Programming of Interactive Systems

Anastasia Bezerianos

introduction.prog.is@gmail.com

Housekeeping

The Minesweeper v2 assignment is due on 13 Oct

(tomorrow show your project progress to TAs)

2

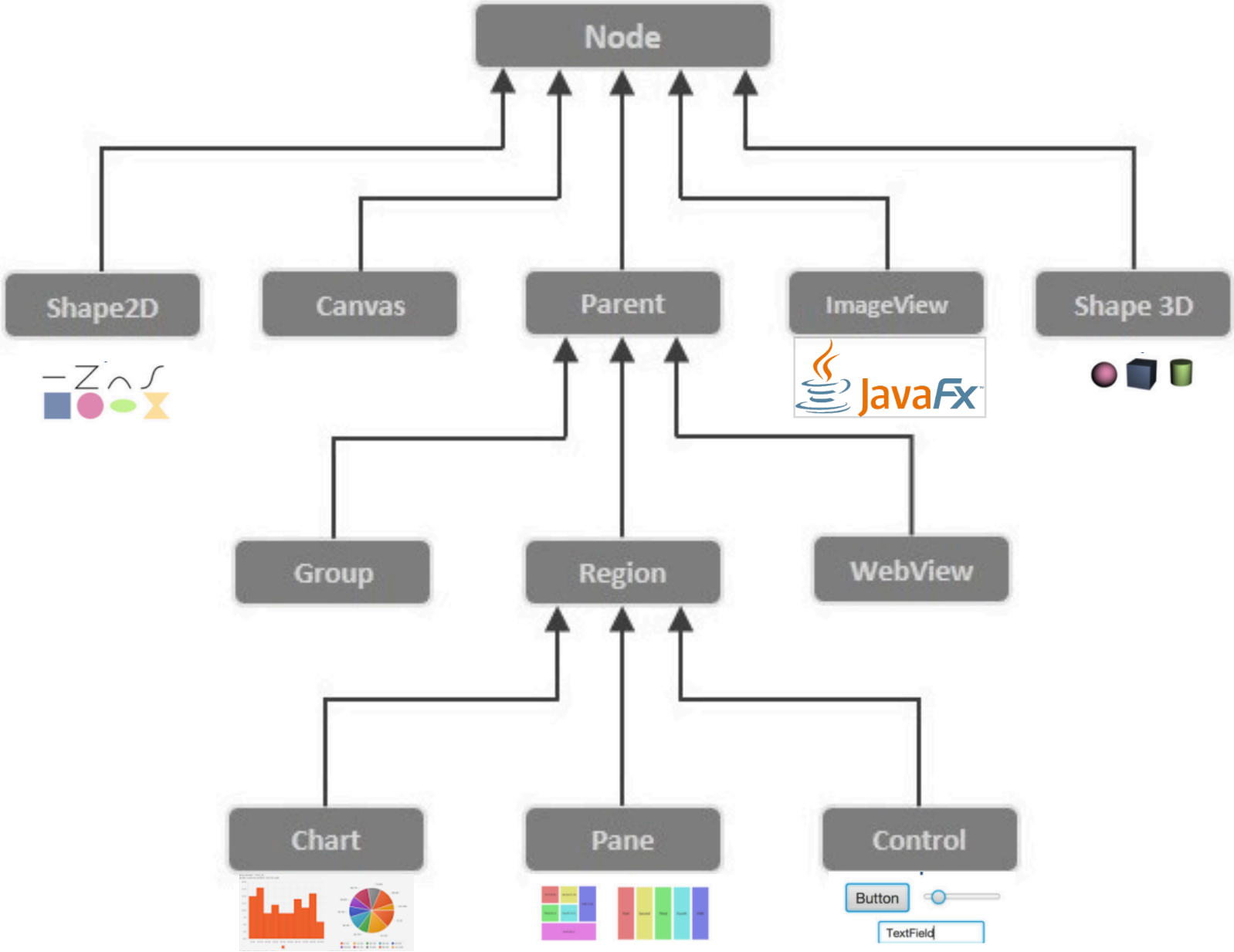
Week 5:
Images & Graphics

Anastasia Bezerianos

introduction.prog.is@gmail.com

3

UI Hierarchy



images

5

Run configurations > (x)=… under VM arguments put
`--add-modules javafx.controls,javafx.fxml &` untick "Use the -XstartOnFirstThread

JavaFX Images

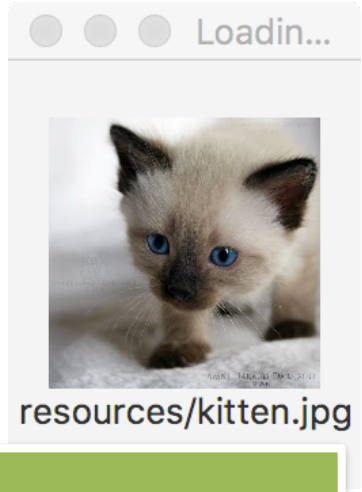
resources/kitten.jpg

In JavaFX we can load and use Images as any other Node in a layout (*)

```
public class DrawingSimpleImage extends Application {
      …
      public void start(Stage primaryStage) throws Exception {
            String path = "resources/kitten.jpg"; // relative path
            Image image = new Image(new FileInputStream(path));

            //Setting the image view
            ImageView imageView = new ImageView(image);

            //listen for mouse click events
            imageView.setOnMouseClicked(e->{
                System.out.println("Clicked on " + path);
              });

            //Creating a layout
            FlowPane root = new FlowPane();
            root.getChildren().add(imageView);  //imageView added as any node
      }
      …
}
```

(*) but also need to store images somewhere, example make a directory resources in your project

JavaFX Images



In JavaFX we can load and us
as any other Node in a la

Java stuff:
- Throw **Exception**, in case there is an error when loading the file
- FileInputStream, a class for reading binary files

```
public class DrawingSimpleImage extends Application {
    …
    public void start(Stage primaryStage) throws Exception {
        String path = "resources/kitten.jpg"; // relative path
        Image image = new Image(new FileInputStream(path));

        //Setting the image view
        ImageView imageView = new ImageView(image);

        //listen for mouse click events
        imageView.setOnMouseClicked(e->{
            System.out.println("Clicked on " + path);
          });

        //Creating a layout
        FlowPane root = new FlowPane();
        root.getChildren().add(imageView);  //imageView added as any node
    }
    …
}
```

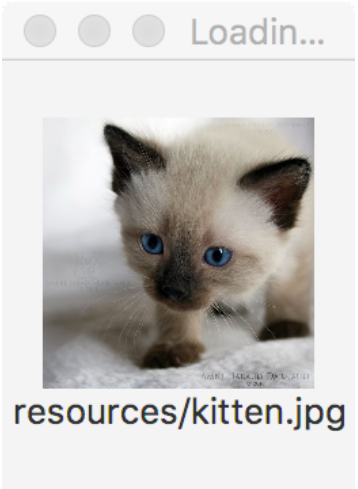
In windows "resources\kitten.jpg"

Image class stores the image (binary)
ImageView is the UI Node that displays it

ImageView is a Node, so we can add handlers/listeners to it

(*) also need to store images somewhere, example make a directory resources in your project

JavaFX Images



In JavaFX we can load and use Images as any other Node in a layout (*)

```
public class DrawingSimpleImage extends Application {
    …
    public void
            String
            Image

            //Sett
            ImageV

            //list
            imageView.setOnMouseClicked(e->{
                System.out.println("Clicked on " + path);
              });

            //Creating a layout
            FlowPane root = new FlowPane();
            root.getChildren().add(imageView);  //imageView added as any node
    }
    …
}
```

Image is loaded and listens to user input !!

… But the size needs fixing

(*) also need to store images somewhere, example make a directory resources in your project

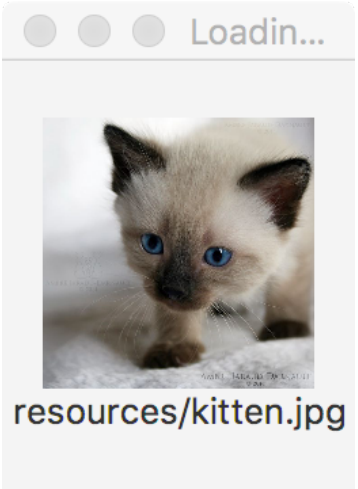
```java
public class DrawingImages extends Application {

    @Override
    public void start(Stage primaryStage) throws Exception {

            String path = "resources/kitten.jpg"; // if in windows fix path
            Image image = new Image(new FileInputStream(path));

            //Setting the image view
            ImageView imageView = new ImageView(image);
            //Setting the starting position of the image, may be overwritten
            imageView.setX(50);
            imageView.setY(25);
            //fit the image view width to this number of pixels
            imageView.setFitWidth(300);
            //Preserve width & height ratio of the image in the image view
            imageView.setPreserveRatio(true);

            //listen for mouse click events
            imageView.setOnMouseClicked(e->{
                System.out.println("Clicked on " + path);
                imageView.setRotate(imageView.getRotate()+30);
              });

            //Creating a layout
            FlowPane root = new FlowPane();
            root.setAlignment(Pos.CENTER);
            root.getChildren().add(imageView);  //imageView added as any node
            root.getChildren().add(new Label(path));
            Scene scene = new Scene(root, 300, 400);
            primaryStage.setTitle("Loading an image");
            primaryStage.setScene(scene);
            primaryStage.show();
        }
        …
}
```

9

```
public class DrawingImages extends Application {

    @Override
    public void start(Stage primaryStage) throws Exception {

            String path = "resources/kitten.jpg"; // if in windows fix path
            Image image = new Image(new FileInputStream(path));

            //Setting the image view
            ImageView imageView = new ImageView(image);
            //Setting the starting position of the image, may be overwritten
            imageView.setX(50);
            imageView.setY(25);
            //fit the image view width to this number of pixels
            imageView.setFitWidth(300);
            //Preserve width & height ratio of the image in the image view
            imageView.setPreserveRatio(true);

            //listen for mouse click events
            imageView.setOnMouseClicked(e->{
                System.out.println("Clicked on " + path);
                imageView.setRotate(imageView.getRotate()+30);
              });

            //Creating a layout
            FlowPane root = new FlowPane();
            root.setAlignment(Pos.CENTER);
            root.getChildren().add(imageView);  //imageView added as any node
            root.getChildren().add(new Label(path));
            Scene scene = new Scene(root, 300, 400);
            primaryStage.setTitle("Loading an image");
            primaryStage.setScene(scene);
            primaryStage.show();
        }
        …
}
```

What happens if I don't use setFitWidth()?

```java
public class DrawingImages extends Application {

    @Override
    public void start(Stage primaryStage) throws Exception {

            String path = "resources/kitten.jpg"; // if in windows fix path
            Image image = new Image(new FileInputStream(path));

            //Setting the image view
            ImageView imageView = new ImageView(image);
            //Setting the starting position of the image, may be overwritten
            imageView.setX(50);
            imageView.setY(25);
            //fit the image view width to this number of pixels
            imageView.setFitWidth(300);
            //Preserve width & height ratio of the image in the image view
            imageView.setPreserveRatio(true);

            //listen for mouse click events
            imageView.setOnMouseClicked(e->{
                System.out.println("Clicked on " + path);
                imageView.setRotate(imageView.getRotate()+30);
              });

            //Creating a layout
            FlowPane root = new FlowPane();
            root.setAlignment(Pos.CENTER);
            root.getChildren().add(imageView);  //imageView added as any node
            root.getChildren().add(new Label(path));
            Scene scene = new Scene(root, 300, 400);
            primaryStage.setTitle("Loading an image");
            primaryStage.setScene(scene);
            primaryStage.show();
        }
        …
}
```



Click now does something more interesting !!!

JavaFX Image classes

Image class represents graphical image files, used for loading images from a specified URL (including online) and accepts several formats (bmp, gif, jpeg, png, svg*)

ImageView class is a Node that displays an Image. It can be used as any other Node in a layout, listen to events, etc.

You can change the Image of the ImageView
More than one ImageView's can show one Image

(*) svg is a bit more complex, see https://edencoding.com/svg-javafx/

12

```
public class TwoImages extends Application {

     @Override
     public void start(Stage primaryStage) throws Exception {
              //Creating a layout
                FlowPane root = new FlowPane();
                root.setAlignment(Pos.CENTER);
                Scene scene = new Scene(root, 100, 200);

                //Creating an image
                String path = "resources/kitten.jpg"; // if in windows fix path
                Image image = new Image(new FileInputStream(path));


                ImageView imageView1 = new ImageView(image);
                imageView1.setOnMouseClicked(e->{
                    System.out.println("Clicked on " + path);
                    imageView1.setRotate(imageView1.getRotate()+30);
                  });

                ImageView imageView2 = new ImageView(image);
                imageView2.setOnMouseClicked(e->{
                    System.out.println("Clicked on " + path);
                    imageView2.setRotate(imageView2.getRotate()-30);
                  });

                root.getChildren().addAll(imageView1,imageView2);

                //Setting scene
                primaryStage.setTitle("Loading an image");
                primaryStage.setScene(scene);
                primaryStage.show();
            }
            …
}
```

```
public class TwoImages extends Application {

    @Override
    public void start(Stage primaryStage) throws Exception {
            //Creating a layout
              FlowPane root = new FlowPane();
              root.setAlignment(Pos.CENTER);
              Scene scene = new Scene(root, 100, 200);

              //Creating an image
              String path = "resources/kitten.jpg"; // if in windows fix path
              Image image = new Image(new FileInputStream(path));


              ImageView imageView1 = new ImageView(image);
              imageView1.setOnMouseClicked(e->{
                  System.out.println("Clicked on " + path);
                  imageView1.setRotate(imageView1.getRotate()+30);
                });

              ImageView imageView2 = new ImageView(image);
              imageView2.setOnMouseClicked(e->{
                  System.out.println("Clicked on " + path);
                  imageView2.setRotate(imageView2.getRotate()-30);
                });

              root.getChildren().addAll(imageView1,imageView2);

              //Setting scene
              primaryStage.setTitle("Loading an image");
              primaryStage.setScene(scene);
              primaryStage.show();
          }
          …
}
```

Same image used in two different ImageViews. Each ImageView reacts differently.

```
public class AnotherTwoImages extends Application {

    @Override
    public void start(Stage primaryStage) throws Exception {

            //Creating a layout
              FlowPane root = new FlowPane();
              root.setAlignment(Pos.CENTER);
              Scene scene = new Scene(root, 100, 200);

              //Creating images
              Image image1 = new Image(new FileInputStream("resources/kitten.jpg"));
              Image image2 = new Image(new FileInputStream("resources/puppy.jpg"));

              ImageView imageView1 = new ImageView(image1);
              imageView1.setOnMouseClicked(e->{
                  imageView1.setRotate(imageView1.getRotate()+30);
                });

              ImageView imageView2 = new ImageView(image1);
              imageView2.setOnMouseClicked(e->{
               if (imageView2.getImage() == image1)
                    imageView2.setImage(image2);
               else
                  imageView2.setImage(image1);
                });

              root.getChildren().addAll(imageView1,imageView2);

              //Setting scene
              primaryStage.setTitle("Loading an image");
              primaryStage.setScene(scene);
              primaryStage.show();
    }
          …
}
```

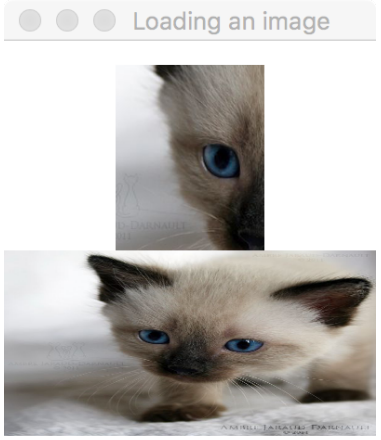
```
public class AnotherTwoImages extends Application {

    @Override
    public void start(Stage primaryStage) throws Exception {

            //Creating a layout
              FlowPane root = new FlowPane();
              root.setAlignment(Pos.CENTER);
              Scene scene = new Scene(root, 100, 200);

              //Creating images
              Image image1 = new Image(new FileInputStream("resources/kitten.jpg"));
              Image image2 = new Image(new FileInputStream("resources/puppy.jpg"));

              ImageView imageView1 = new ImageView(image1);
              imageView1.setOnMouseClicked(e->{
                  imageView1.setRotate(imageView1.getRotate()+30);
                });

              ImageView imageView2 = new ImageView(image1);
              imageView2.setOnMouseClicked(e->{
               if (imageView2.getImage() == image1)
                    imageView2.setImage(image2);
               else
                  imageView2.setImage(image1);
                });

              root.getChildren().addAll(imageView1,imageView2);

              //Setting scene
              primaryStage.setTitle("Loading an image");
              primaryStage.setScene(scene);
              primaryStage.show();
    }
          …
}
```

Loading images in 1 line

Changing images in an ImageView, triggered by user clicks

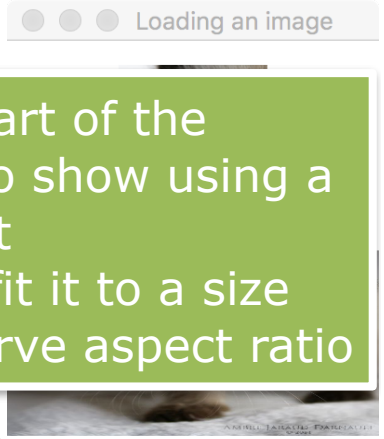
JavaFX Image classes

ImageView class has many properties for dealing with positioning and resizing



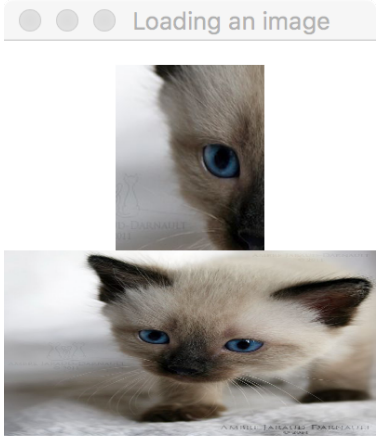

```
public class ResizingImages extends Application {

    @Override
    public void start(Stage primaryStage) throws Exception {
            FlowPane root = new FlowPane();
            root.setAlignment(Pos.CENTER);
            Scene scene = new Scene(root, 100, 200);

            //Creating an image
             String path = "resources/kitten.jpg"; // if in windows fix path
            Image image = new Image(new FileInputStream(path));

            // in this ImageView zooming into one area of the image
            Rectangle2D viewPort = new Rectangle2D(30, 30, 120, 80); // what part to pick
            ImageView imageView1 = new ImageView(image);
            imageView1.setViewport(viewPort); // only seeing the part defined in viewport
            imageView1.setFitWidth(100); // how big to show it in width
            imageView1.setFitHeight(100); // how big to show it in height
            imageView1.setPreserveRatio(true); // this preserves ratio no matter the fit
            root.getChildren().add(imageView1);

            // in this ImageView change the resizing behavior
            ImageView imageView2 = new ImageView(image);
            imageView2.setFitHeight(100); // height always the same
            imageView2.fitWidthProperty().bind(scene.widthProperty());
                // bind width size to that of the scene width

            root.getChildren().add(imageView2);

            //Setting scene
            primaryStage.setTitle("Loading an image");
            primaryStage.setScene(scene);
            primaryStage.show();
        }
        …
}
```

```
public class ResizingImages extends Application {

    @Override
    public void start(Stage primaryStage) throws Exception {
            FlowPane root = new FlowPane();
            root.setAlignment(Pos.CENTER);
            Scene scene = new Scene(root, 100, 200);

            //Creating an image
             String path = "resources/kitten.jpg"; // if in windows fix path
            Image image = new Image(new FileInputStream(path));

            // in this ImageView zooming into one area of the image
            Rectangle2D viewPort = new Rectangle2D(30, 30, 120, 80); // what part to pick
            ImageView imageView1 = new ImageView(image);
            imageView1.setViewport(viewPort); // only seeing the part defined in viewport
            imageView1.setFitWidth(100); // how big to show it in width
            imageView1.setFitHeight(100); // how big to show it in height
            imageView1.setPreserveRatio(true); // this preserves ratio no matter the fit
            root.getChildren().add(imageView1);

            // in this ImageView change the resizing behavior
            ImageView imageView2 = new ImageView(image);
            imageView2.setFitHeight(100); // height always the same
            imageView2.fitWidthProperty().bind(scene.widthProperty());
                // bind width size to that of the scene width

            root.getChildren().add(imageView2);

            //Setting scene
            primaryStage.setTitle("Loading an image");
            primaryStage.setScene(scene);
            primaryStage.show();
        }
        …
}
```



select part of the image to show using a viewport
… then fit it to a size
… preserve aspect ratio

stretch the second image, by binding its width to window width




```
public class ResizingImages extends Application {

    @Override
    public void start(Stage primaryStage) throws Exception {
            FlowPane root = new FlowPane();
            root.setAlignment(Pos.CENTER);
            Scene scene = new Scene(root, 100, 200);

            //Creating an image
             String path = "resources/kitten.jpg"; // if in windows fix path
            Image image = new Image(new FileInputStream(path));

            // in this ImageView zooming into one area of the image
            Rectangle2D viewPort = new Rectangle2D(30, 30, 120, 80); // what part to pick
            ImageView imageView1 = new ImageView(image);
            imageView1.setViewport(viewPort); // only seeing the part defined in viewport
            imageView1.setFitWidth(100); // how big to show it in width
            imageView1.setFitHeight(100); // how big to show it in height
            imageView1.setPreserveRatio(true); // this preserves ratio no matter the fit
            root.getChildren().add(imageView1);

            // in this ImageView change the resizing behavior
            ImageView imageView2 = new ImageView(image);
            // imageView2.setFitHeight(100);
            imageView2.fitWidthProperty().bind(scene.widthProperty());
                // bind width size to that of the scene width
            imageView1.setPreserveRatio(true);
                // but preserve width/height ratio when stretching
            root.getChildren().add(imageView2);

            //Setting scene
            primaryStage.setTitle("Loading an image");
            primaryStage.setScene(scene);
            primaryStage.show();
        }
        …
}
```

```
public class ResizingImages extends Application {

    @Override
    public void start(Stage primaryStage) throws Exception {
            FlowPane root = new FlowPane();
            root.setAlignment(Pos.CENTER);
            Scene scene = new Scene(root, 100, 200);

            //Creating an image
             String path = "resources/kitten.jpg"; // if in windows fix path
            Image image = new Image(new FileInputStream(path));

            // in this ImageView zooming into one area of the image
            Rectangle2D viewPort = new Rectangle2D(30, 30, 120, 80); // what part to pick
            ImageView imageView1 = new ImageView(image);
            imageView1.setViewport(viewPort); // only seeing the part defined in viewport
            imageView1.setFitWidth(100); // how big to show it in width
            imageView1.setFitHeight(100); // how big to show it in height
            imageView1.setPreserveRatio(true); // this preserves ratio no matter the fit
            root.getChildren().add(imageView1);

            // in this ImageView change the resizing behavior
            ImageView imageView2 = new ImageView(image);
            // imageView2.setFitHeight(100);
            imageView2.fitWidthProperty().bind(scene.widthProperty());
                // bind width size to that of the scene width
            imageView1.setPreserveRatio(true);
                // but preserve width/height ratio when stretching
            root.getChildren().add(imageView2);

            //Setting scene
            primaryStage.setTitle("Loading an image");
            primaryStage.setScene(scene);
            primaryStage.show();
        }
        …
}
```

Second image now keeps ratio and stretches to best fit the window

21

There are many things to do with images

- Clipping images (Image.setClip(Shape))
- Transformations (ImageView.setRotate(), etc)
- Writing to images (PixelReader, PixelWriter, WritableImage)

geometric shapes

23

JavaFX Shapes

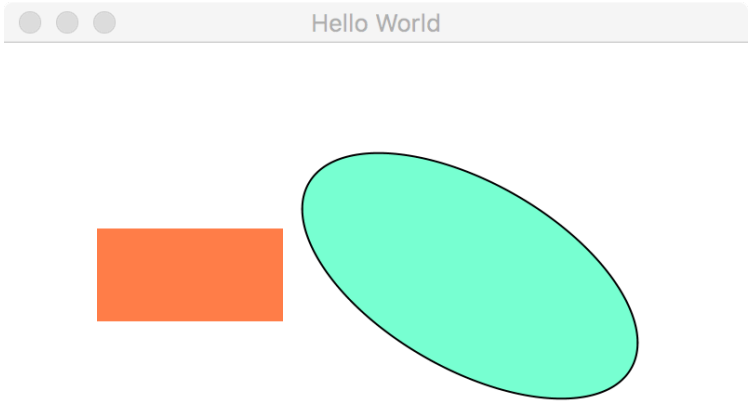
In the same way we use ImageView as a node,
we can also display basic graphical elements

JavaFX Shapes

Nodes like Rectangles, Circles, etc.

can be used and added as any other node in JavaFX

```
public class DrawingShapeNodes extends Application {
    …

    public void start(Stage primaryStage) {
        primaryStage.setTitle("Hello World");

        …

        Rectangle rec = new Rectangle(100,50);
        rec.setFill(Color.CORAL);
        rec.setOnMouseClicked(click_ev);

        …

    }
}
```

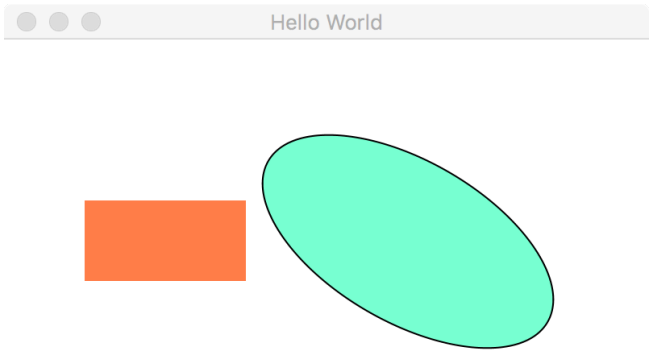


25

```
public class DrawingShapeNodes extends Application {
     …
    @Override
    public void start(Stage primaryStage) {
        primaryStage.setTitle("Hello World");

        EventHandler click_ev = new EventHandler<Event>() { … };
        EventHandler enter_ev = new EventHandler<Event>() { … };
        EventHandler exit_ev = new EventHandler<Event>() { … };

        FlowPane root = new FlowPane();
        root.setAlignment(Pos.CENTER);

        Rectangle rec = new Rectangle(100,50);
        rec.setFill(Color.CORAL);
        rec.setOnMouseClicked(click_ev);
        rec.setOnMouseEntered(enter_ev);
        rec.setOnMouseExited(exit_ev);

        Ellipse cir = new Ellipse(100,50);
        cir.setFill(Color.AQUAMARINE);
        cir.setRotate(30);
        cir.setOnMouseClicked(click_ev);
        cir.setOnMouseEntered(enter_ev);
        cir.setOnMouseExited(exit_ev);

        Scene scene = new Scene(root, 400, 250);
        root.getChildren().addAll(rec,cir);
        primaryStage.setScene(scene);
        primaryStage.show();
    }

}
```

More complex example with event listeners

… shared by all shapes

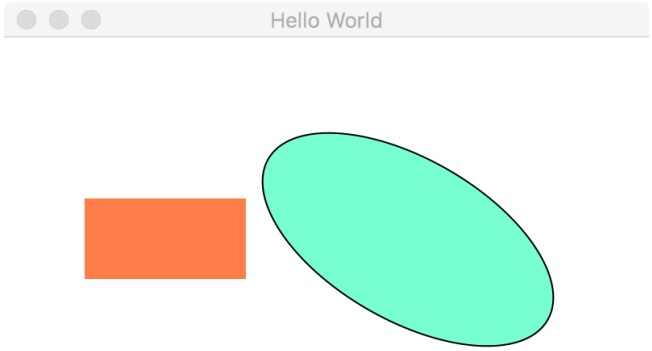


```
public class DrawingShapeNodes extends Application {
     …
    @Override
    public void start(Stage primaryStage) {
        primaryStage.setTitle("Hello World");

        EventHandler click_ev = new EventHandler<Event>() {
          @Override public void handle(Event e) {
                        Shape s = (Shape) (e.getSource())  ;
                        s.setRotate(s.getRotate()+30);
                    }
            };
        EventHandler enter_ev = new EventHandler<Event>() { … };
        EventHandler exit_ev = new EventHandler<Event>() { … };

       …
}
```

Even if your handler is not an anonymous inner class (like this example)

… you can still access the shape using the source of the event.

Remember to cast it to a Shape or another Node class (`Ellipse,Rectangle`)



Transformations

(Node / Shape) rotate, scale, translate, shear methods

```
Ellipse e = new Ellipse(100,50);
e.setRotate(30);
```

(Node / Shape) rotate, scale, translate, shear methods (affine transforms)

```
Rotate r = new Rotate(30, 100, 50);
// rotation angle, pivot x, pivot y
Ellipse e = new Ellipse(100,50);
e.getTransforms.addAll(r);
```

https://docs.oracle.com/javase/8/javafx/api/javafx/scene/transform/Affine.html

JavaFX Shapes

This approach to drawing shapes allows us to:

1. Use layouts to arrange shapes
2. Listen for input events happening on these objects (mouse clicked, hover, enter/leave, etc.)

JavaFX Shapes limitations

But we may want more control

(e.g., moving, resizing objects,

allowing the user to draw free-form lines

…)

Canvas nodes help us with that

But before we describe the Canvas class,

a bit about drawing

drawing on canvas

JavaFX

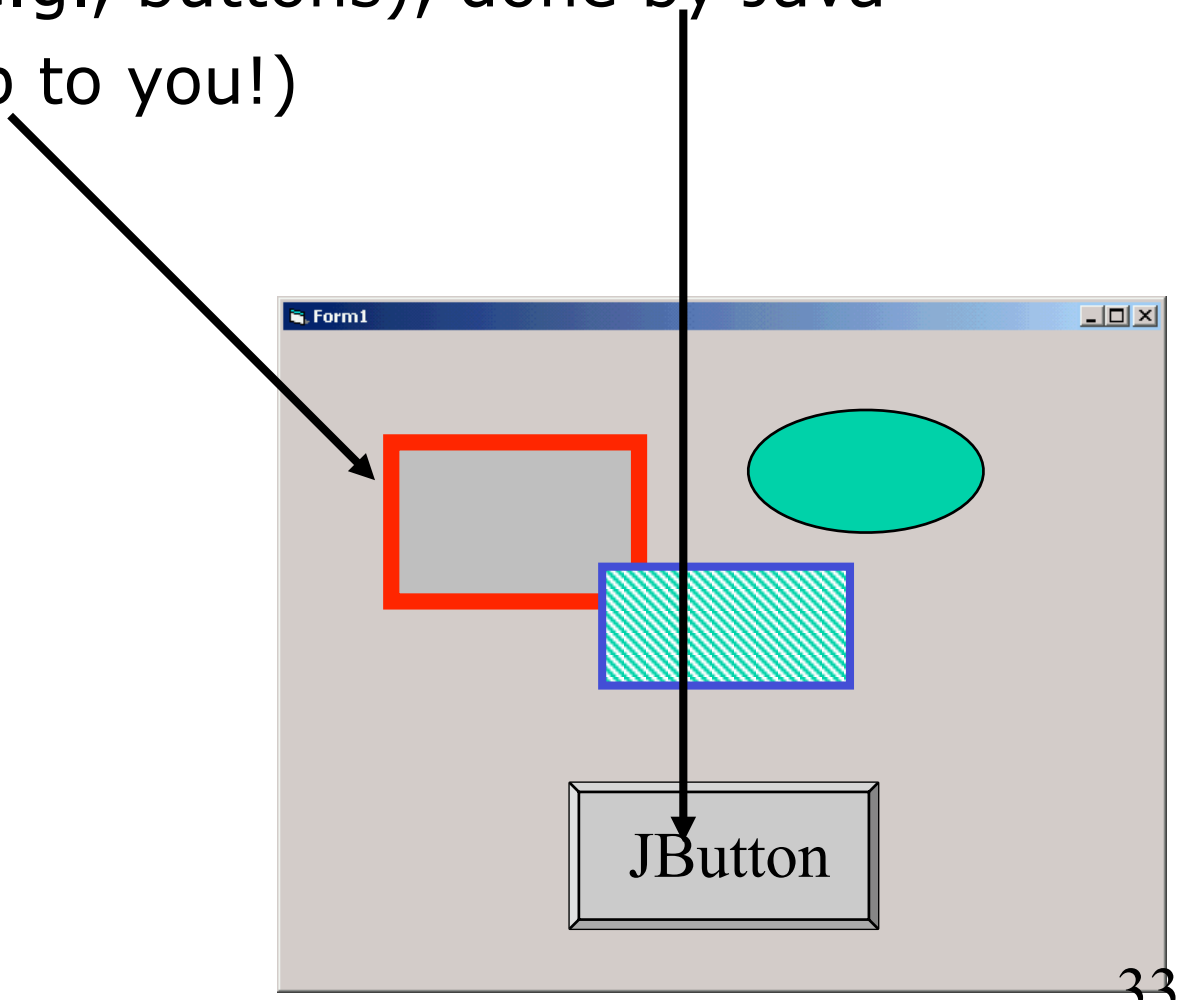
Provides 2D graphics, text & image capabilities

- Wide range of geometric primitives
- Mechanisms for hit-detection of shapes, text, images
- Color & transparency
- Transformations
- Printing
- Control of the quality of rendering

Designing components

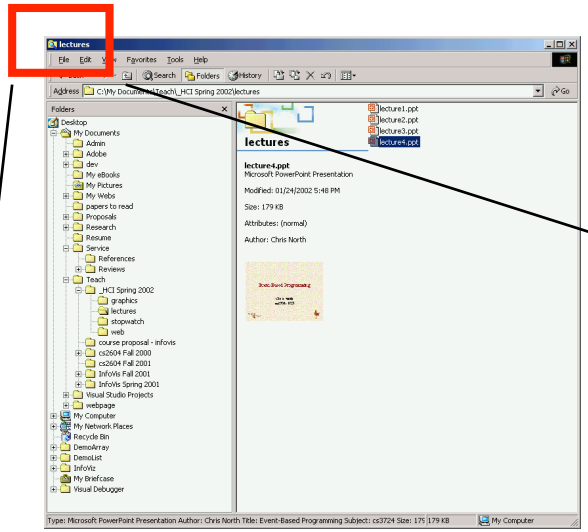
A window is a canvas on which applications draw:

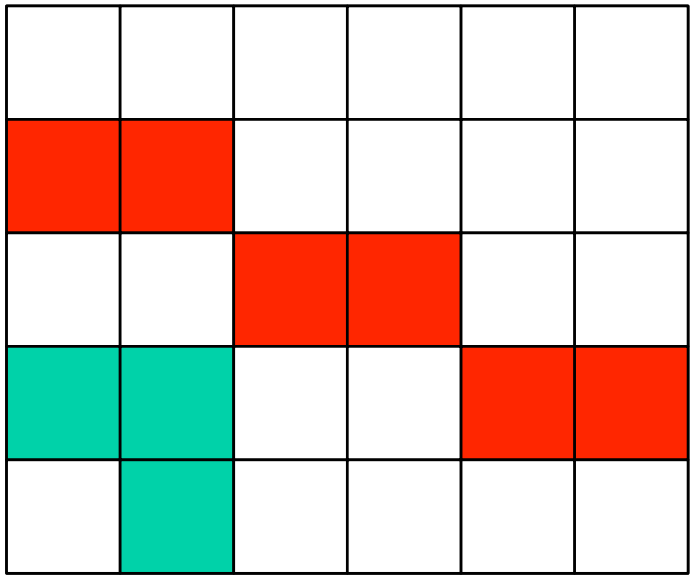
- API components (e.g., buttons), done by Java
- The rest (that is up to you!)



33

Pixel = picture element

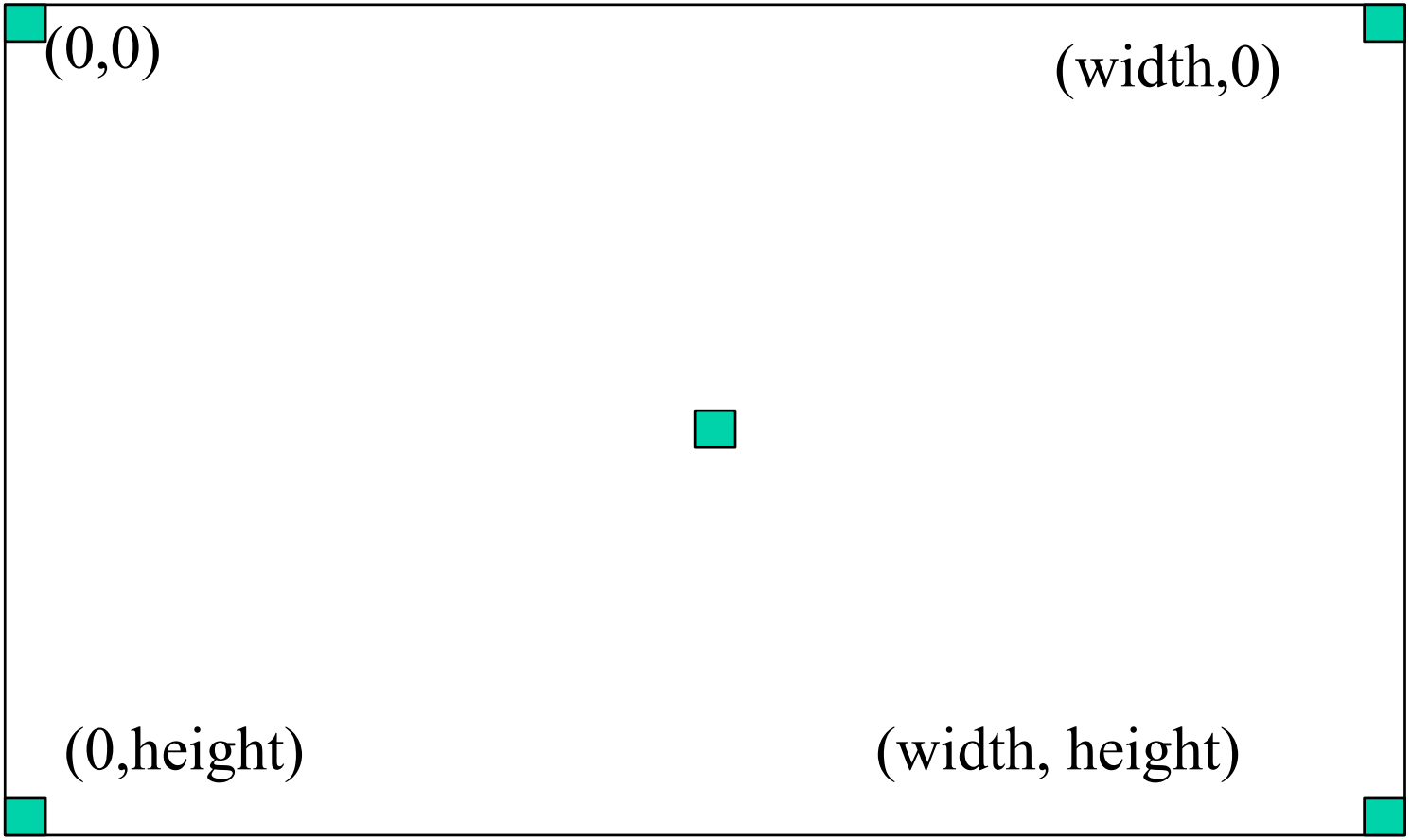




coordinate system

Almost cartesian :

(0,0) top left

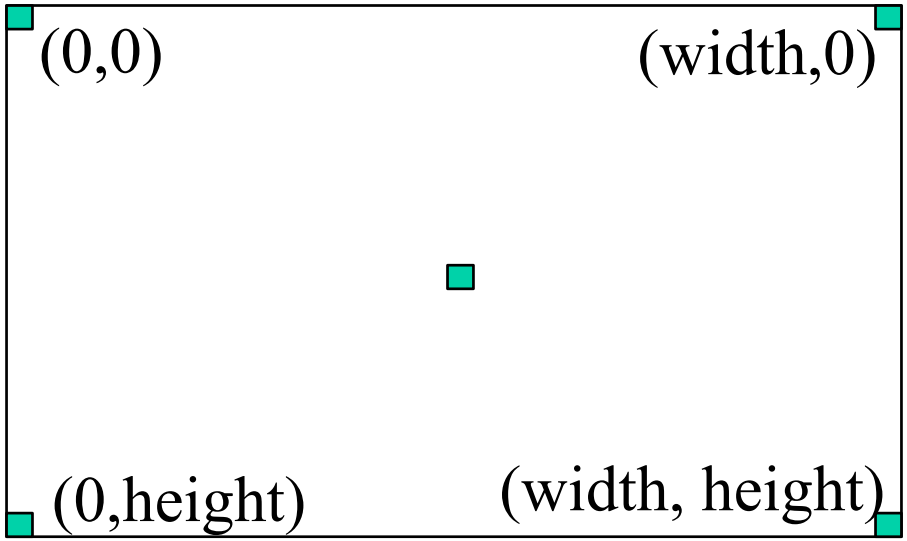


windows and subwindows

Each component has its own design space: its *subwindow*

Subwindow = the rectangular space inside the parent of the component, where the component is designed.

It has its own coordinate system



36

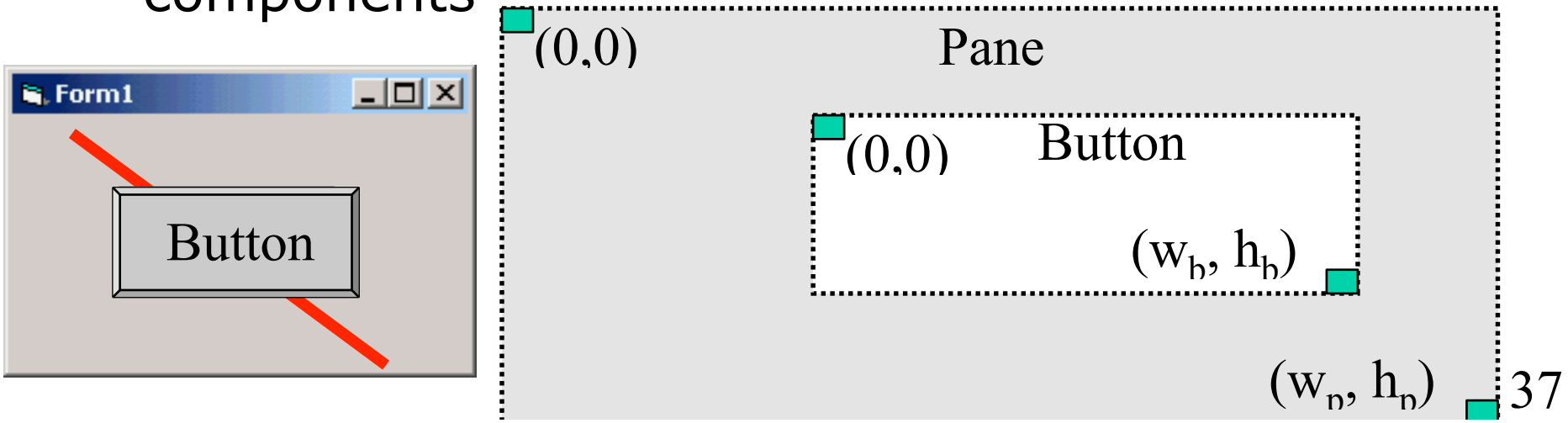
windows and subwindows

Each component has its own design space: its *subwindow*

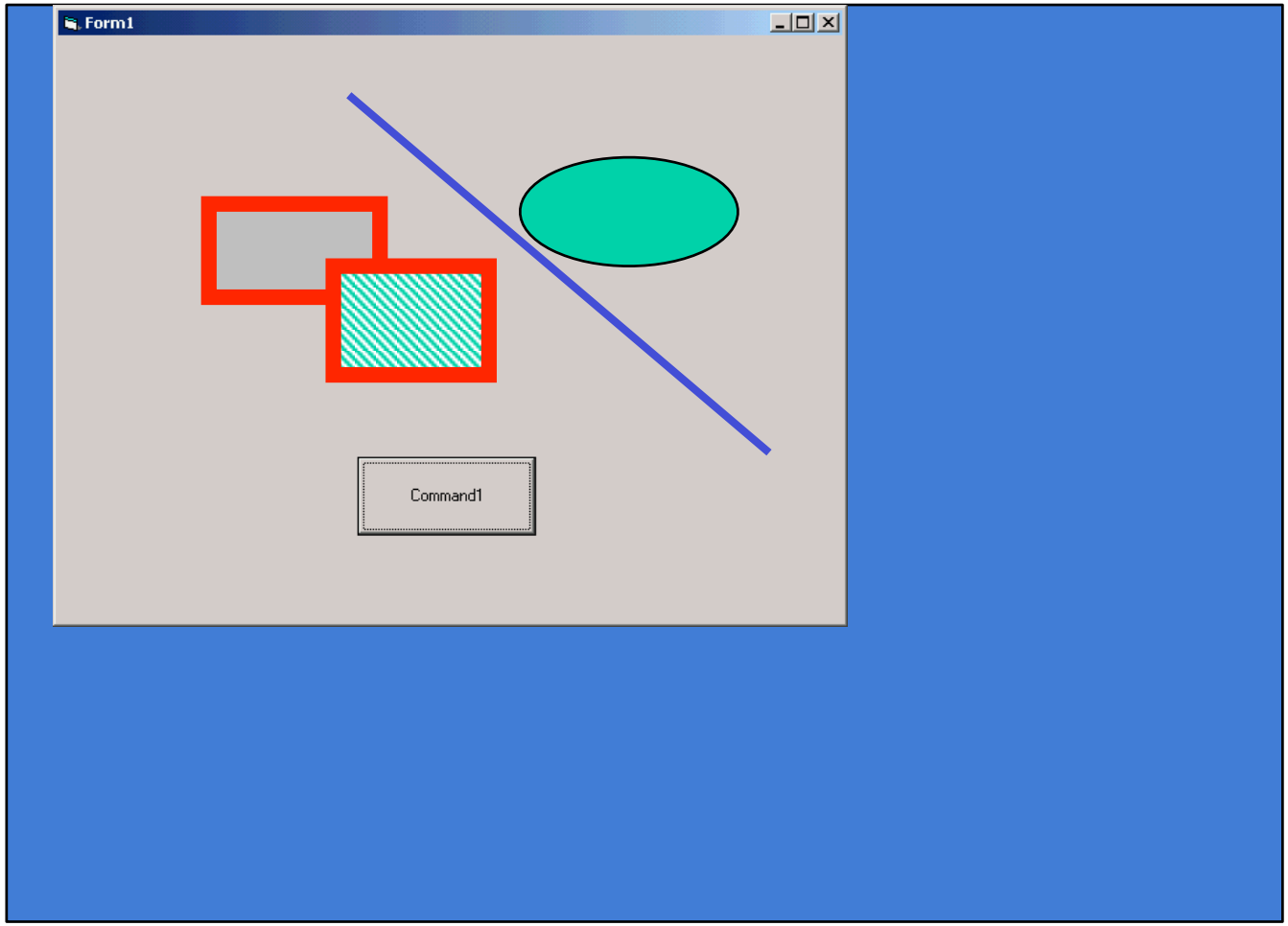
Subwindow = the rectangular space inside the parent of the component, where the component is designed. It has its own coordinate system

Clipping, rules: a component can not draw

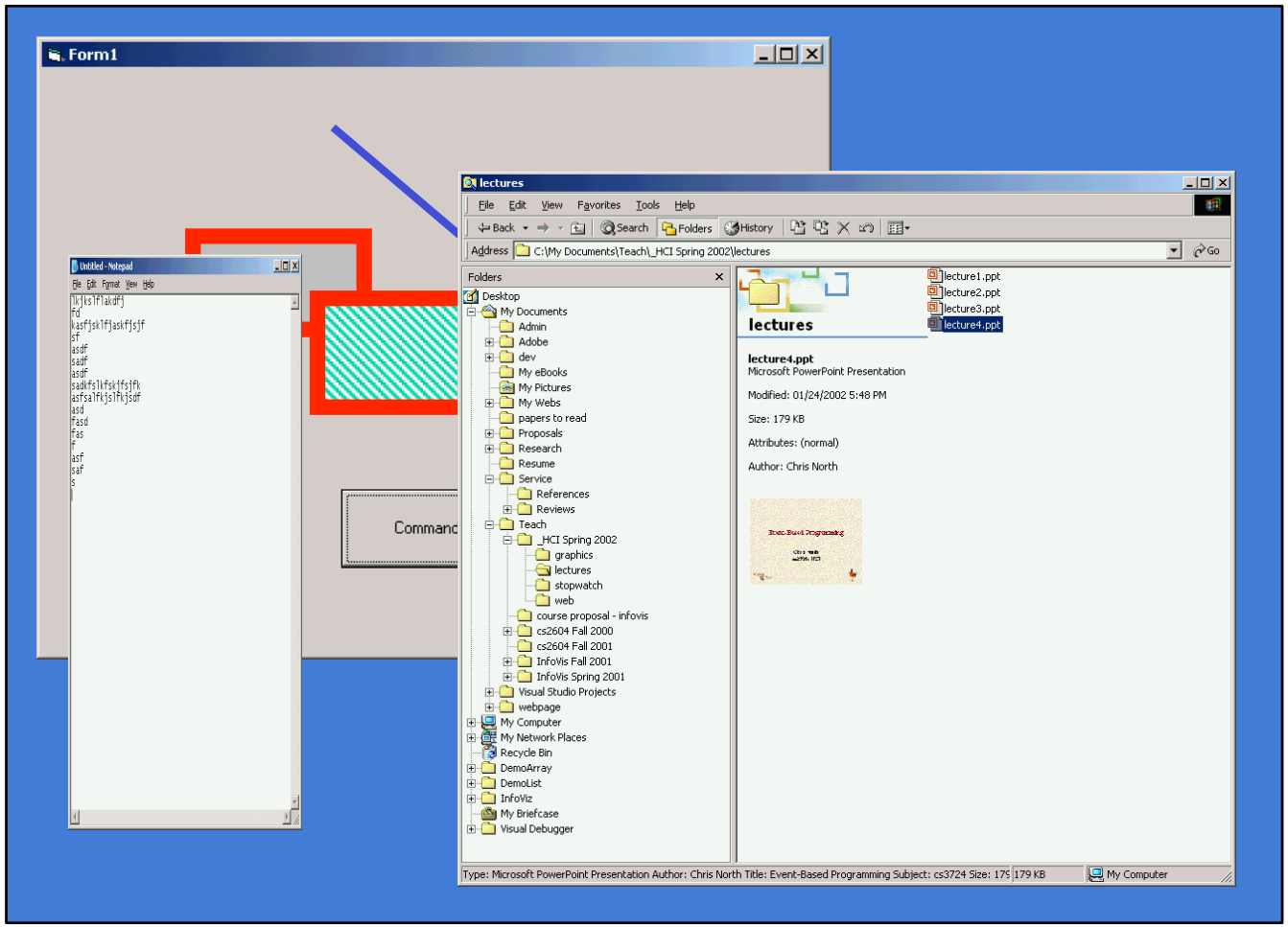
outside its subwindow, nor on one of its own components



a screen with an application

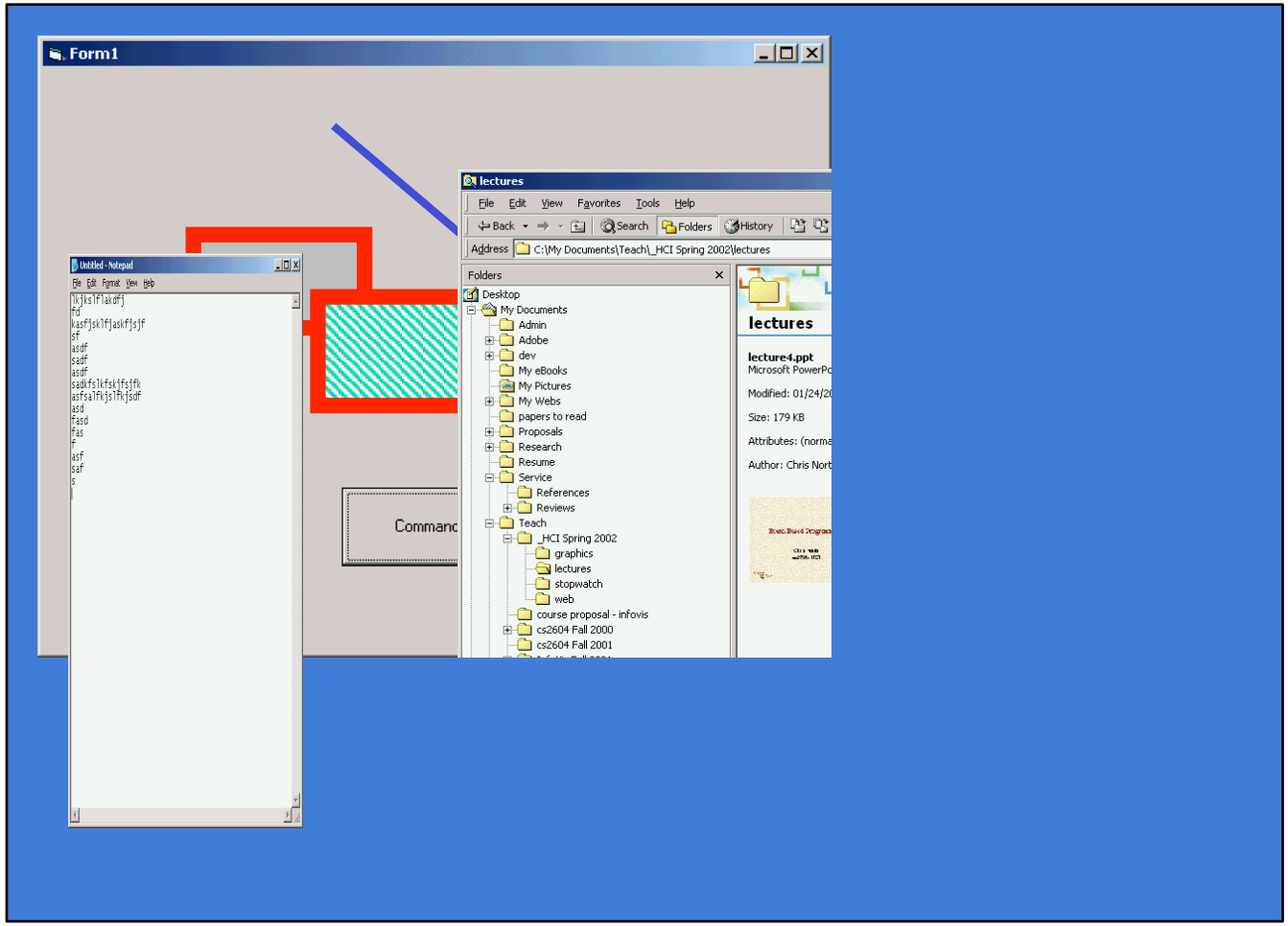


a screen with 3 applications



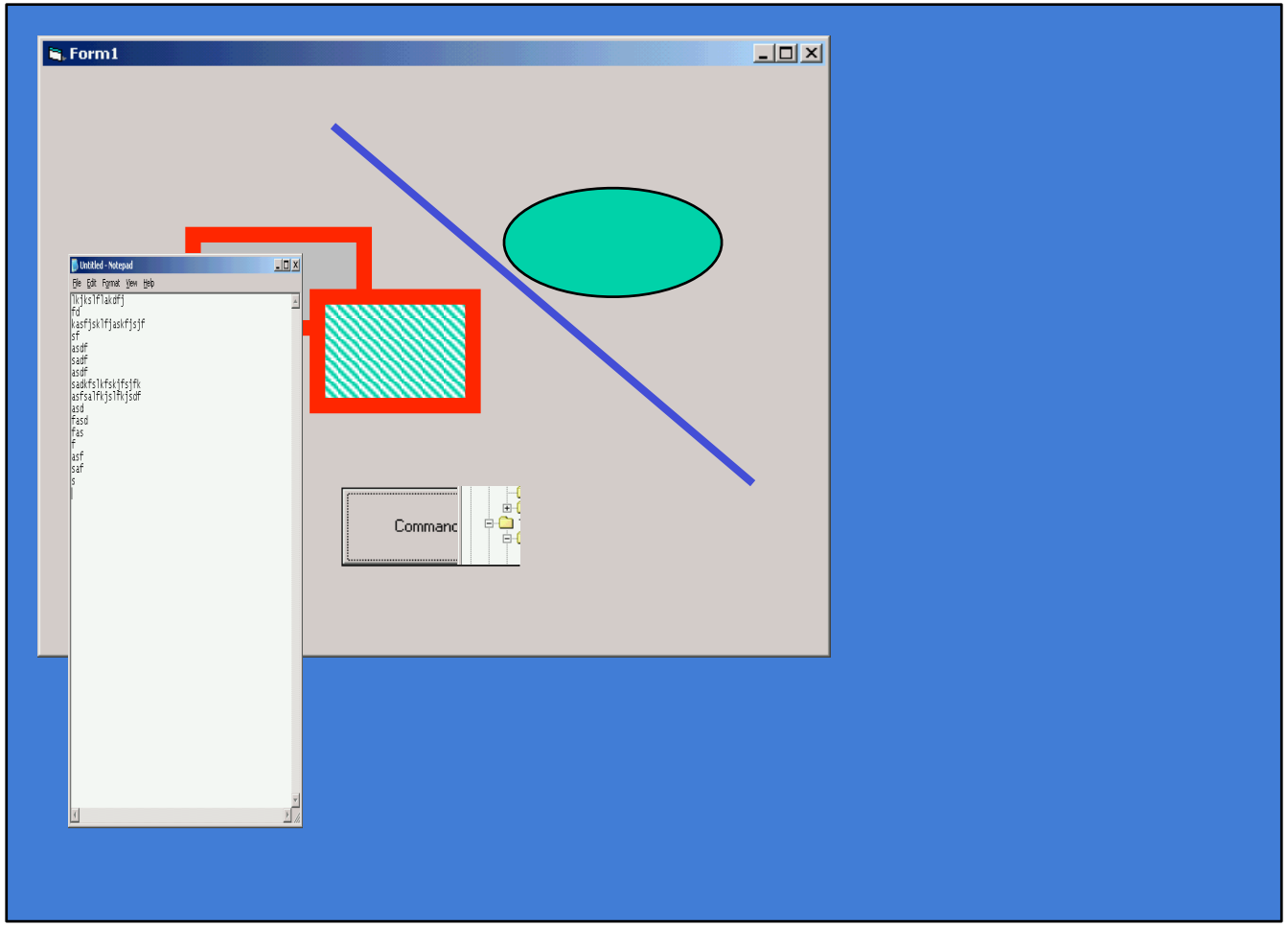
a screen, after closing one application -1

closing window notifies system of state, screen sends msg for repaint



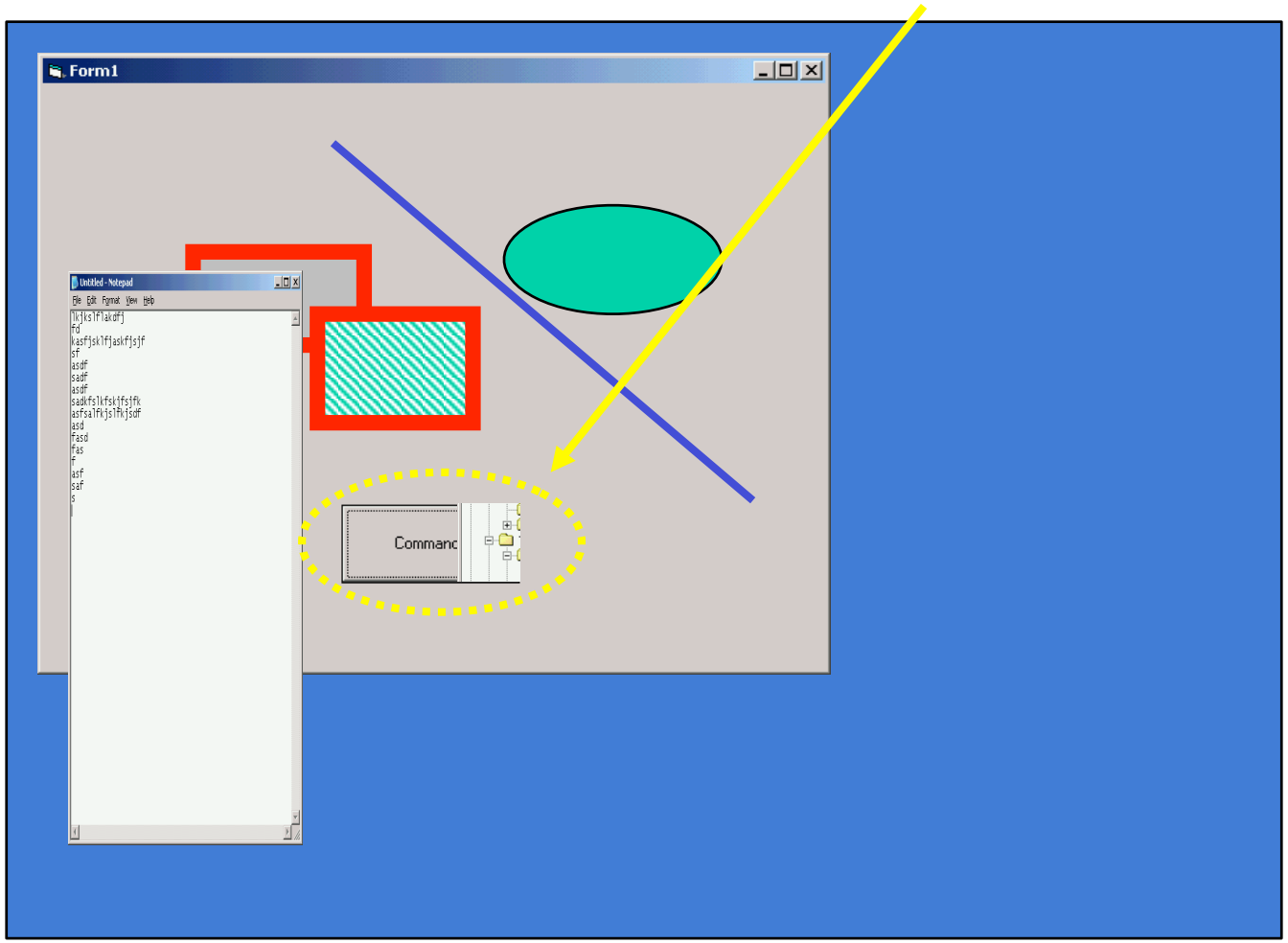
a screen, after closing one application - 2

remaining windows receive the message to repaint themselves



a screen, after closing one application - 3

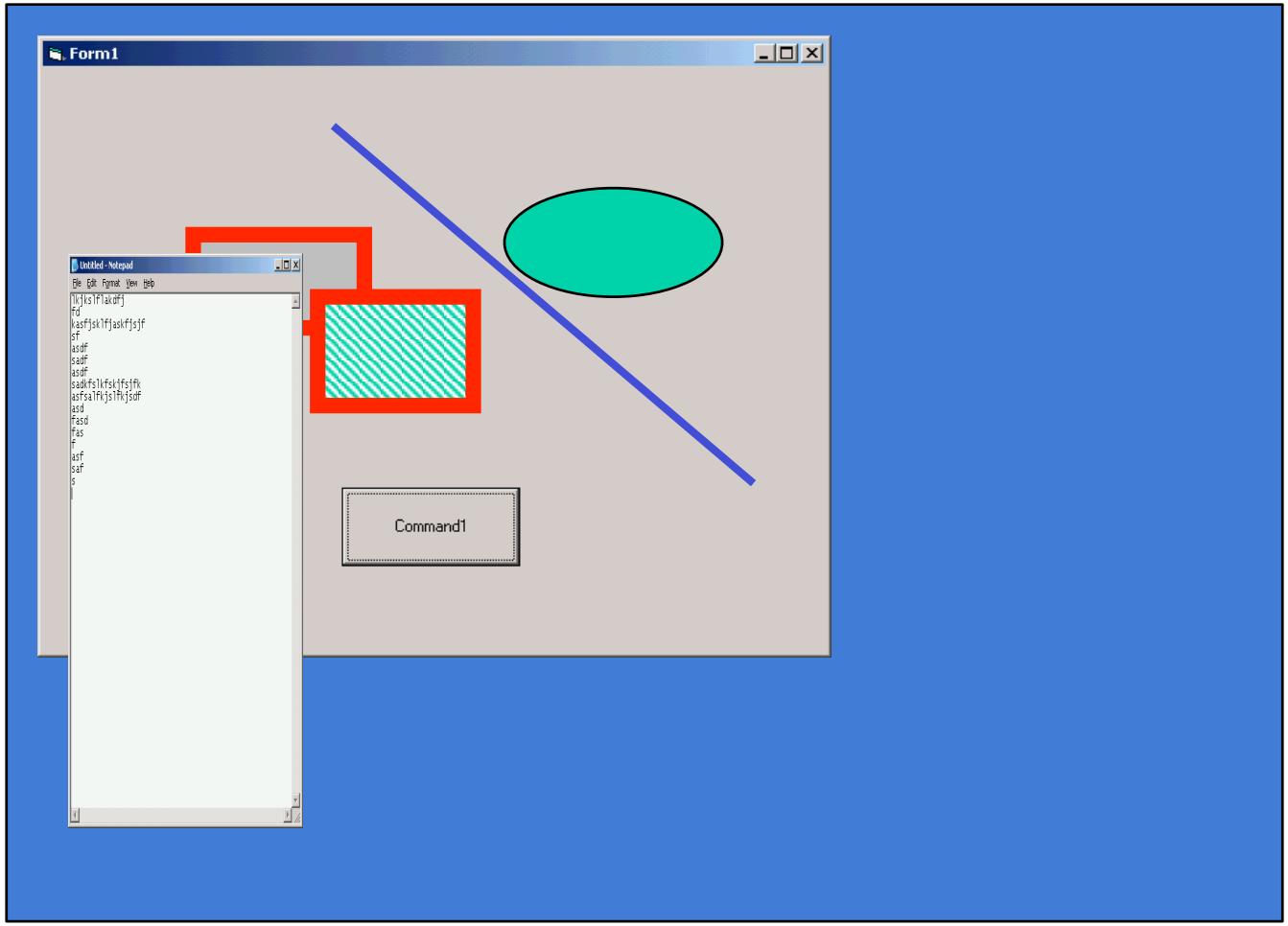
each window asks their components to repaint themselves



42

a screen, after closing one application - 4

all updated!



Canvas class

(a bit advanced, you can do a lot with what you have seen so far)

Canvas class

Canvas:

inherits from class *Node*

helps update and redraw our component

defined in *GraphicsContext*

stores drawing primitives

inherits methods (that we need to keep track of)

setFill(), setStroke(), setLineWidth(), …

strokeArc(), strokePolygon()

drawing

```
import javafx.scene.canvas.GraphicsContext;
```

1. get a hold of the “graphics context” of your component

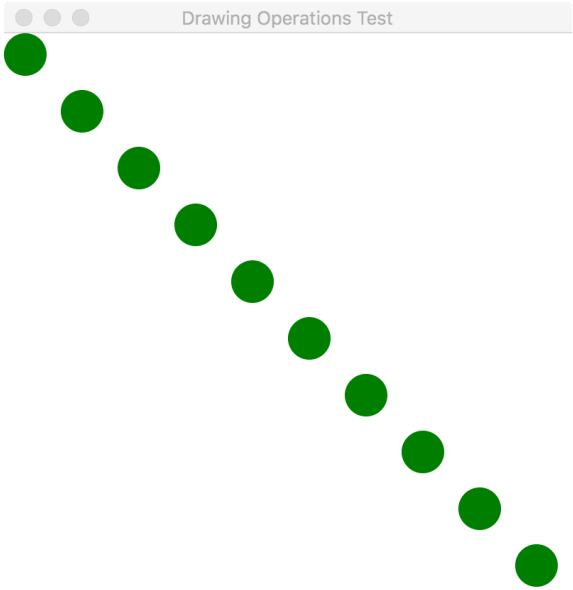
```
GraphicsContext gc = canvas.getGraphicsContext2D();
```

2. draw different shapes

```
gc.strokeOval(60, 60, 30, 30);
```

new component, an example

```
public class FirstCanvas extends Application {

    …

    @Override
    public void start(Stage primaryStage) throws Exception {
        primaryStage.setTitle("Drawing Operations Test”);

        Group root = new Group();
        Canvas canvas = new Canvas(400, 400);
        GraphicsContext gc = canvas.getGraphicsContext2D();
        drawShapes(gc);
        root.getChildren().add(canvas);
        primaryStage.setScene(new Scene(root));
        primaryStage.show();
    }

    void drawShapes(GraphicsContext gc) {
        gc.setFill(Color.GREEN);
        gc.setStroke(Color.BLUE);
        for(int i = 0; i <10;++i)
            gc.fillOval(i*40, i*40, 30, 30);
    }
}
```



GraphicsContext

An instance of *GraphicsContext* is provided by Java for components to draw on it

The Graphics object has a state, such as

Transformation, for example translation from the origin for rendering: translate() // (0,0) at top left by default

Color of design

```
Color col1 = new Color.rgb (255, 0, 0); // can have HSV
```

Character font

```
Font font1 = new Font("SansSerif", Font.BOLD, 12);
```

48

design functions in GraphicsContext

Example : `public void strokeLine (x1, y1, x2, y2)`

properties depend on current color (i.e., last defined color)

`fill*() / stroke*()`: draw filled shape or just contour

`*` = { `Rect, Oval, String, Arc, Polygon, PolyLine` }

Function `clear()` to remove all drawn shapes

Function `FontMetrics getFontMetrics()`

Returns an instance/object with info on the size of the text

Function `drawImage()` to draw an image

Needs an instance/object of class Image

shape examples with « draw » « stroke/fill »

strokeLine

strokePolyLine

(stroke/fill)Arc

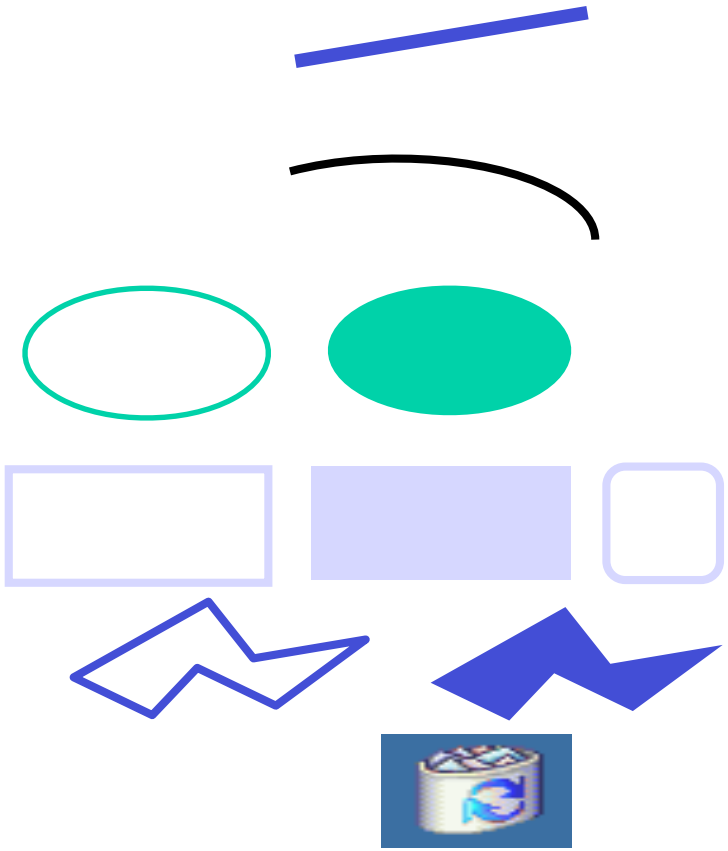
(stroke/fill)Oval

(stroke/fill)Rec or RoundRec

(stroke/fill)Polygon

drawImage

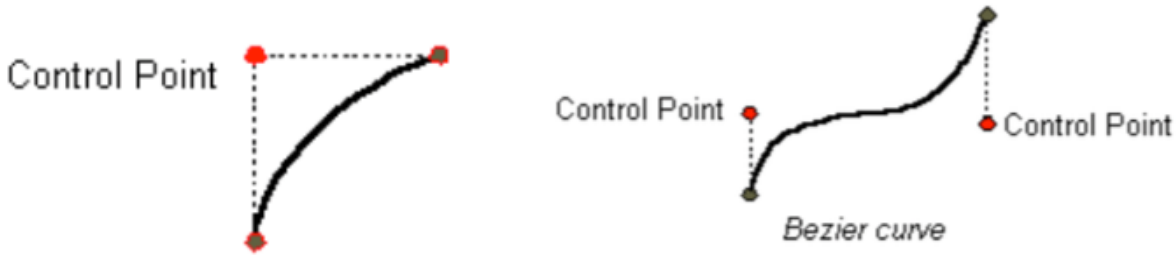
(stroke/fill)Text



https://docs.oracle.com/javase/8/javafx/api/javafx/scene/canvas/GraphicsContext.html 50

more examples with « stroke » and « fill »

Rect RoundRect Oval Arc



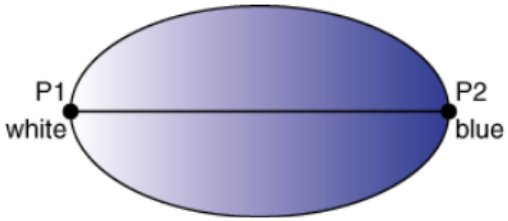
bezier curves

general paths

stroking and painting

stroke patterns

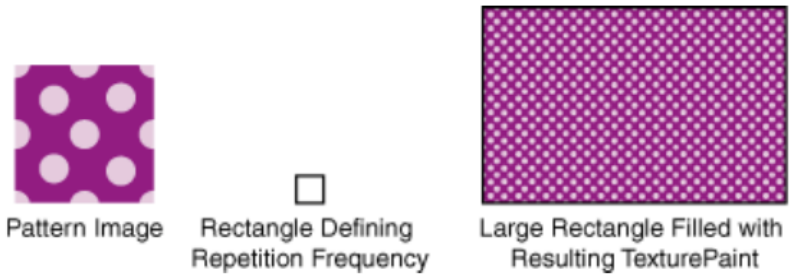
`getStrokeDashArray()`



gradient filling colors

`Paint.LinearGradient`

`Paint.RadialGradient`



filling patterns

`Paint.ImagePattern`

Transformations

(Node / Shape) rotate, scale, translate, shear methods

```
Ellipse e = new Ellipse(100,50);
e.setRotate(30);
```

(Node / Shape)rotate, scale, translate, shear methods (affine transforms)

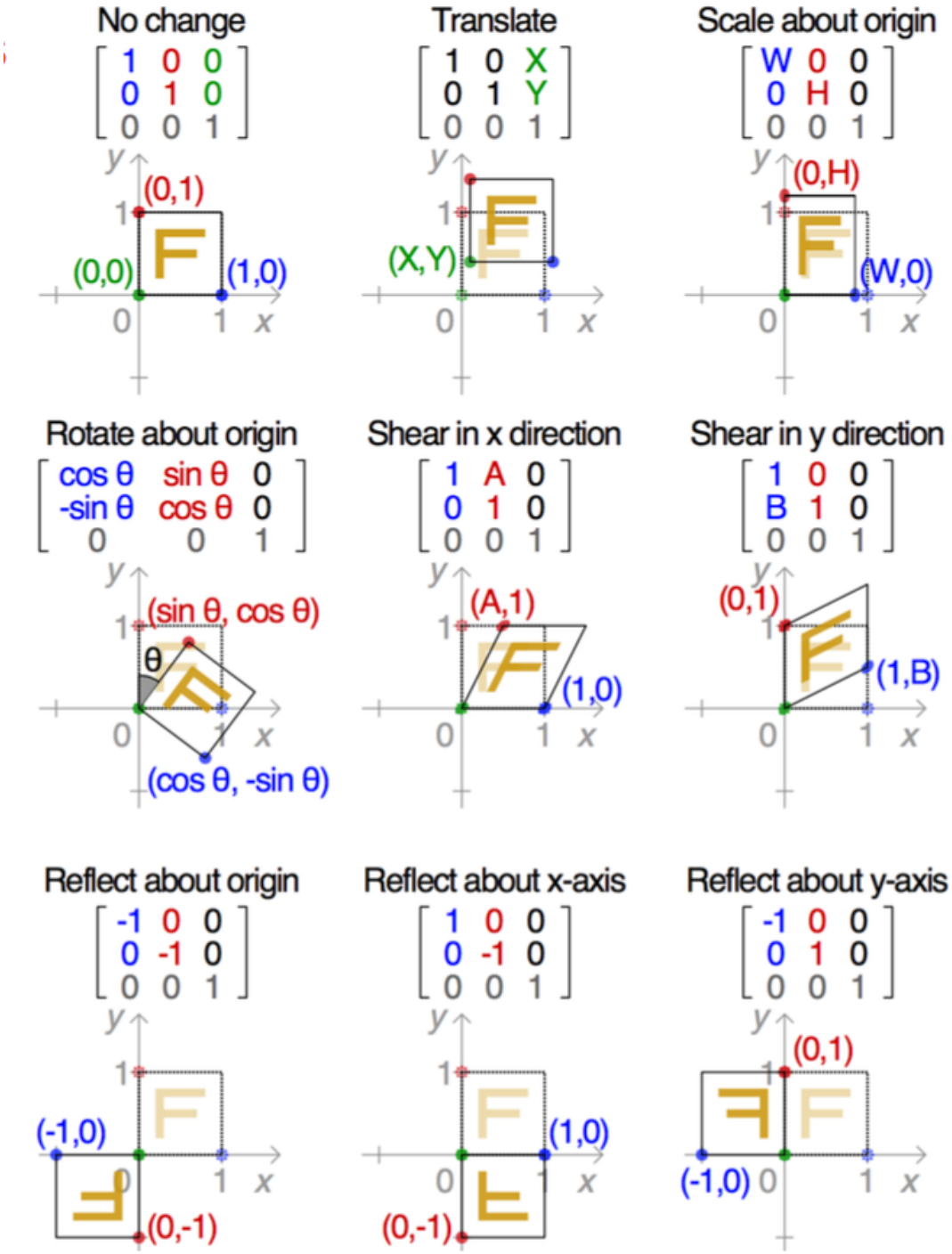
```
Rotate r = new Rotate(30, 100, 50);
// rotation angle, pivot x, pivot y
Ellipse e = new Ellipse(100,50);
e.getTransforms.addAll(r);
```

(Canvas) Affine matrices (also applicable to GraphicsContext)

```
Affine atransf = new Affine();
atransf.prependRotation(30); // rotate 90 degrees
gc.setTransform(atransf);
```

https://docs.oracle.com/javase/8/javafx/api/javafx/scene/transform/Affine.html

Affine Transformations



more on images
https://docs.oracle.com/javafx/2/image_ops/jfxpub-image_ops.htm

more on canvas
https://docs.oracle.com/javafx/2/canvas/jfxpub-canvas.htm

more on transformations
https://docs.oracle.com/javase/8/javafx/visual-effects-tutorial/transforms.htm
https://o7planning.org/en/11157/javafx-transformations-tutorial