

LAYOUT

CMPT 381



Overview

What is layout?

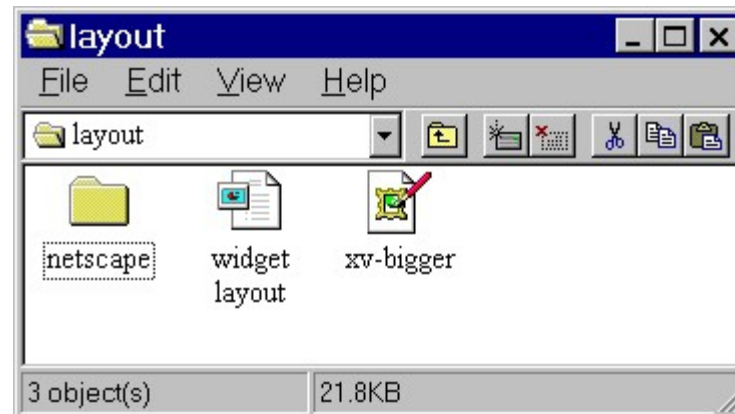
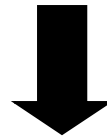
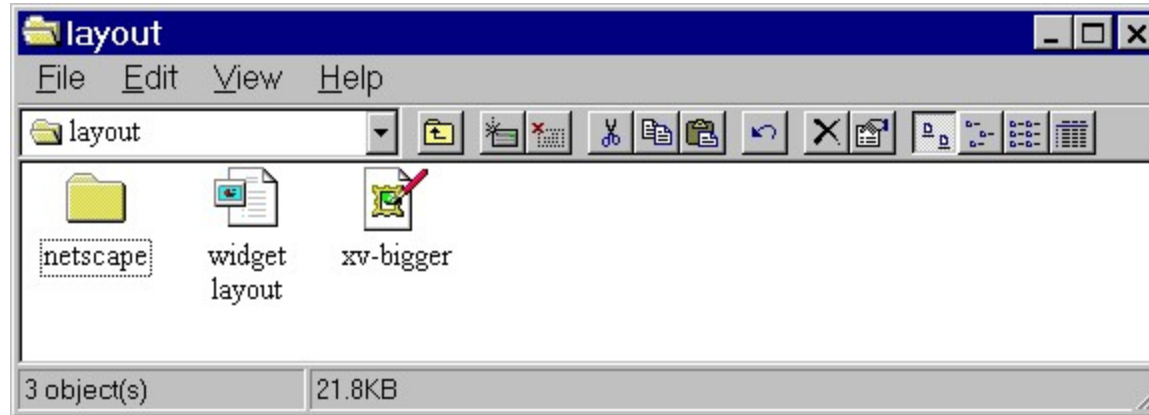
Examples

Layout models

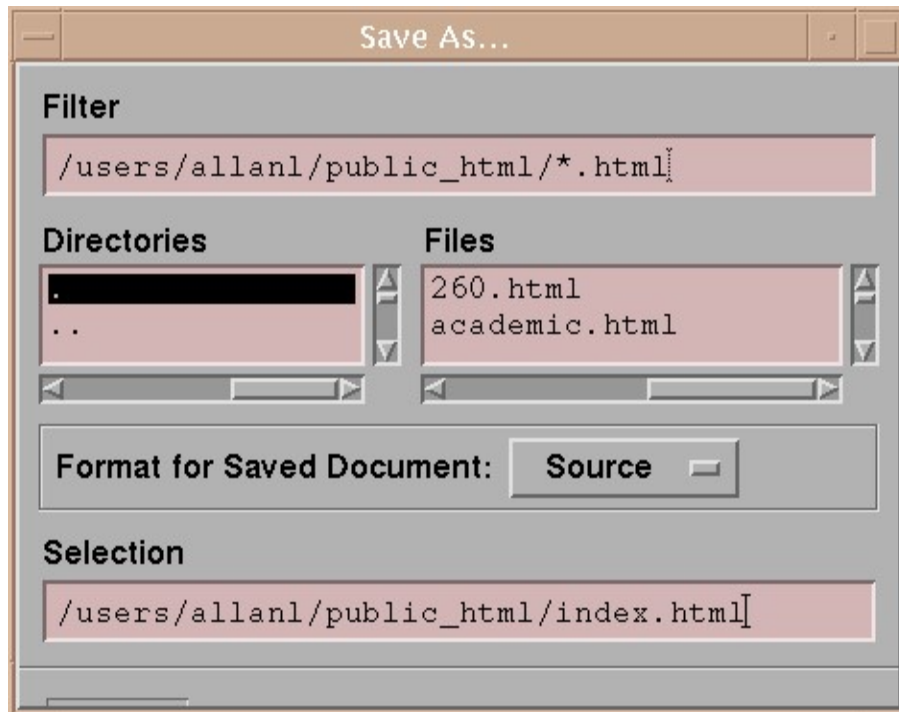
What is Layout?

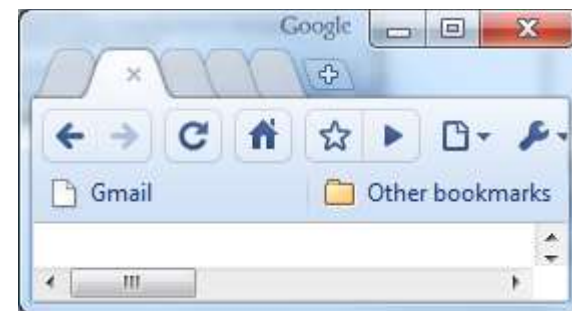
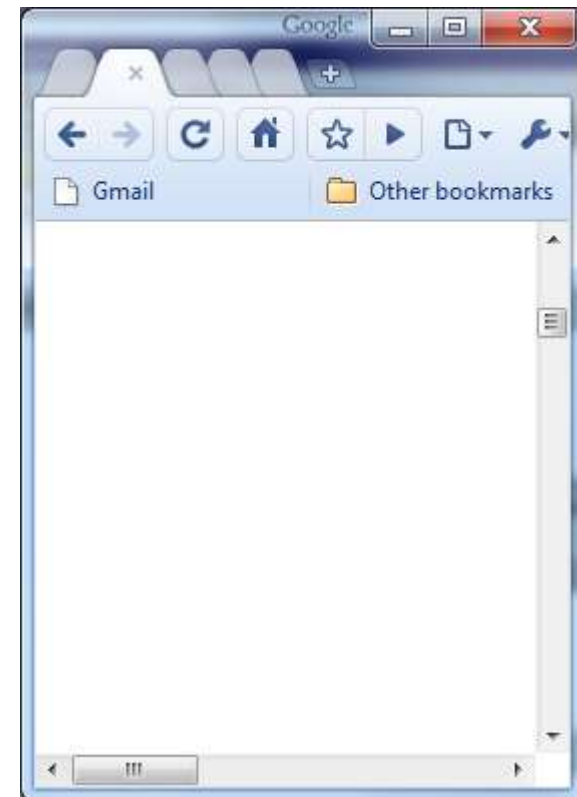
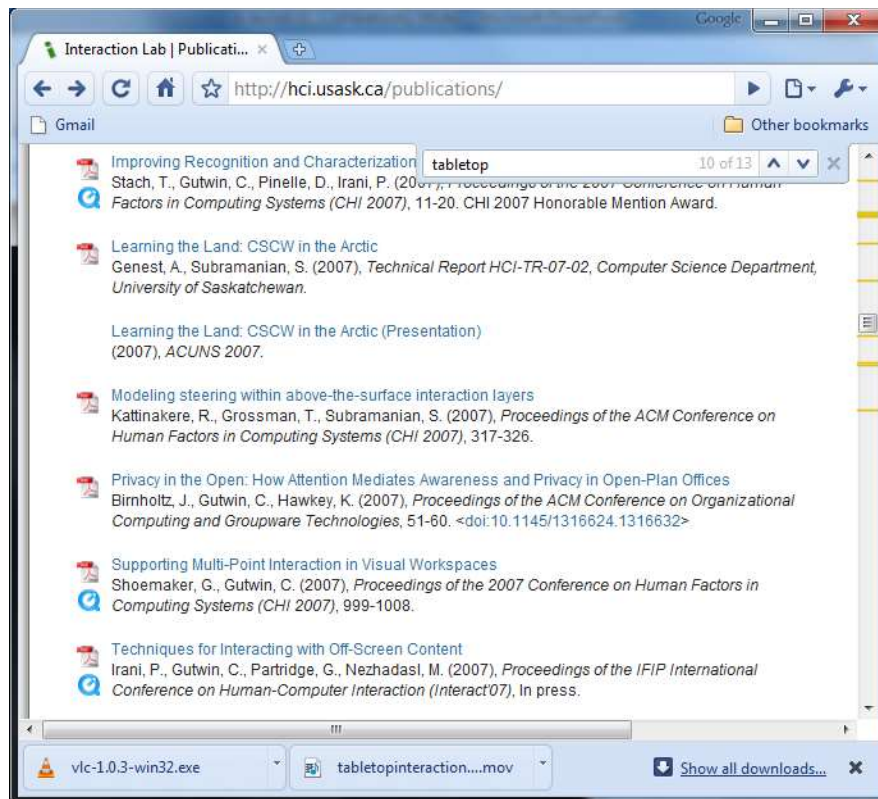
- Position of widgets in the containing window
- What to do when the window changes size?
 - Must be programmed
- **Layout Managers** greatly reduce the need to worry about positioning, sizing and alignment of widgets

Resizing problems



Resizing problems





Review: Window Tree

- hierarchical layout

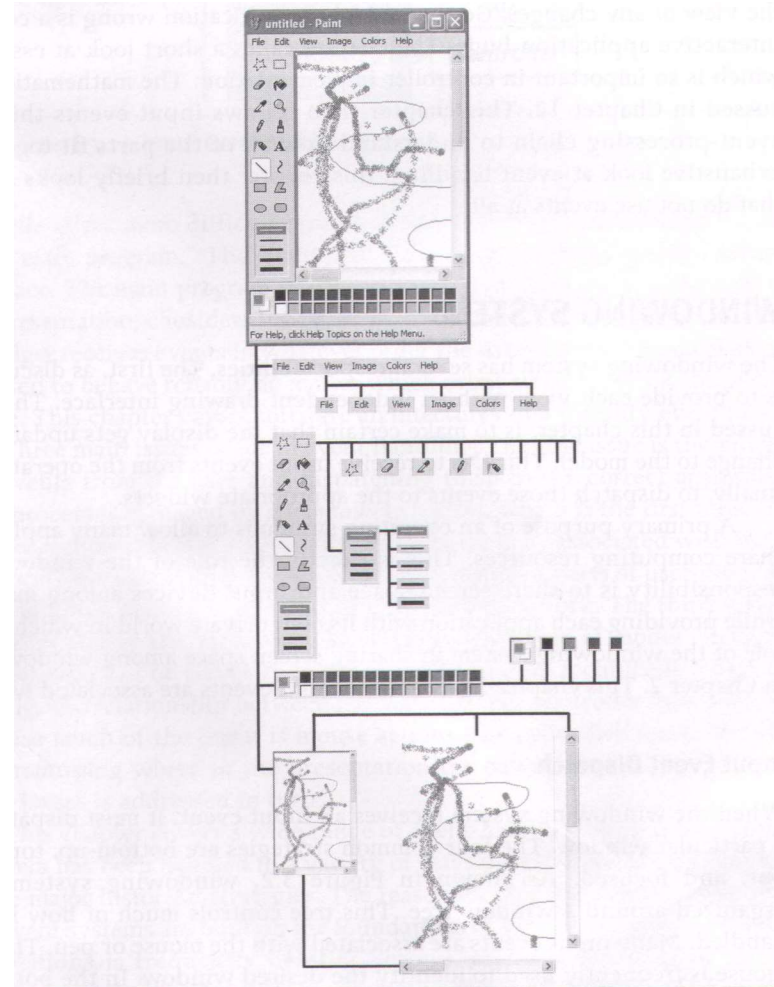
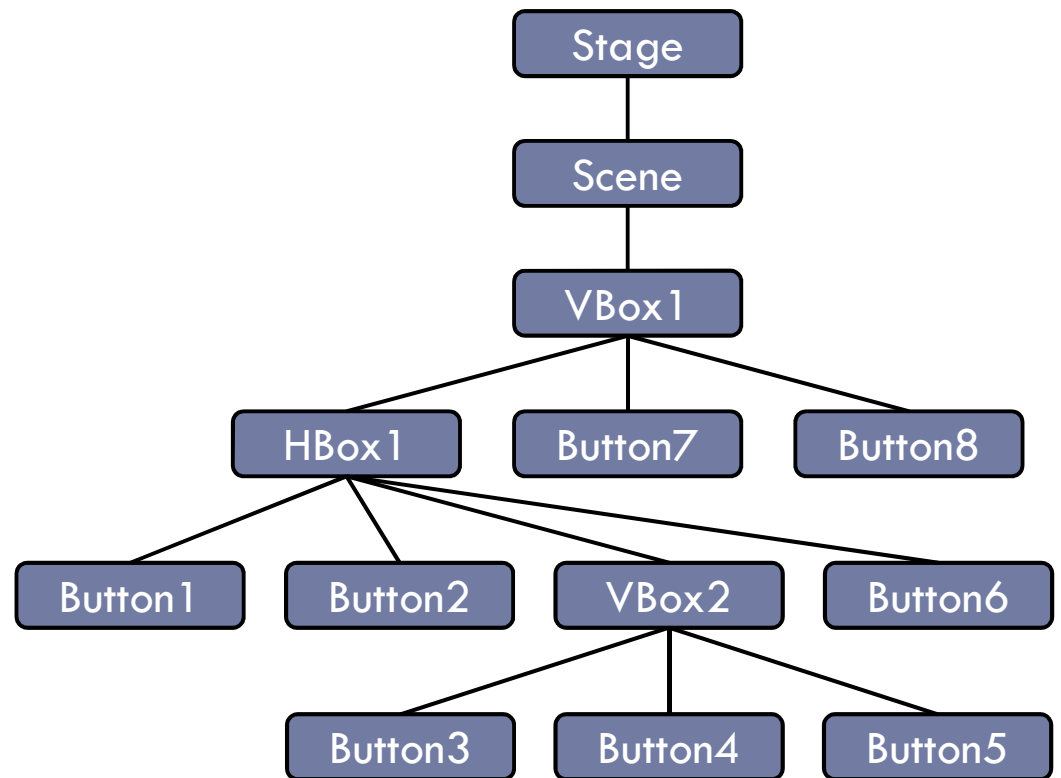
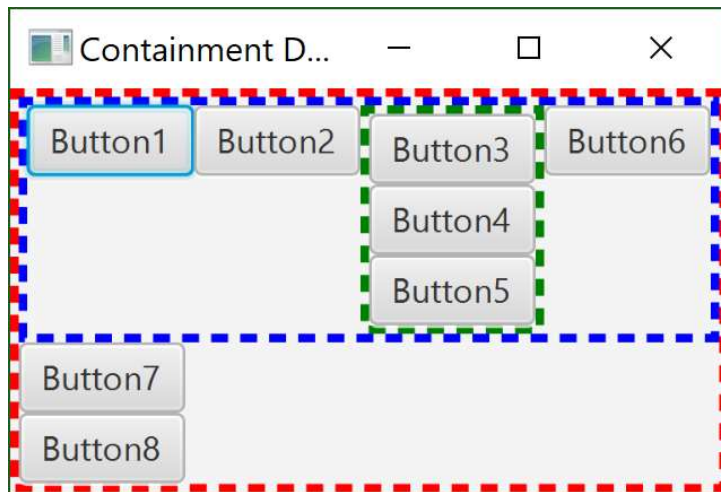


Figure 3.2 – Window tree

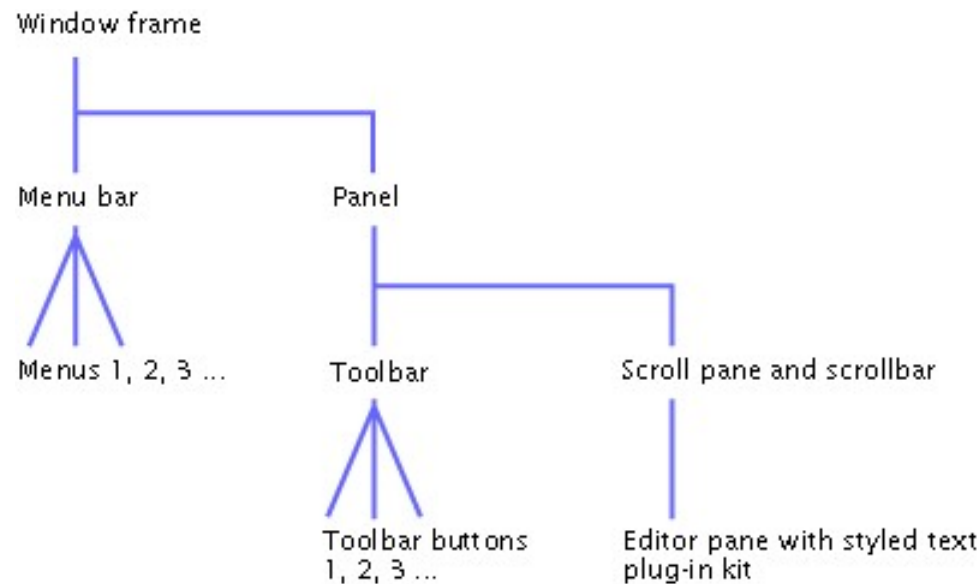
Review: Containment Hierarchy



Used for event routing – but also important for layout

Hierarchical Layout Basics

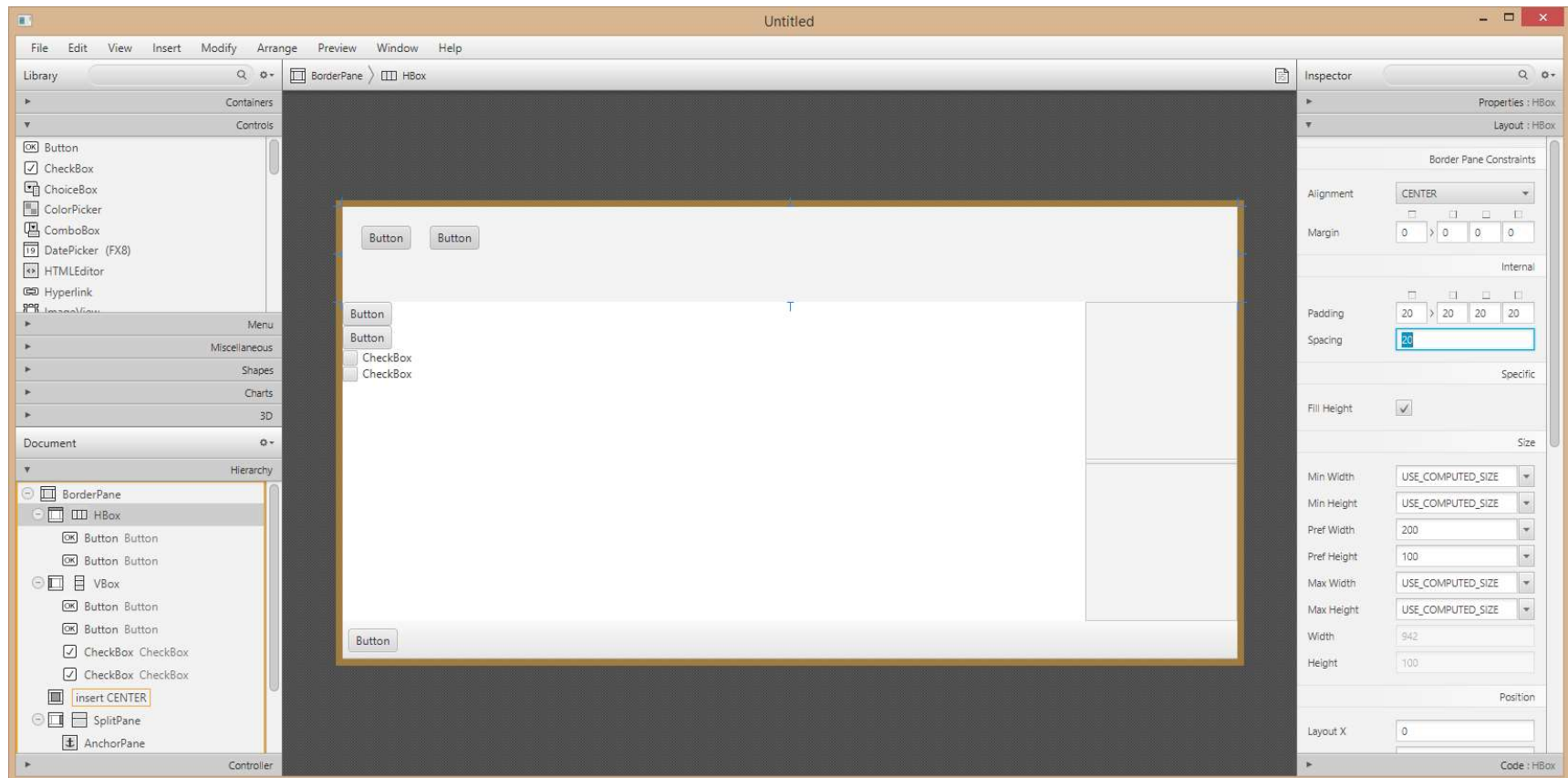
- Widgets positioned relative to their parent
 - Many different (0,0) locations!
- Widgets positioned according to a layout algorithm



Layout Approaches

- Fixed Layout
- Stacked Layout
- Edge-Anchored Layout
- Variable Intrinsic Size Layout
- Automated Approaches

(but what about...) GUI Builders

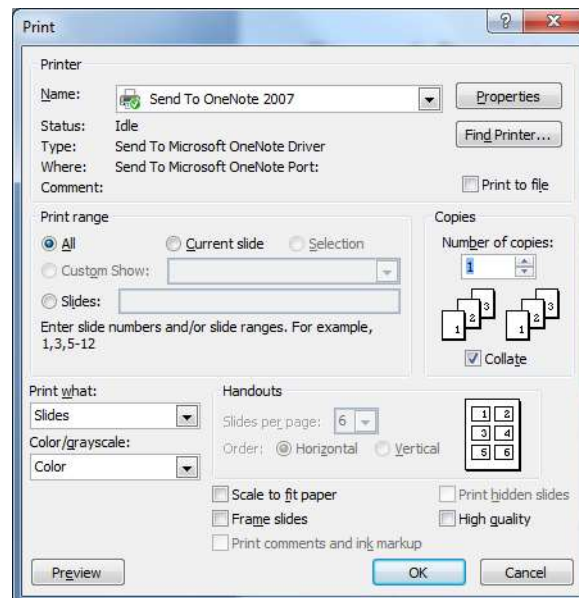


Why not use a GUI builder?

- These are getting better, BUT:
 - They can cause problems
 - e.g., they may use simplistic models
 - Resizing still an issue!
 - The generated code is often not maintainable
 - Fixing problems requires understanding the underlying principles anyway
 - leepoint.net/JavaBasics/gui/gui-commentary/guicom-20-byhand.html
 - gluonhq.com/labs/scene-builder/
- SceneBuilder and Android now include the standard layout containers (better than previous versions)
 - Only generates XML

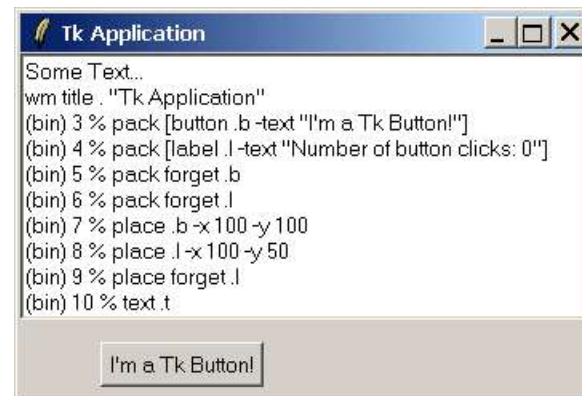
Fixed Position Layout

- Store a rectangle for each component
- Simple to implement
- Widgets stay where you put them
- Makes sense for a dialog that is not resizable



Tk Placer

- `button .b -text "I'm a Tk Button!"`
- `text .t -height 10 -width 50`
- `place .b -x 50 -y 180`
- `place .t -x 0 -y 0`



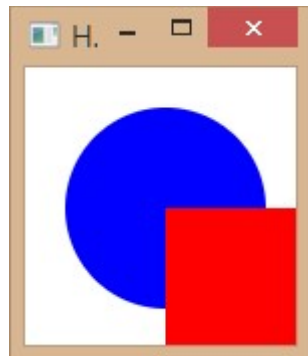
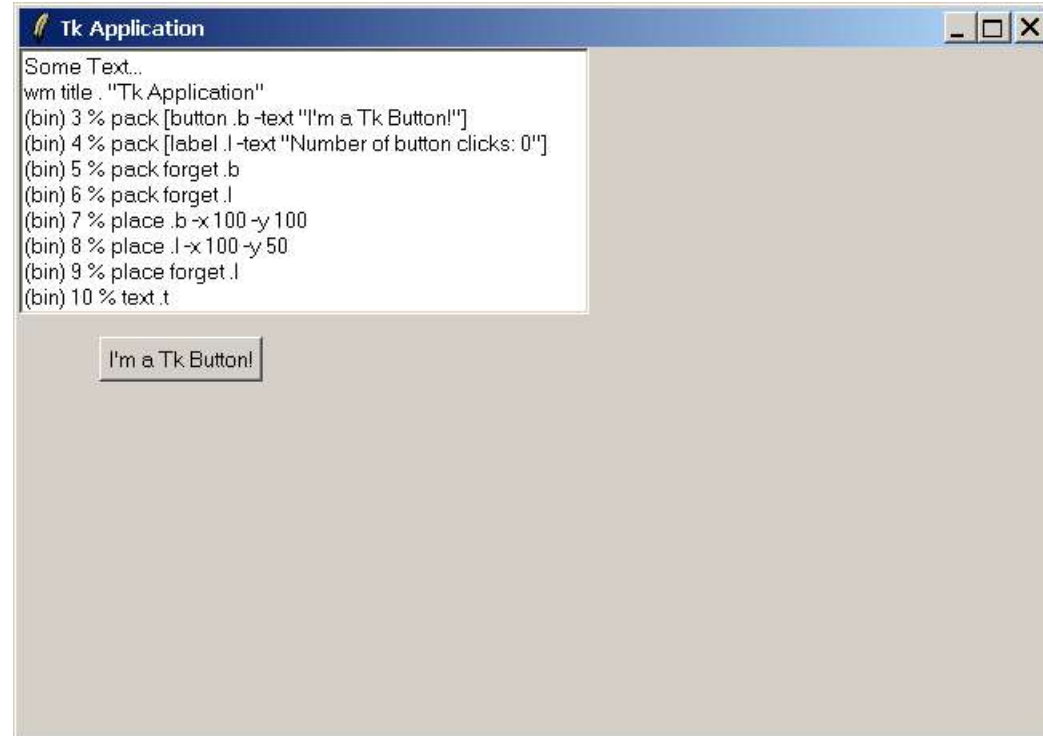
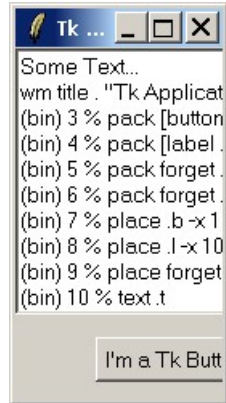
JavaFX Pane

```
public class LayoutTest extends Application {
    @Override
    public void start(Stage primaryStage) {
        Pane root = new Pane();
        Circle circle = new Circle(50, Color.BLUE);
        circle.relocate(20, 20);
        Rectangle rectangle = new Rectangle(100, 100, Color.RED);
        rectangle.relocate(70, 70);
        root.getChildren().addAll(circle, rectangle);

        Scene scene = new Scene(root, 300, 250);

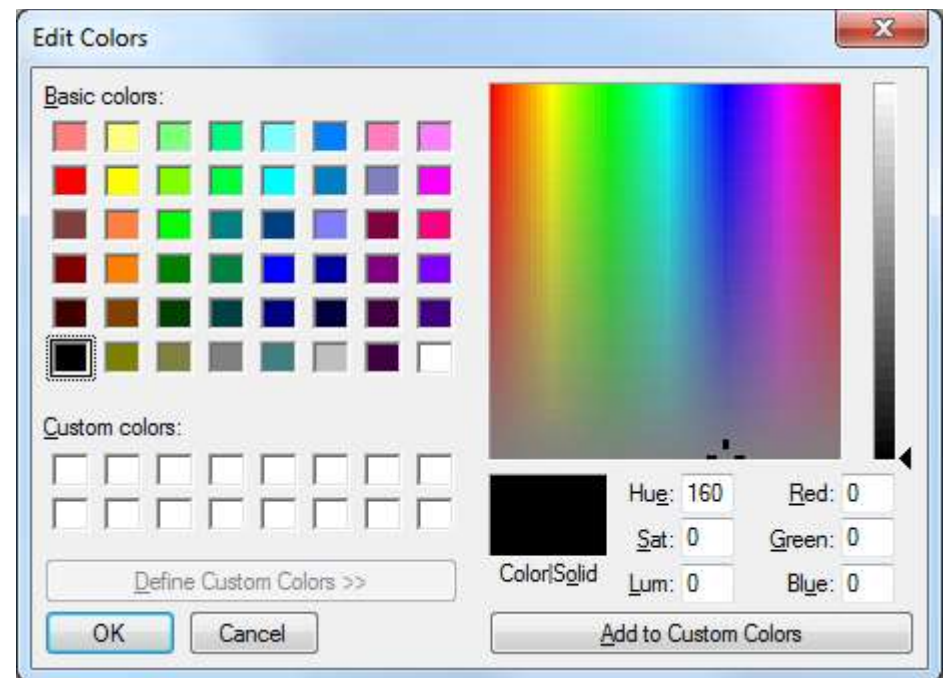
        primaryStage.setScene(scene);
        primaryStage.show();
    }
}
```

Problems with fixed positioning



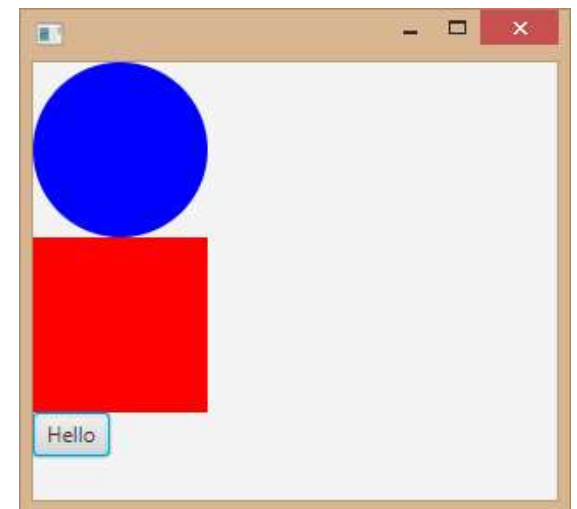
Linear Layouts

- Place sub-containers in horizontal or vertical lists
- No need to worry about specific locations
- Widgets arranged inside containers horizontally or vertically



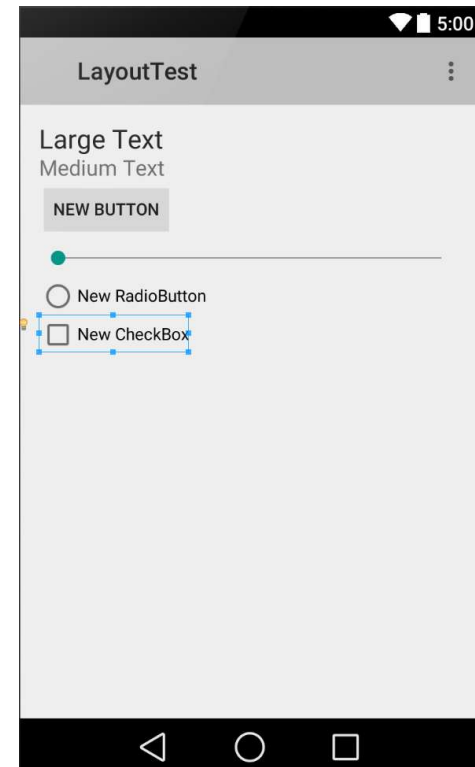
JavaFX VBox (HBox is similar)

```
public class LayoutTest extends Application {  
    @Override  
    public void start(Stage primaryStage) {  
        VBox root = new VBox();  
        Circle circle = new Circle(50, Color.BLUE);  
        Rectangle rectangle = new Rectangle(100, 100, Color.RED);  
        Button button = new Button("Hello");  
        root.getChildren().addAll(circle, rectangle, button);  
  
        Scene scene = new Scene(root, 300, 250);  
  
        primaryStage.setScene(scene);  
        primaryStage.show();  
    }  
}
```



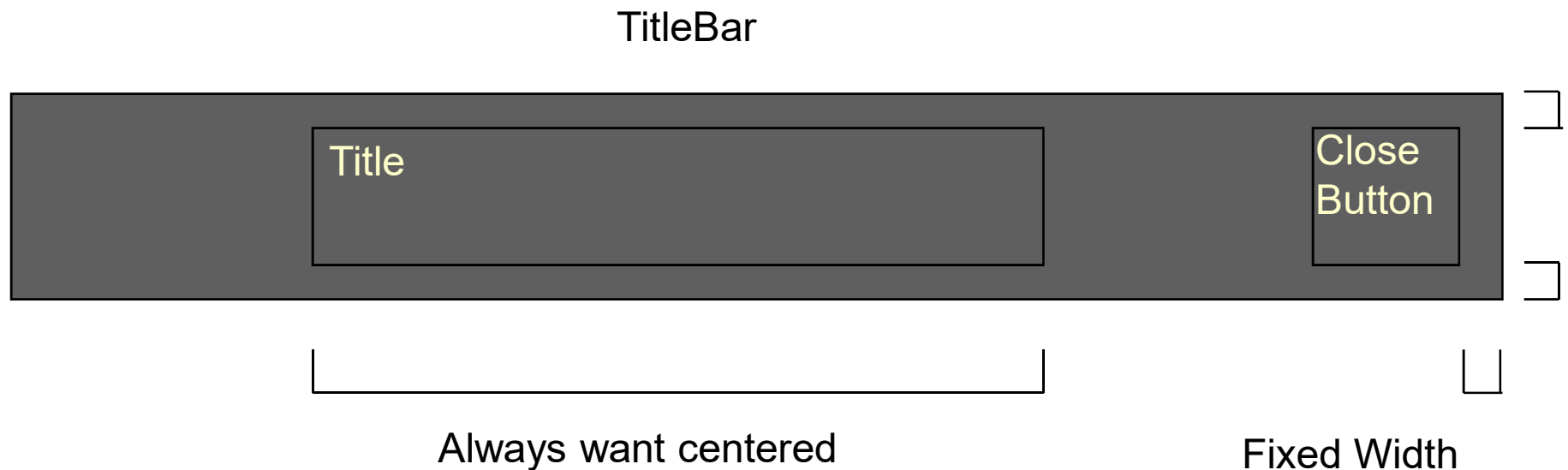
Android LinearLayout

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical">
    <TextView
        android:text="Large Text"
        android:id="@+id/textView2" />
    <TextView
        android:text="Medium Text"
        android:id="@+id/textView" />
    <Button
        android:text="New Button"
        android:id="@+id/button" />
    [...]
</LinearLayout>
```



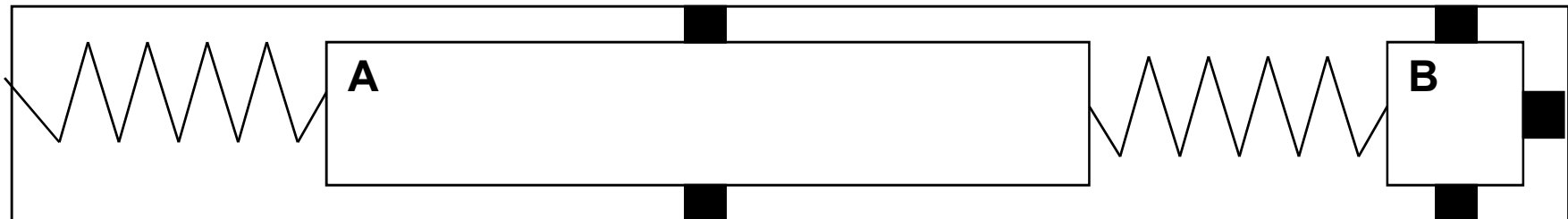
Edge-Anchored Layout

- We want something as easy to use as fixed position but that will also handle resizing
- Programmer doesn't worry about specific positions



Struts and Springs

- Place struts and springs in the layout
- Struts represent fixed lengths
- Springs push as much as they can
- Could use constraints:
 - $\text{CloseButton.RIGHT} = \text{TitleBar.RIGHT} - 5$
 - $\text{Title.CENTER} = \text{TitleBar.CENTER}$



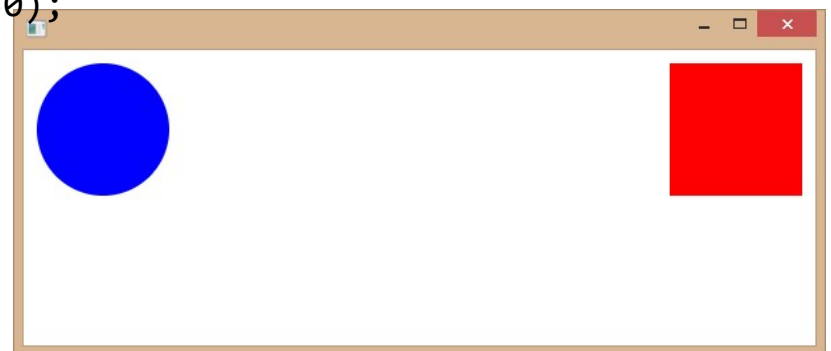
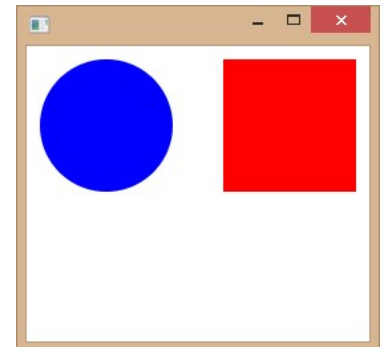
Struts and Springs

- Relative constraints in Tk placer:
 - entry .e
 - button .b –text “Search”
 - place .e -anchor c -relx 0.5 -rely 0 -y 15
 - place .b1 -anchor e -relx 1.0



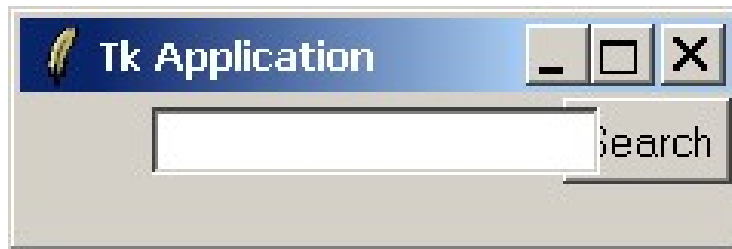
JavaFX AnchorPane

```
public void start(Stage primaryStage) {  
    AnchorPane root = new AnchorPane();  
    Circle circle = new Circle(50, Color.BLUE);  
    AnchorPane.setTopAnchor(circle, 10.0);  
    AnchorPane.setLeftAnchor(circle, 10.0);  
    AnchorPane.setRightAnchor(circle, 65.0);  
    Rectangle rectangle = new Rectangle(100, 100, Color.RED);  
    AnchorPane.setTopAnchor(rectangle, 10.0);  
    AnchorPane.setRightAnchor(rectangle, 10.0);  
    root.getChildren().addAll(circle, rectangle);  
  
    Scene scene = new Scene(root, 300, 250);  
    primaryStage.setScene(scene);  
    primaryStage.show();  
}
```



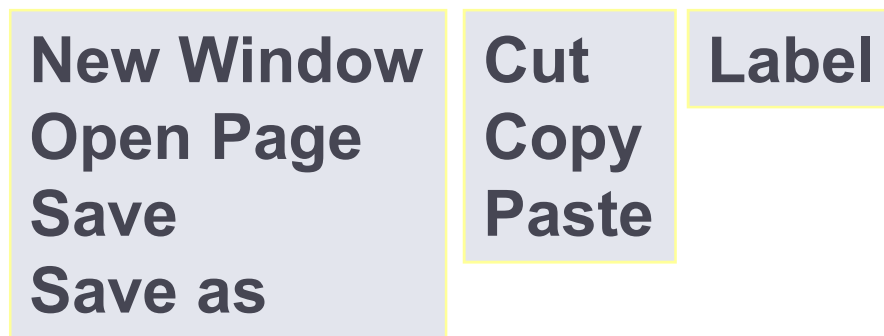
Problems with constraints

- Complicated
 - Programmer must keep track of geometry
 - With large numbers of widgets this is difficult
 - Often impossible to specify constraints adequately



Variable Intrinsic Size

- Size of widget determined by sizes of items within
 - e.g. Menus, most Java widgets



- Intrinsic size does not handle resizing
- Variable Intrinsic Size
 - Each widget reports its size needs (recursively if necessary)
 - Each widget also reports how much it can be reasonably squeezed or expanded

Automated Layout

- A layout manager positions and sizes widgets
 - Programmer provides constraints
- Many different algorithms to determine where widgets should go when added to a container
 - JavaFX:
 - VBox, HBox, BorderPane, FlowPane, GridPane, StackPane, TilePane (and others)
 - Android:
 - LinearLayout, RelativeLayout, ConstraintLayout, GridLayout (now listed as “legacy”) (and others)

Two-Stage Layout Process

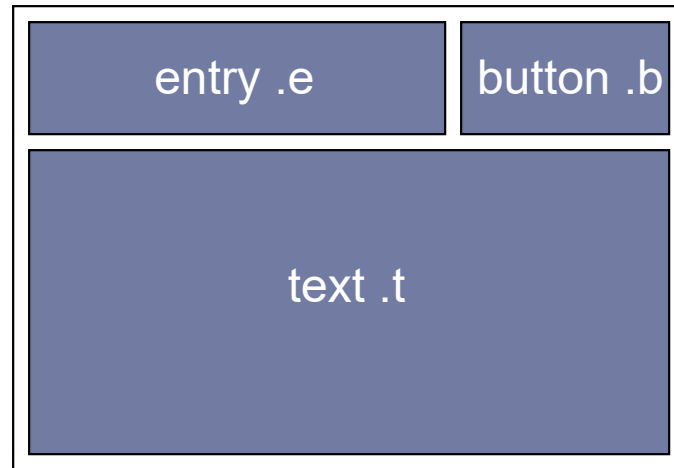
- Stage 1: allocation of space to each child
 - Determine each child's intrinsic size (recursively)
 - Allocate space according to the layout algorithm
 - Use the container-level constraints, if any
 - This gives each child an area of the screen (the *parcel*)
- Stage 2: layout within the parcel
 - Determine where the child goes inside its allocation
 - Alignment: top, bottom, left, right, middle
 - Padding: space at edges
 - Fill: horizontal, vertical, both

A Simple Layout Algorithm: Packing

- Tk packer:
 - Places widgets by packing them around window edges
 - Keeps track of remaining space (the *cavity*)
 - Algorithm:
 - For each widget to be packed:
 - Allocate a rectangular *parcel* on the indicated side of the cavity (keeping the cavity rectangular)
 - Determine the size of the widget
 - Position the widget in its parcel
 - Subtract the parcel from the cavity

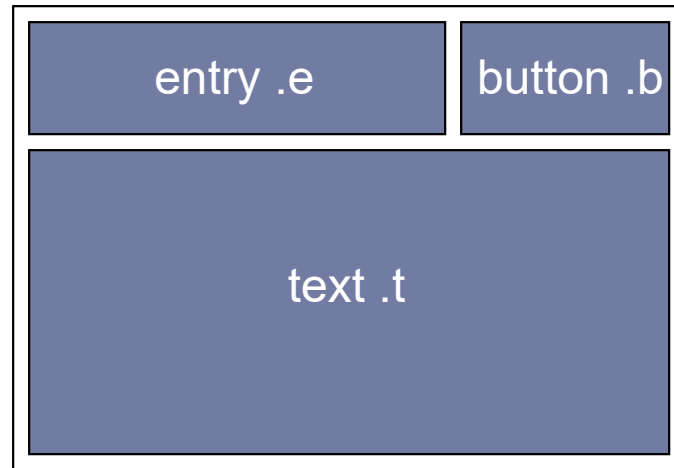
Packing example

- We want to end up with this:



Packing example

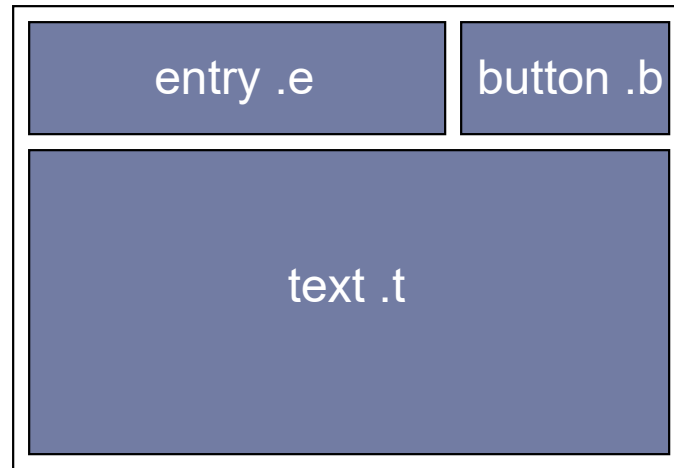
- We want to end up with this:



- For each widget to be packed:
 - Allocate a rectangular *parcel* on the indicated side of the cavity (keeping the cavity rectangular)
 - Determine the size of the widget
 - Position the widget in its parcel
 - Subtract the parcel from the cavity

Packing example

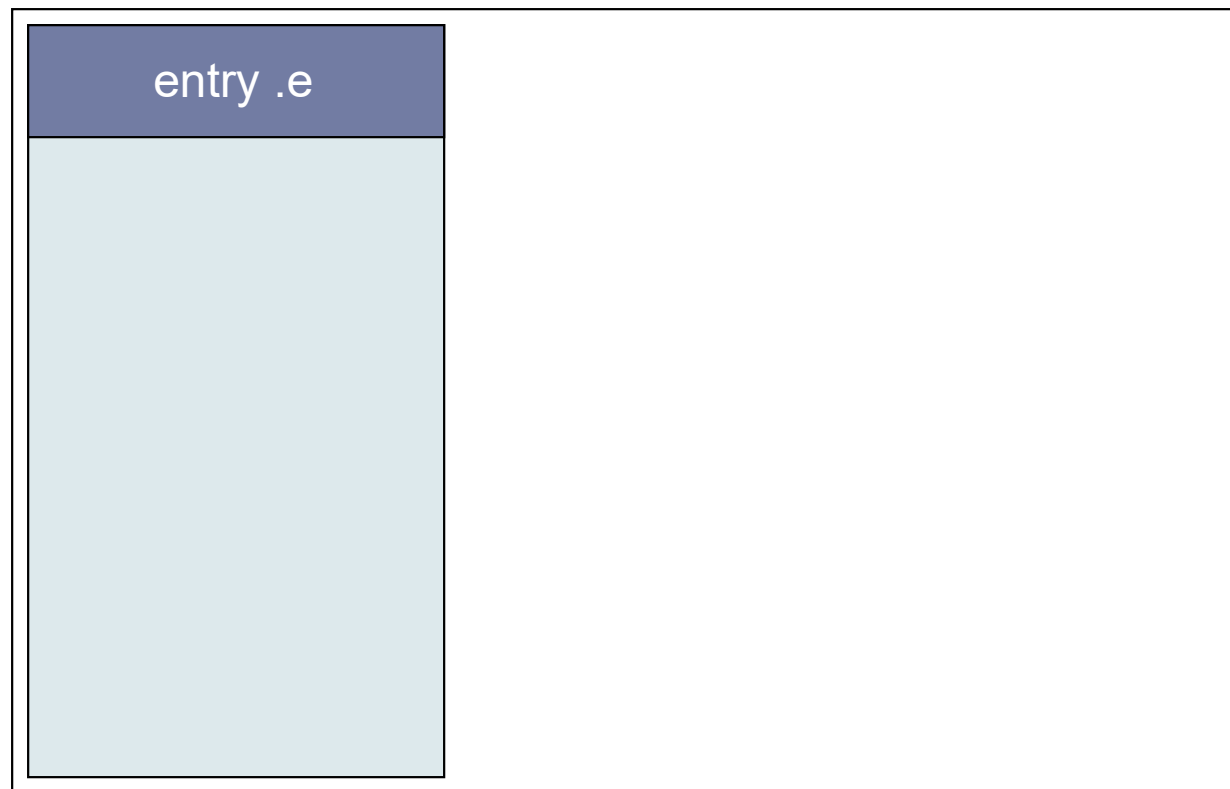
- We want to end up with this:



- Try:
 - pack .e -side left
 - pack .b -side right
 - pack .t -side bottom

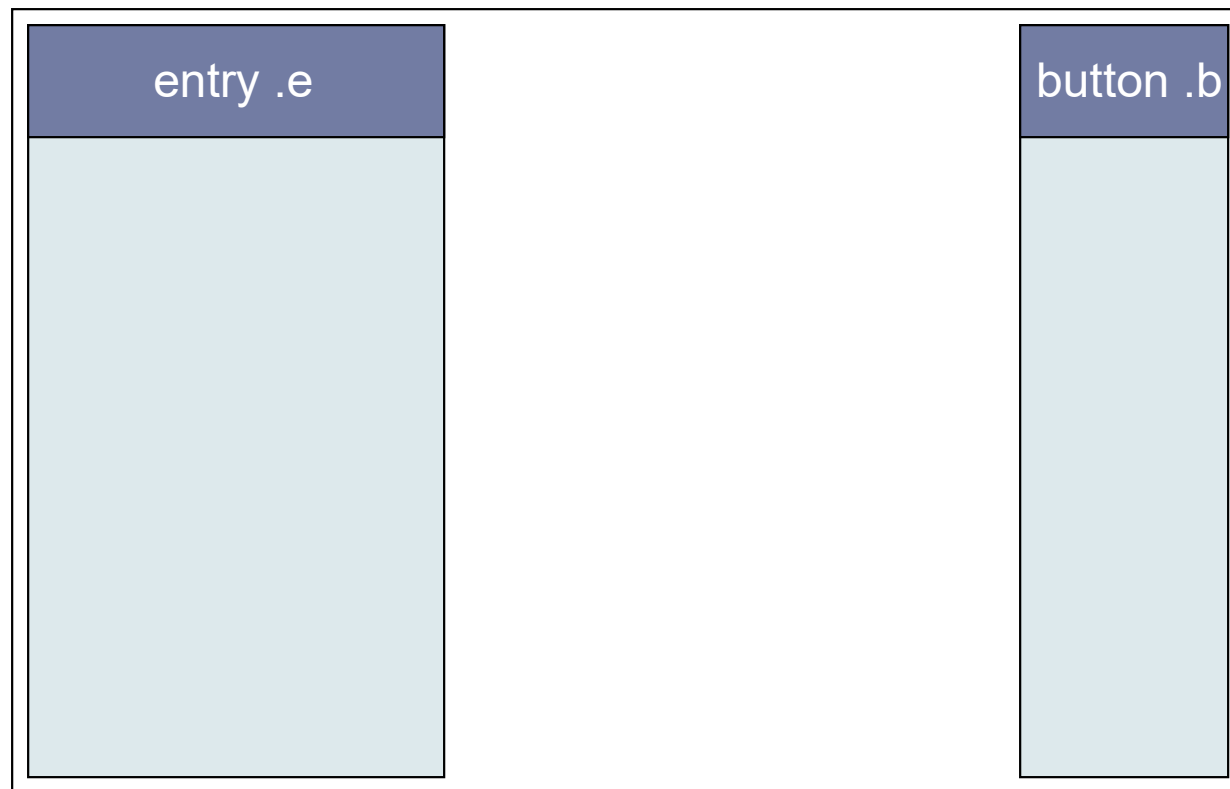
Packing example

- `pack .e -side left -anchor n`



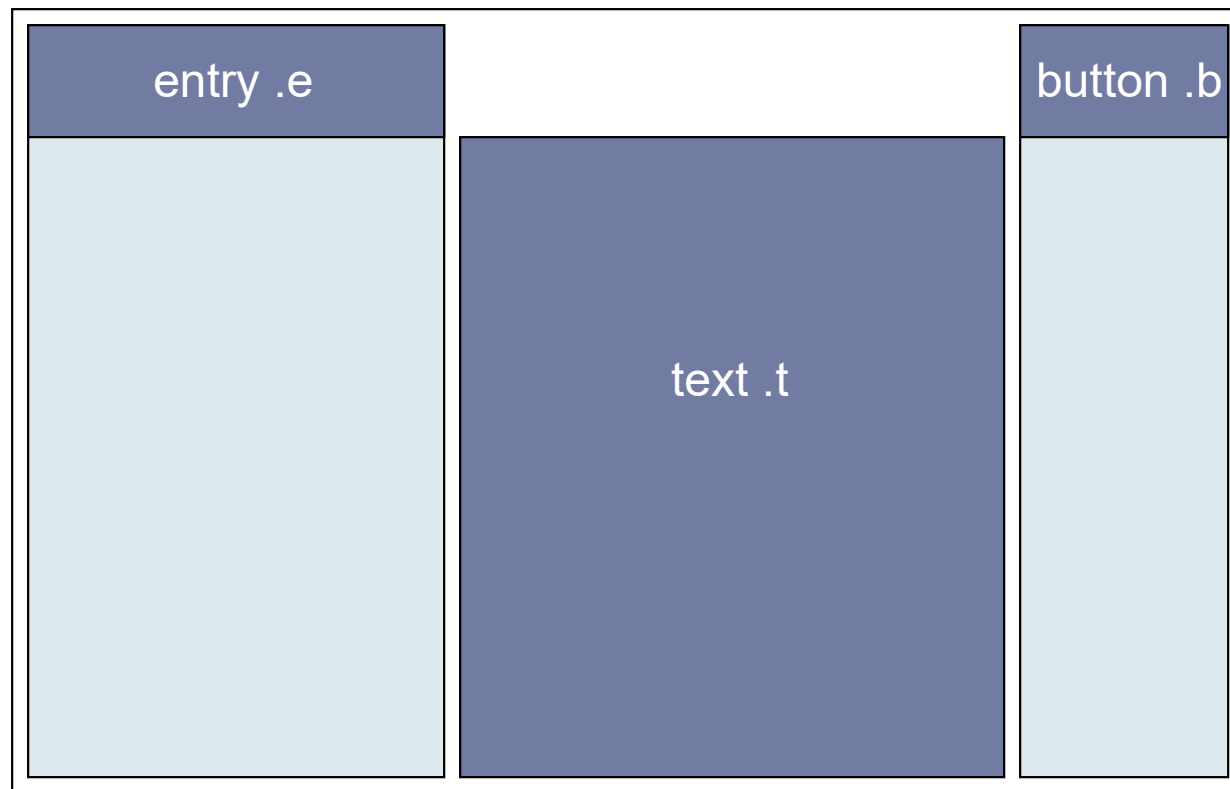
Packing example

- `pack .b -side right -anchor n`



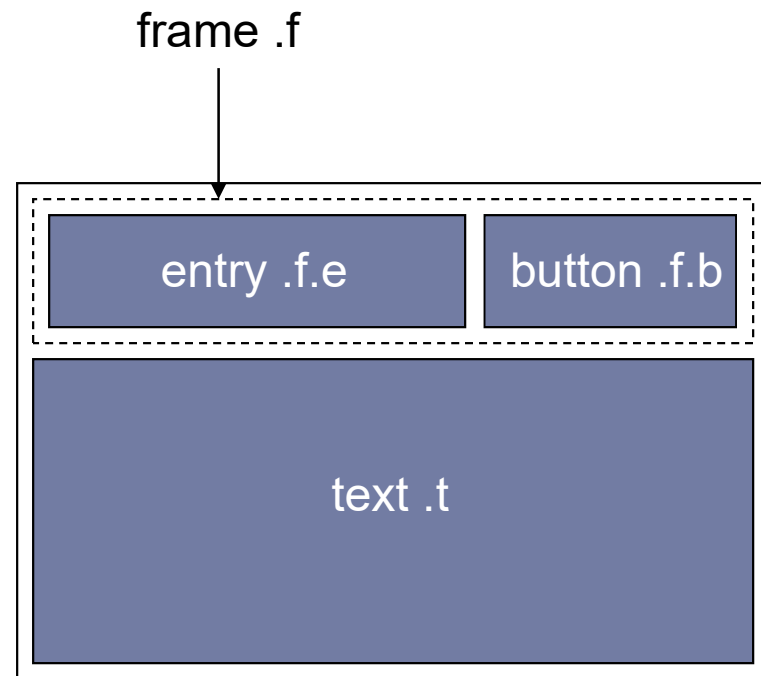
Packing example

- `pack .t -side bottom`

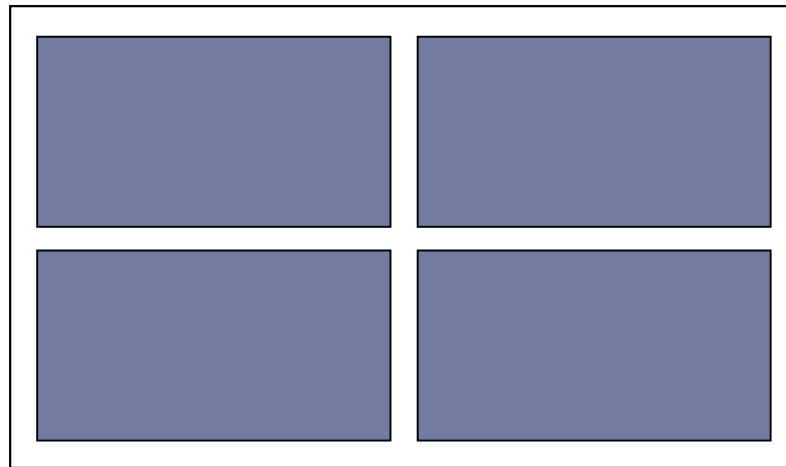


Packing example

- How to make this work?
 - Do we need an internal frame?

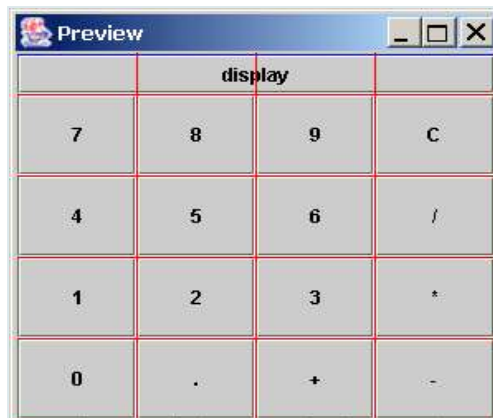


Pack layout challenge



Grid-based layout

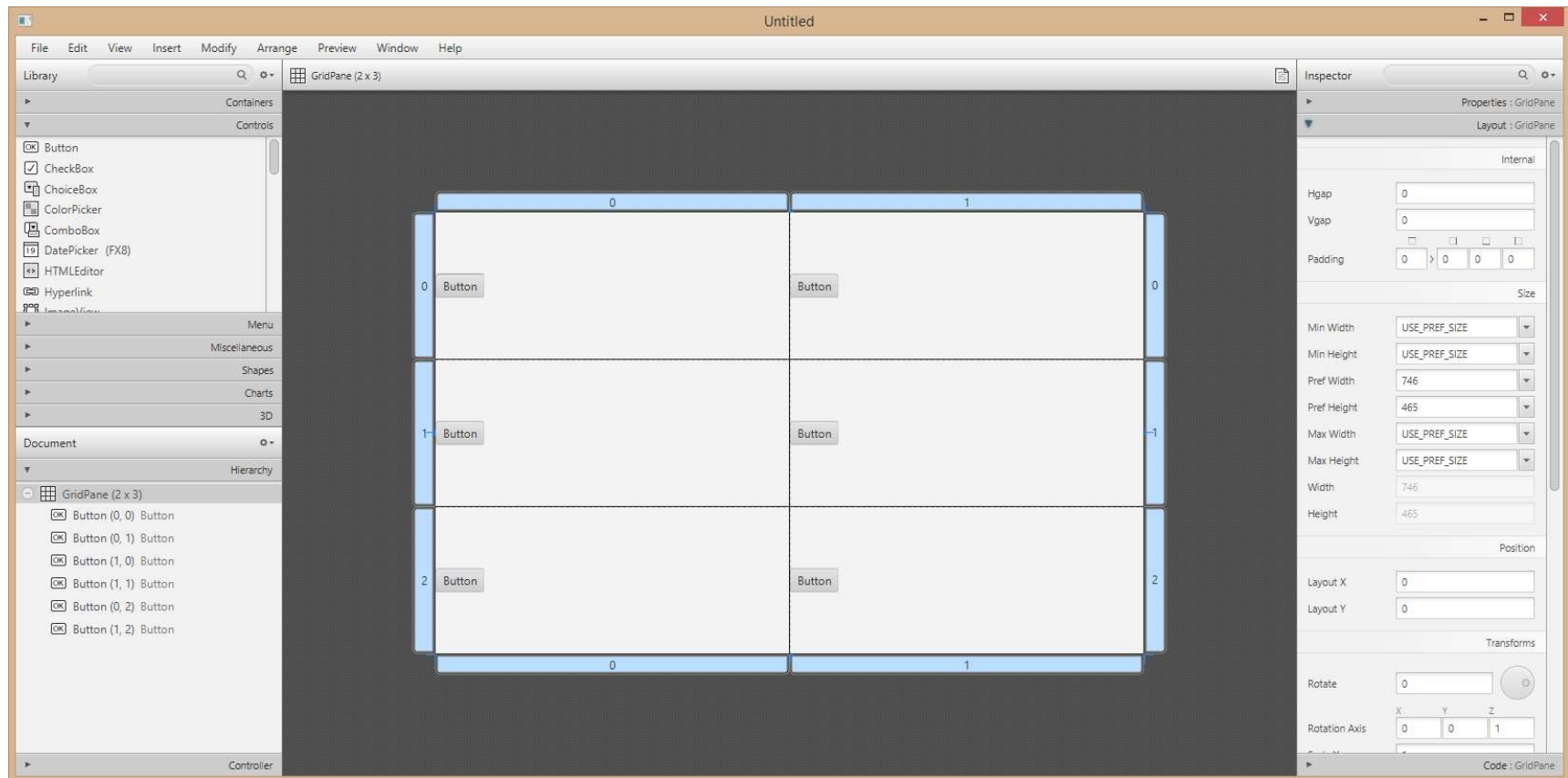
- Puts components on a virtual grid
 - Rows and columns
 - Components fit into one or more cells
- Easy for programmers to understand
 - Works well with grid-based screen design
- Layout controlled by constraints



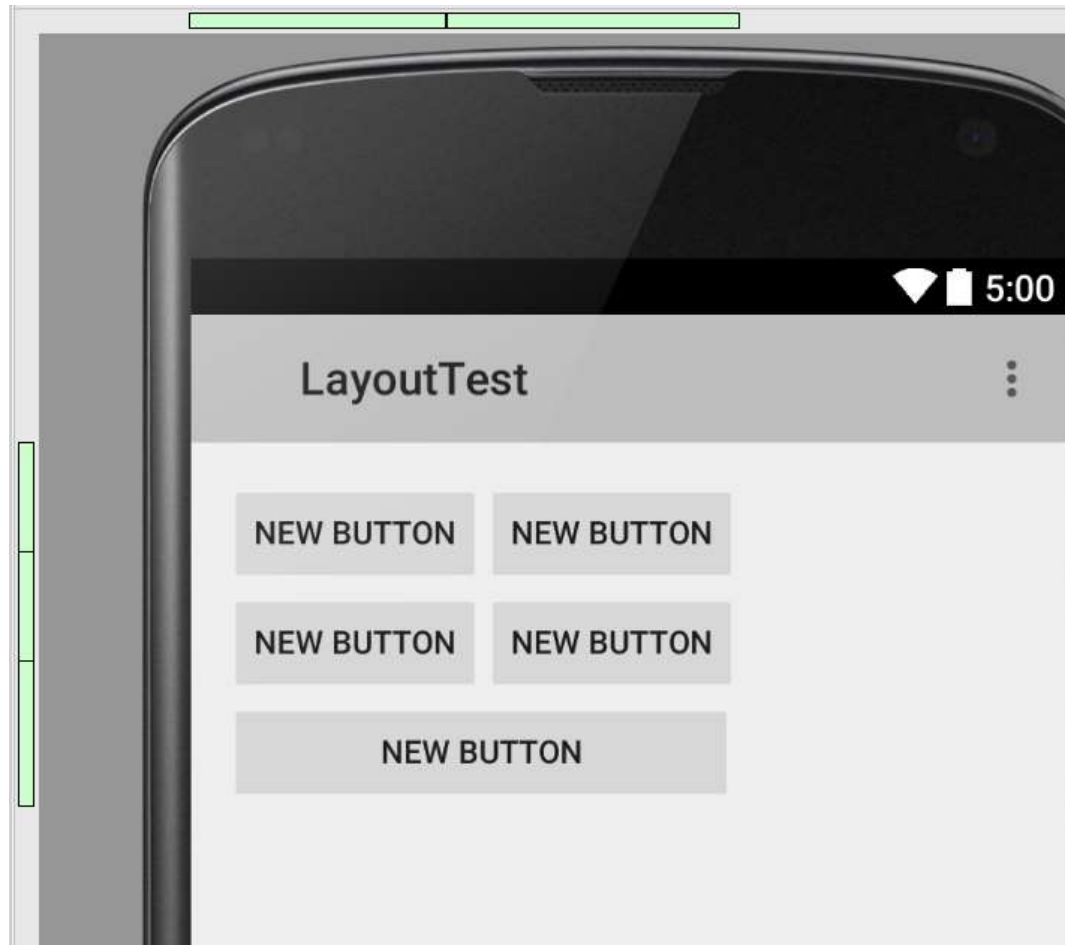
Grid constraints

- Where in the grid to start
 - JavaFX: `setRowIndex()`, `setColumnIndex()`
 - Android: `android:layout_column`, `android:layout_row`
- How many cells to take up
 - JavaFX: `setRowSpan()`, `setColumnSpan()`
 - Android: `android:layout_columnSpan`, `layout_rowSpan`
- What to do when the cell resizes
 - JavaFX: `setHgrow()`, `setVgrow()`
 - Android: `android:layout_rowWeight`, `columnWeight`

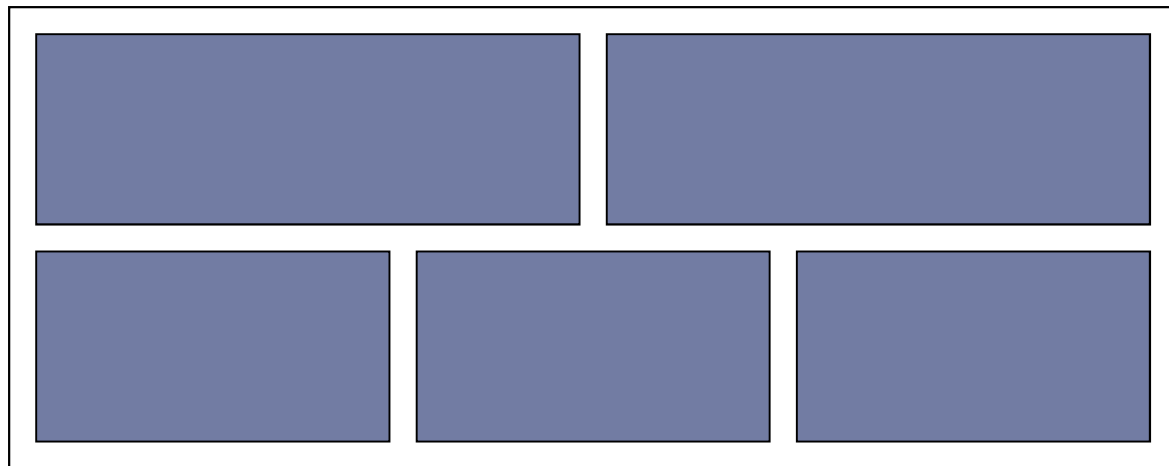
JavaFX GridPane (in Scene Builder)



Android GridLayout



Grid layout challenge



In Practice

- May combine layout managers
- e.g., in JavaFX:
 - Use GridPane or BorderPane to place main components
 - Use linear layouts (e.g., HBox) for sets of tools

Choosing a Layout Manager

- JavaFX
 - <http://docs.oracle.com/javase/8/javafx/layout-tutorial/index.html>
- Android
 - <http://developer.android.com/guide/topics/ui/declaring-layout.html>