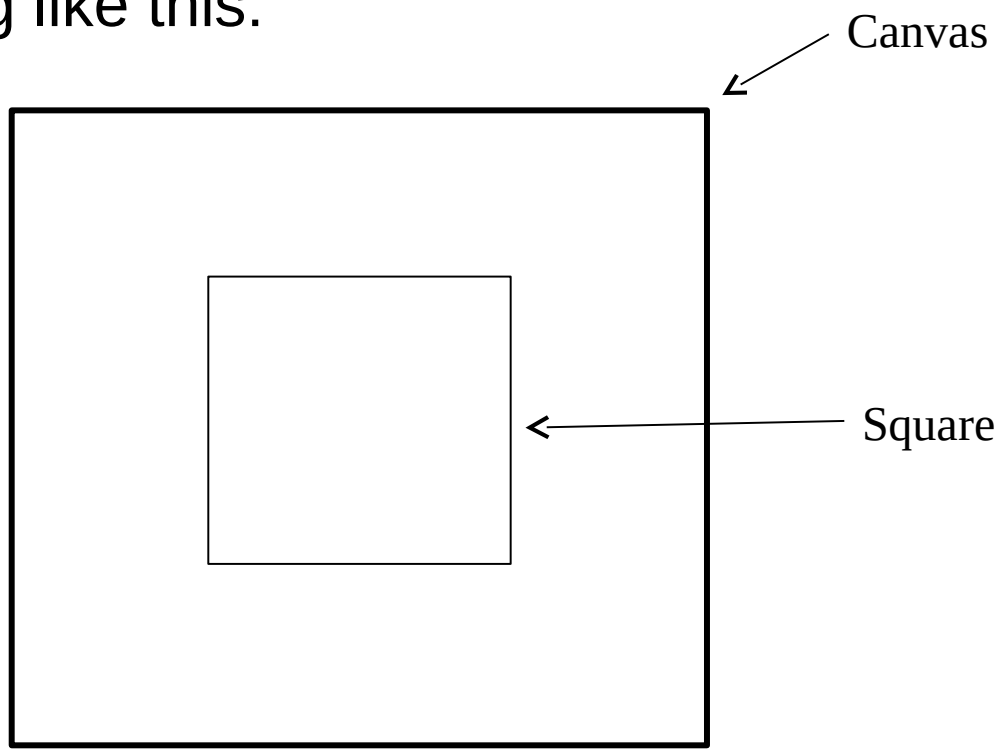
- So if you write in your program

```
StdDraw.setScale(0.0, 512.0);
```

you will let both x and y coordinates run from 0 to 512

Line Drawing methods

- Draw the perimeter of a square
  - Centered in the middle of the canvas
  - User specifies size of the square
- Something like this:



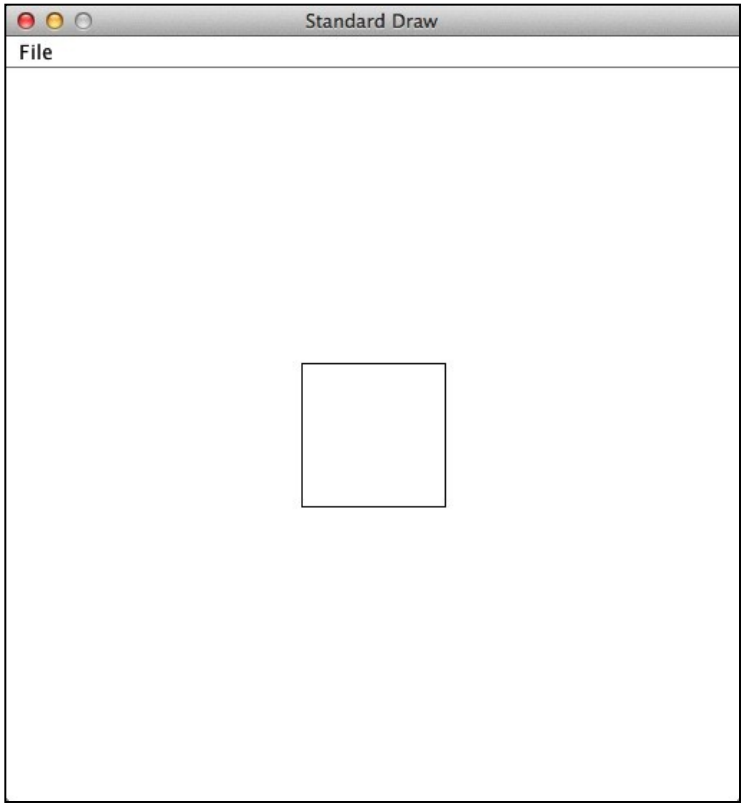
Line Drawing methods

```
// DrawSquare: draws a square centered in the middle and
// of the size specified by the user in the command line
public class DrawSquare {
   public static void main(String[] args) {
      // Endpoint coordinates of the square
      double p1_x, p1_y, p2_x, p2_y, p3_x, p3_y, p4_x, p4_y;
      // size of the canvas
      int canvassize = 512;

      // set canvascoordinates to be between 0 and 512
      // instead of the default 0 and 1
      StdDraw.setCanvasSize(canvassize,canvassize);
      StdDraw.setXscale(0.0,(double)canvassize);
      StdDraw.setYscale(0.0,(double)canvassize);

      // Fill corner coordinates of the square:
      // up left
      p1_x=(double)canvassize/2.0-(double)size/2.0;
      p1_y=(double)canvassize/2.0+(double)size/2.0;
      // up right
      p2_x=(double)canvassize/2.0+(double)size/2.0;
      p2_y=(double)canvassize/2.0+(double)size/2.0;
      // down left
      p3_x=(double)canvassize/2.0-(double)size/2.0;
      p3_y=(double)canvassize/2.0-(double)size/2.0;
      // down right
      p4_x=(double)canvassize/2.0+(double)size/2.0;
      p4_y=(double)canvassize/2.0-(double)size/2.0;

      StdDraw.line(p1_x,p1_y,p2_x,p2_y);
      StdDraw.line(p2_x,p2_y,p4_x,p4_y);
      StdDraw.line(p4_x,p4_y,p3_x,p3_y);
      StdDraw.line(p3_x,p3_y,p1_x,p1_y);

   } // main
} // DrawSquare
```

Line Drawing methods

- Compile, run by typing in the CLI:
  >java DrawSquare

- …and voila!



More Drawing methods

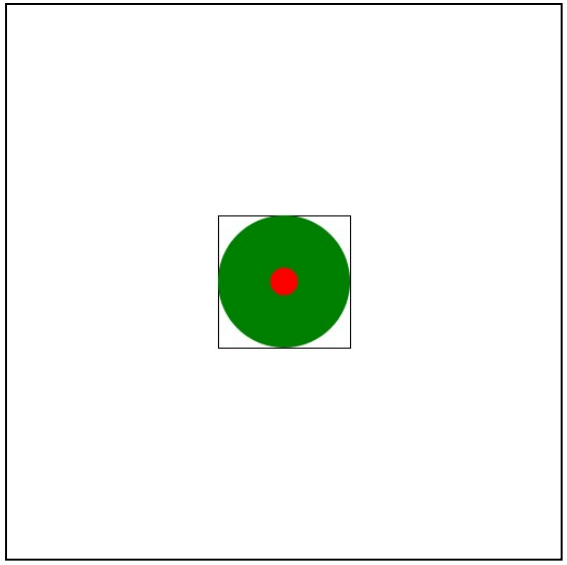
- Now we can draw, let's add more drawing functions:
  - `square(double x, double y, double radius)`
  - `circle(double x, double y, double radius)`
  - `rectangle(double x, double y, double halfWidth, double halfHeight)`
  - `ellipse(double x, double y, double semiMajorAxis, double semiMinorAxis)`
- Filled versions:
  - `filledSquare(double x, double y, double radius)`
  - `filledCircle(double x, double y, double radius)`
  - `filledRectangle(double x, double y, double halfWidth, double halfHeight)`
  - `filledEllipse(double x, double y, double semiMajorAxis,double semiMinorAxis)`
- Notice: x and y are the CENTER of the object

More Drawing methods

- If I add the following code to the program above

```
StdDraw.setPenColor(0,128,0);
StdDraw.filledCircle((double)canvassize/2.0,(double)canvassize/2.0,
                     (double)size/2.0);

StdDraw.setPenColor(255,0,0);
StdDraw.setPenRadius(0.05);
StdDraw.point((double)canvassize/2.0,(double)canvassize/2.0);
```

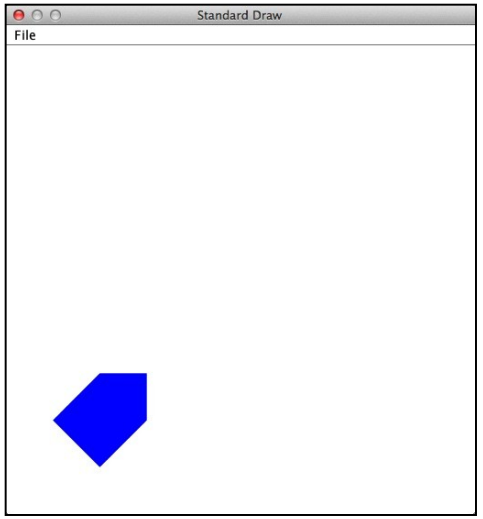
one obtains this
in the canvas:



More drawing methods

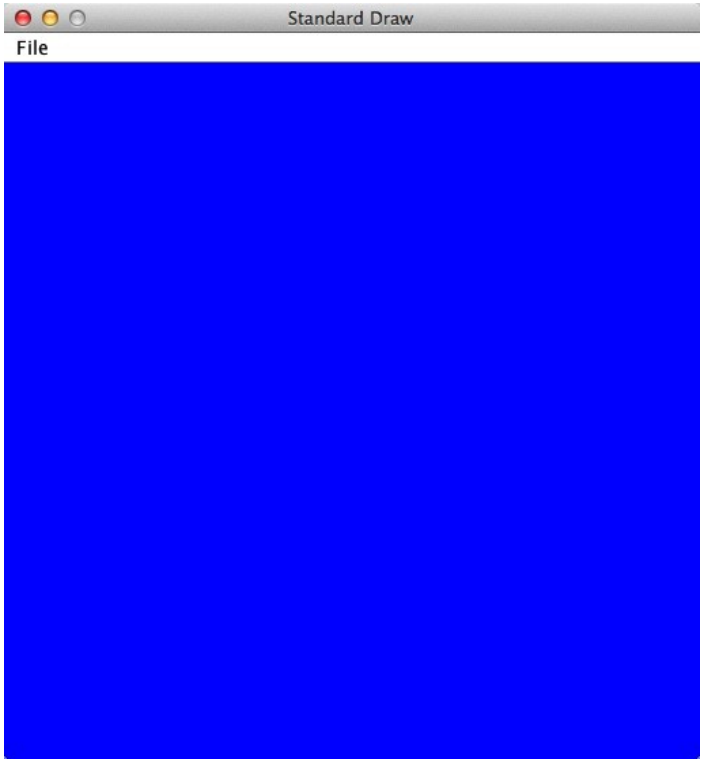
- More drawing functionality:
  - Circular arcs:
    `arc(double x, double y, double radius, double angle1, double angle2)`
  - Polygons: you need two vectors of `x[]` and `y[]`:
    `polygon(double[] x, double[] y)`
  - Filled polygons:
    `filledPolygon(double[] x, double[] y)`
- Ex:

```
// DrawPoly: draws a Polygon
public class DrawPoly {
    public static void main(String[] args) {
        // Poly coords
        double[] x = { 0.1, 0.2, 0.3, 0.3, 0.2 };
        double[] y = { 0.2, 0.3, 0.3, 0.2, 0.1 };

        // set pen blue
        StdDraw.setPenColor(0,0,255);
        // draw poly
        StdDraw.filledPolygon(x, y);

    } // main
} // DrawPoly
```



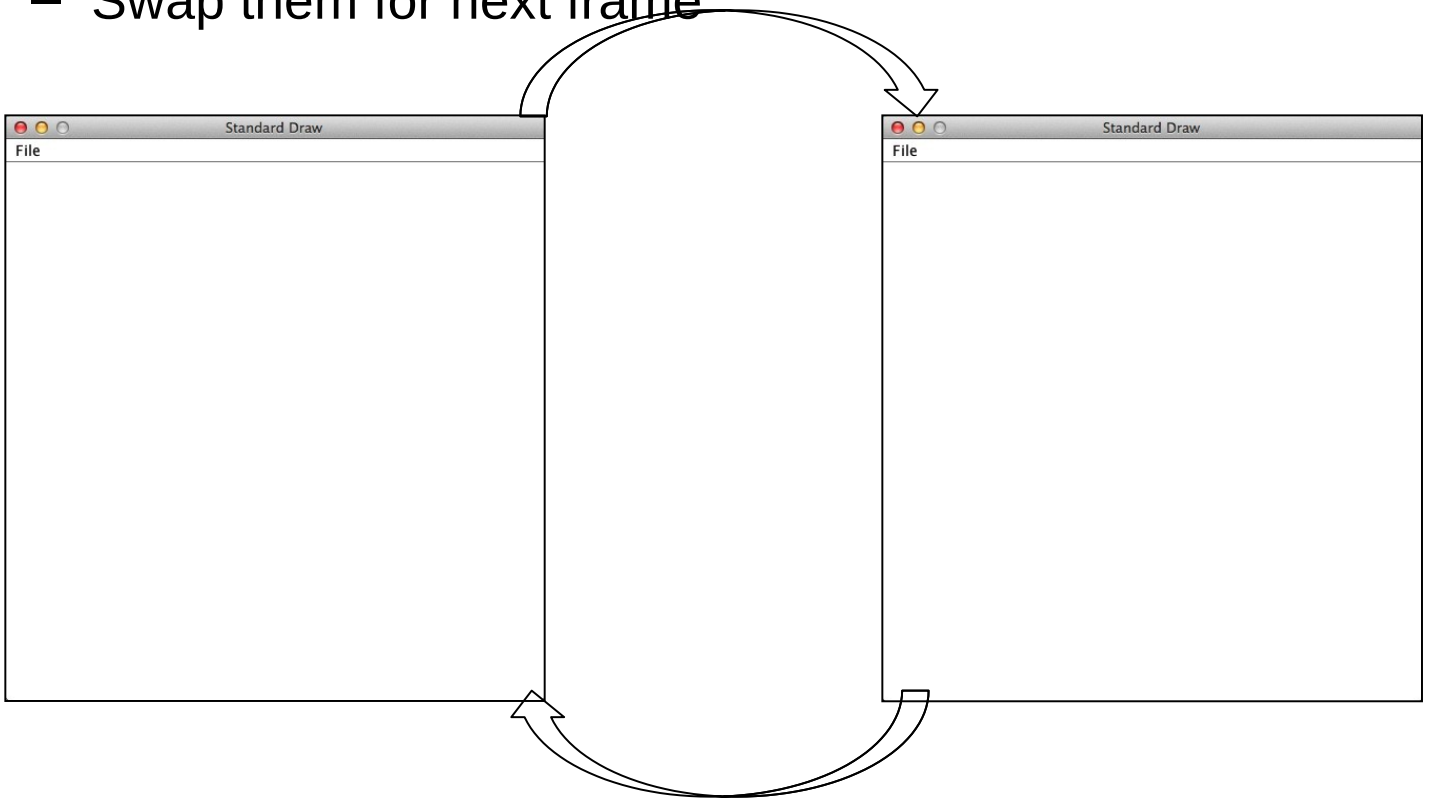
Clear the canvas

- Two methods for clearing your canvas:
  - `clear();`
  - `clear(Color color);`
- Example: `StdDraw.clear(StdDraw.BLUE);`



Double buffering

- For animation, you need two canvases
  - Write in one
  - Display the other one
  - Swap them for next frame



Double Buffering

- Four relevant methods:
  - `enableDoubleBuffering()`
  - `disableDoubleBuffering()`
  - `show()` swaps the buffers
  - `pause(int t)` waits `t` milliseconds

Double Buffering

```
// PingBall: Pings red ball left-right

public class PingBall {
    public static void main(String[] args) {

        // Endpoint coordinates
        int canvassize = 512;
        double dcanvassize,dsize;
        double x,y;
        double t;
        double dsign=-1.0;

        // dcanvassize=(double)canvassize;

        // read canvas size
        int size = Integer.parseInt(args[0]);
        dcanvassize=(double)size;
        // int size = 36;

        // set canvascoordinates to be between 0 and the maximum
        // canvas size instead of the default 0 and 1
        StdDraw.setCanvasSize((int)dcanvassize,(int)dcanvassize);
        StdDraw.setXscale(0.0,dcanvassize);
        StdDraw.setYscale(0.0,dcanvassize);

        StdDraw.setPenColor(255,0,0);
```

```
        // Enable doublebuffering
        StdDraw.enableDoubleBuffering();

        for (t = 0.0; true; t += 0.02) {
           dsign=dsign*(-1.0);
           x = dcanvassize/2.0+dsign*dcanvassize/4.0;
           y = dcanvassize/2.0;
           StdDraw.clear();
           // StdDraw.point(x, y);
           StdDraw.filledCircle(x, y, 0.05*dcanvassize);
           StdDraw.show();
           StdDraw.pause(1000); // milliseconds
        }


    } // main
} // PingBall
```

Moving objects

```
// MoveBall: moves a ball

public class MoveBall {
    public static void main(String[] args) {

        // Endpoint coordinates
        int canvassize = 512;
        double dcanvassize,dsize;
        double x,y;
        double t;
        double dsign=-1.0;
        double radius=0.0;

        // read canvas size
        int size = Integer.parseInt(args[0]);
        dcanvassize=(double)size;

        // set canvascoordinates to be between 0 and the maximum
        // canvas size instead of the default 0 and 1
        StdDraw.setCanvasSize((int)dcanvassize,(int)dcanvassize);
        StdDraw.setXscale(0.0,dcanvassize);
        StdDraw.setYscale(0.0,dcanvassize);
        // StdDraw.setPenRadius(0.05);
        StdDraw.setPenColor(255,0,0);
        StdDraw.enableDoubleBuffering();

        radius=dcanvassize*0.05;
        x=0; y=0;  // set begin at (0,0)

        for (t = 0.0; true; t += 0.02) {
           // dsign=dsign*(-1.0);
           x = t*dcanvassize/2.0;
           y = t*dcanvassize/2.0;
           StdDraw.clear();
           // StdDraw.point(x, y);
           StdDraw.filledCircle(x, y, 0.05*dcanvassize);
           StdDraw.show();
           StdDraw.pause(20); // milliseconds
           // System.out.println(dsign);
        }

    } // main
} // MoveBall
```

Bouncing objects

```
// BounceRedBall
// usage: java BounceRedBall canvassize startx starty
// where: canvassize: size of the square canvas in pixels
//        startx: x of the starting point in pixels
//        starty: y of the starting point in pixels

public class BounceRedBall {
    public static void main(String[] args) {

        // Endpoint coordinates
        int canvassize = 512;
        double dcanvassize,dsize;
        double x,y;
        double s,t;
        int signs, signt;
        double dincrement;
        double dsign=-1.0;
        double radius=0.0;
        int i=0;

        // read canvas size
        int size = Integer.parseInt(args[0]);
        dcanvassize=(double)size;
        int startx = Integer.parseInt(args[1]);
        int starty = Integer.parseInt(args[2]);


        // set canvascoordinates to be between 0 and the maximum
        // canvas size instead of the default 0 and 1
        StdDraw.setCanvasSize((int)dcanvassize,(int)dcanvassize);
        StdDraw.setXscale(0.0,dcanvassize);
        StdDraw.setYscale(0.0,dcanvassize);
```

```
// StdDraw.setPenRadius(0.05);
StdDraw.setPenColor(255,0,0);
StdDraw.enableDoubleBuffering();

radius=dcanvassize*0.05;
x=0; y=0;  // set begin at (0,0)
dincrement=0.02;
s=0.0;
t=0.0;
signt=1;
signs=1;
s=(double)starty;
t=(double)startx;

s=(s/dcanvassize);
t=(t/dcanvassize);
```

Bouncing objects

```
        for (i=0; true; i++) {
            t = t+ dincrement*(double)signt;
            s = s+ dincrement*(double)signs;

            x = t*dcanvassize;
            y = s*dcanvassize;

            StdDraw.clear();
            // StdDraw.point(x, y);
            StdDraw.filledCircle(x, y, 0.05*dcanvassize);
            StdDraw.show();
            StdDraw.pause(20); // milliseconds
            if((x > dcanvassize)||(x<0.0)){
                signt=signt * (-1);
            }
            if((y > dcanvassize)||(y<0.0)){
                signs=signs * (-1);
            }
        }


    } // main
} // BounceRedBall
```

Mouse and keyboard events

- StdDraw has also methods for querying mouse events:
  - `MousePressed()` true if mouse has been pressed, else false (*)
  - `mouseX()` returns the mouse X coordinate as a double
  - `mouseY()` returns the mouse Y coordinate as a double
- And similarly for keyboard keys
  - `hasNextKeyTyped()` true if user pressed a new key
  - `nextKeyTyped()` returns the character the user has typed
  - `isKeyPressed(int keycode)` returns true if the key of code `keycode` has been pressed

  - `(*) newer versions use isMousePressed() instead`

Example

```
public class MouseFollower {
   public static void main(String[] args) {

      StdDraw.enableDoubleBuffering();

      while (true) {

         // mouse click
         if (StdDraw.mousePressed()){
            StdDraw.setPenColor(StdDraw.CYAN);
         }
         else{
            StdDraw.setPenColor(StdDraw.BLUE);
         }
         // mouse location
         StdDraw.clear();
         double x = StdDraw.mouseX();
         double y = StdDraw.mouseY();
         StdDraw.filledCircle(x, y, 0.05);
         StdDraw.show();
         StdDraw.pause(10);
      }
   }
}
```

Example 2

```
// MouseClick: Draws a circle at mouse position
// usage: java MouseClick

public class MouseClick {
   public static void main(String[] args) {
      double x1,y1;
      boolean ispressed=false;
      double radius; // radius of circle

      StdDraw.setPenColor(255,0,0);
      radius=0.01;

      while (true) {
        ispressed=StdDraw.mousePressed();
        if(ispressed){
          x1=StdDraw.mouseX();
          y1=StdDraw.mouseY();
          StdDraw.filledCircle(x, y, radius);
          ispressed=false;
        }
      }
   } // main
} // MouseClick
```

End

+++ Ende - The end - Finis - Fin - Fine +++ Ende - The end - Finis - Fin - Fine +++