# CS 683 Emerging Technologies: Embracing Change
# Spring Semester, 2001
# Doc 18 Squeak GUI
# Contents

References ..... 1
Squeak GUI ..... 2
  Pen ..... 3
  Some Pen Operations ..... 4
  Color ..... 8
  Forms ..... 9
Observer ..... 12
  Structure ..... 13
  Collaborations ..... 14
GUI Examples ..... 16
  A Button Example ..... 16
  Text Field Example ..... 19
    PluggableButtonMorph ..... 21
  Text & Button Example ..... 22
    Counter - No GUI ..... 22
    Counter - With GUI ..... 23

## References

Squeak: Object-Oriented Design with Multimedia Applications, Guzdial, 2001, Chapter 3, Chapter 5

Design Patterns: Elements of Reusable Object-Oriented Software, Gamma, Helm, Johnson, Vlissides, 1995

Squeak Source Code

Copyright©, All rights reserved. 2001 SDSU & Roger Whitney, 5500 Campanile Drive, San Diego, CA 92182-7700 USA. OpenContent (http://www.opencontent.org/opl.shtml) license defines the copyright on this document.

# Squeak GUI
# For Fun - Worlds Smallest Drawing Program

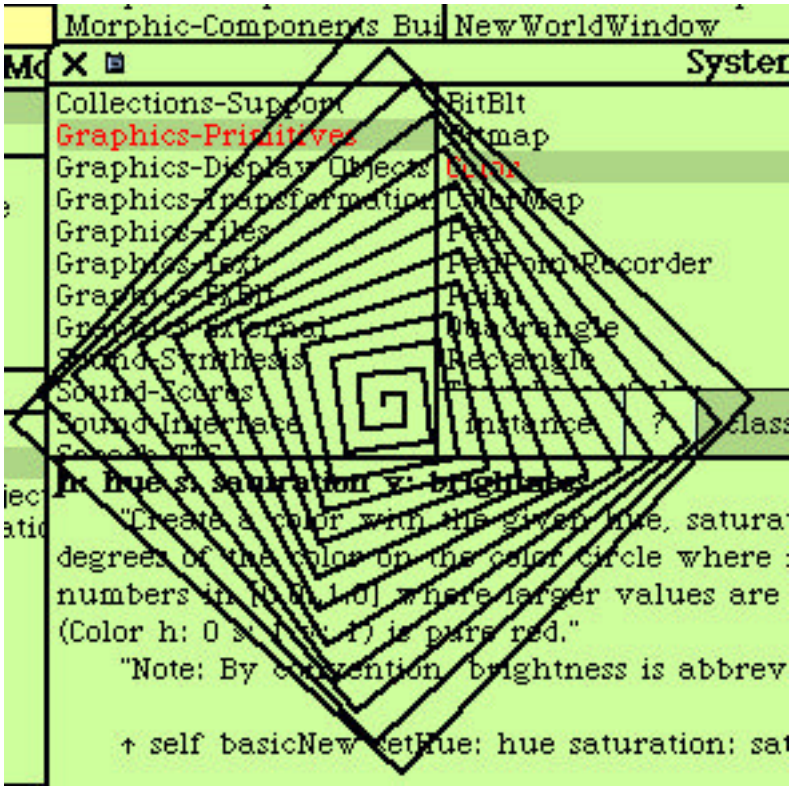
Draws while Mouse is down (left-click windows)

End program with yellow button
(right-click 2 button, left click 3-button)

```
| aPen color |
aPen := Pen new.
color := 0.
[Sensor yellowButtonPressed]
  whileFalse:
    [aPen
      place: Sensor cursorPoint;
      color: (color := color + 1).
    [Sensor redButtonPressed]
      whileTrue: [aPen goto: Sensor cursorPoint]].
Sensor waitNoButton.
```

# Pen

```
| pen |
pen := Pen newOnForm: Display.
pen defaultNib: 2.
1 to: 50 do:
   [:i |
   pen
      go: i*4;
      turn: 89].
```




Note that we are drawing directly on the screen over what ever is displayed

# Some Pen Operations

## Initializing

defaultNib: nibSize
square nib of given size, black ink

roundNib: nibSize
Sets nib shape to round with given diameter

squareNib: nibSize
Sets nib shape to square with given diameter

## Operations

color: aColor
Set the pen color

down
Set the pen down, so it will draw

fill: aDrawblock color: aColor
Draw a shape and fill it with a color. Shape must be closed

go: n
Go n pixels in the current direction. Pen draws if is down

goto: aPoint
Go to given point. Pen draws if it is down

home
Move pen to home, the center of the drawing area. Does not draw

## Pen Operations continued

north
Set the pen direction heading north

place: aPoint
Move the pen to the given location. Does not draw.

print: aString withFont: aFont
Print aString using aFont, in the current direction and color.

turn: degrees
Turn the pen degrees clockwise

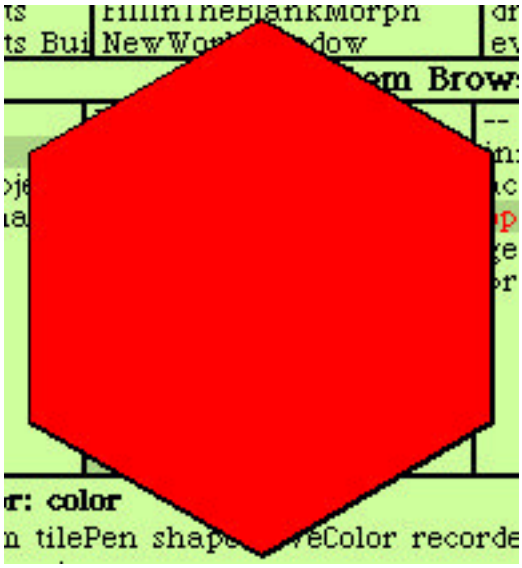
up
Lift the pen up. Pen only draws when down

## Geometric Designs

dragon:
filberts:side:
hilbert:side:
hilberts:
mandala:
spiral:angle:
web

# Some More Examples

```
| pen |
pen := Pen newOnForm: Display.
pen defaultNib: 2.
pen
   fill: [:each | 6 timesRepeat: [each go: 100;turn: 60]]
   color: Color red
```



## Display restoreAfter:

Drawing directly on the screen requires cleaning up afterwards

Display restoreAfter: aBlock

Performs the block
Updates the display after the user clicks the mouse

## Draw a Mandala

```
Display restoreAfter: [Pen new mandala: 30]
```

## Draw Some Text

```
Display restoreAfter:
   [| pen |
   pen := Pen new.
   pen
      squareNib: 2;
      color: Color red;
      turn: 45.
   pen
      print: 'The owl and the pussycat went to sea
in a beautiful pea green boat.'
      withFont: TextStyle defaultFont]
```

# Color
## Instance Creation

The Color class responds to the following color names:

| | | | |
|---|---|---|---|
| black | blue | brown | cyan |
| darkGray | gray | green | lightBlue |
| lightBrown | lightCyan | lightGray | lightGreen |
| lightMagenta | lightOrange | lightRed | lightYellow |
| magenta | orange | paleBlue | paleBuff |
| paleGreen | paleMagenta | paleOrange | palePeach |
| paleRed | paleTan | paleYellow | red |
| tan | transparent | veryDarkGray | veryLightGray |
| veryPaleRed | veryVeryDarkGray | veryVeryLightGray | white |
| yellow | | | |

Some other instance creation methods
All parameters are floats in the range 0.0 to 1.0

gray: aFloat

h:s:v:
hue, saturation, brightness

r:g:b:
r:g:b:alpha:
red, green, blue, alpha

fromUser
User selects a color from a color palate

# Forms

A rectangular array of pixels, used for holding images

Using a form we can draw an image then display it

```
| form pen |
form := Form extent: 150@150 depth: Display depth.
form fillColor: Color green.
pen := Pen newOnForm: form.
pen
   roundNib: 2;
   color: Color red;
   dragon: 9.
form displayAt: Display center - form center.
```

# A Less Ephemeral Drawing

Up to now we have been drawing directly on the Screen

Usually one displays a drawing in a window so it can be moved, resized etc.

A form can be converted to a morph and displayed

```
| form pen |
form := Form extent: 150@150 depth: Display depth.
form fillColor: Color green.
pen := Pen newOnForm: form.
pen
   roundNib: 2;
   color: Color red;
   dragon: 9.
form asMorph openInWorld
```

# In a Window

```
| form pen |
form := Form extent: 150@150 depth: Display depth.
form fillColor: Color green.
pen := Pen newOnForm: form.
pen
   roundNib: 2;
   color: Color red;
   dragon: 9.
window := SystemWindow labelled: 'Image'.
window
   addMorph: form asMorph
   frame: (0@0 extent: 1@1).
window openInWorld
```

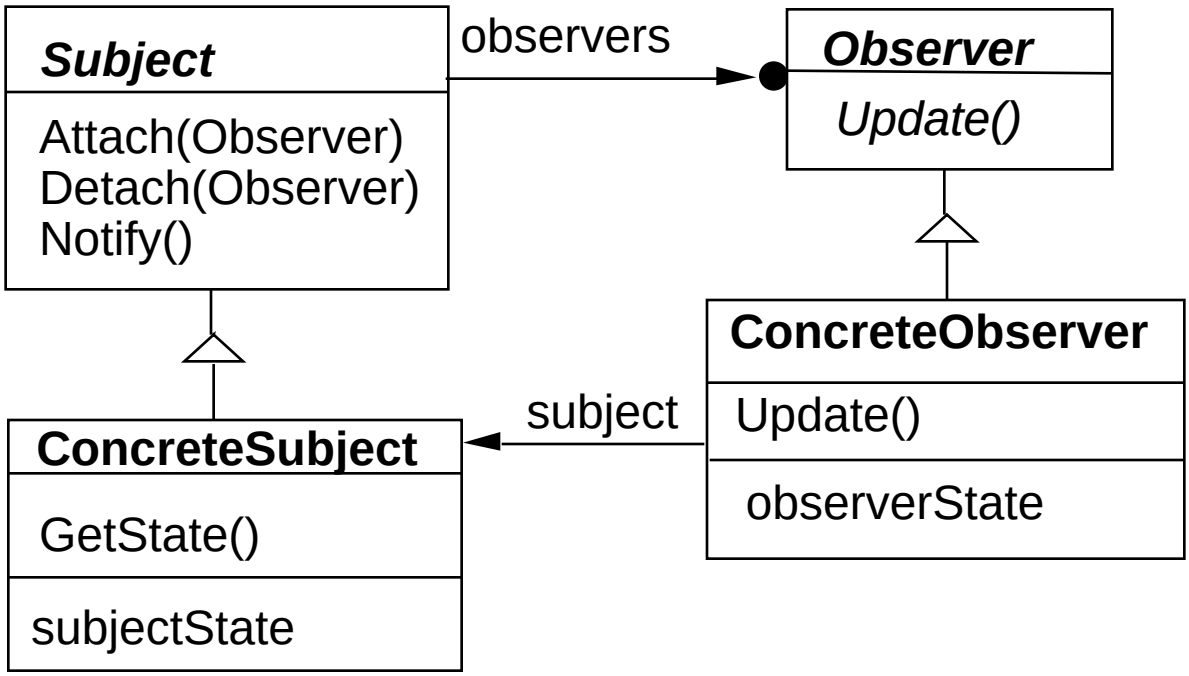
# Observer

Defines a one-to-many dependency between objects so that when one object changes state, all its dependents are notified and updated automatically

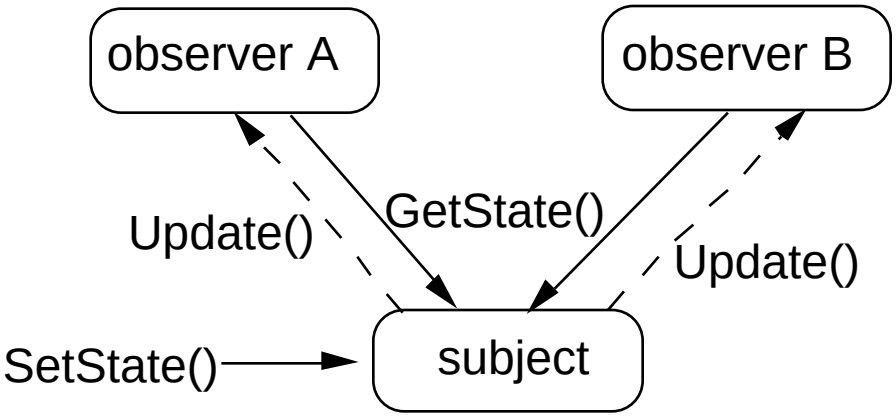
Use the Observer pattern:

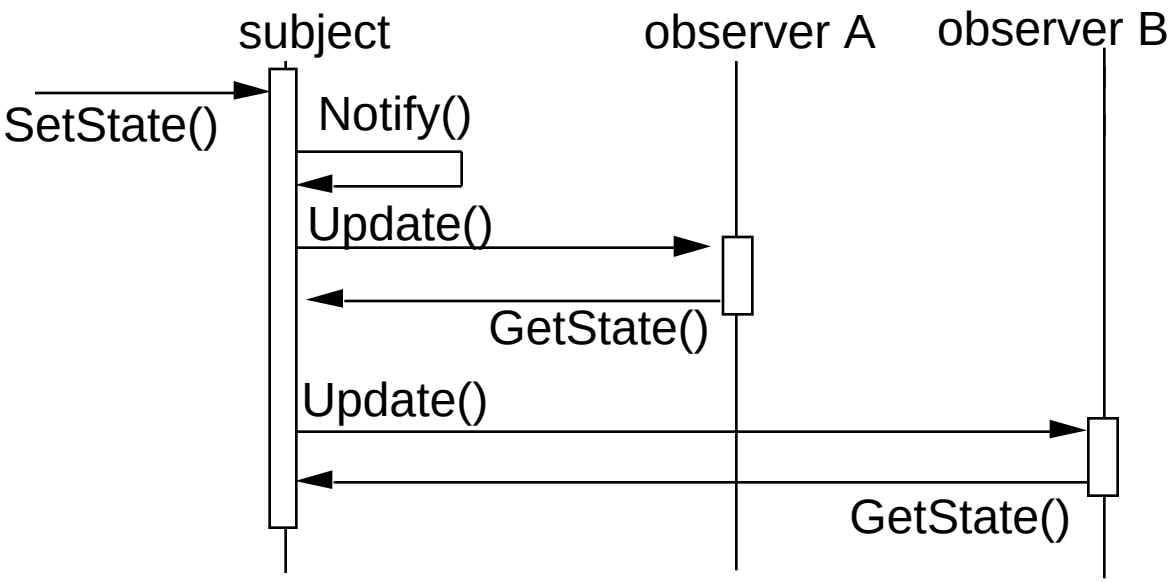
- When an abstraction has two aspects, one dependent on the other.
- When a change to one object requires changing others, and you don't how many objects need to be changed
- When an object should be able to notify other objects without making assumptions about who these objects are.

# Structure


Subject
Attach(Observer)
Detach(Observer)
Notify()
observers
Observer
Update()
ConcreteObserver
Update()
observerState
subject
ConcreteSubject
GetState()
subjectState

# Collaborations



# Smalltalk Observer

Subject is called model

A model can be:

- Subclass of Object
- Subclass of Model

  Handles circular references better

# GUI Examples
# A Button Example

```
Object subclass: #Toggle
   instanceVariableNames: 'isOn '
   classVariableNames: ''
   poolDictionaries: ''
   category: 'Whitney-Examples'
```

An example of using PluggableButtonMorph. This model toggles the color and text of the button when pressed.

Structure:
isOn boolean -- toggle between on and off state

## Class Methods

```
new
   ^super new initialize

openInWorld
   "Toggle openInWorld"
   ^self new openInWorld
```

## Instance Methods

```
initialize
   isOn := true.

isOn
   ^isOn
```

# Instance Methods - Continued

```
changeStateLabel
   ^isOn
      ifTrue:['Turn Off']
      ifFalse:['Turn On']

toggle
   isOn := isOn not.
   self changed: #changeStateLabel

openInWorld
   | window toggleButton |
   window := SystemWindow labelled: 'Toggle Example'.
   window model: self.
   toggleButton := PluggableButtonMorph
      on: self
      getState: #isOn
      action: #toggle
      label: #changeStateLabel
      menu: nil.
   toggleButton
      onColor: Color red
      offColor: Color blue.
   toggleButton
      feedbackColor: Color yellow.
   window
      addMorph: toggleButton
      frame: (0@0 extent: 1@1).
   window openInWorld.
```

## How Does this Work

```
toggleButton := PluggableButtonMorph
    on: self
    getState: #isOn
    action: #toggle
    label: #changeStateLabel
    menu: nil.
```

## PluggableTextMorph

getTextSelector
- Method sent to model to get text for display
- Sent when model changes with getTextSelector as parameter

setTextSelector
- Method sent to model to send it the text
- Sent when user edits the text
- If model return false, changes are rejected
- If model returns true

getSelectionSelector
- Method sent to model to get which text to select
- Sent when model changes with getSelectionSelector

# Text Field Example

```
Object subclass: #TextModelExample
   instanceVariableNames: 'changeCount text '
   classVariableNames: ''
   poolDictionaries: ''
   category: 'Whitney-Examples'
```

This class is an example of using a PluggableTextMorph. When a user changes and saves text in the text field, the text and its reverse is displayed in the text field. A count of the number of changes is shown before the text. The original text saved by the user is highlighted.

Structure:

```
 changeCount   integer   -- number of times user has saved text
 text          String or Text -- text last saved by user
```

## Class Methods

```
new
   ^super new initialize

openInWorld
   "TextModelExample openInWorld"
   ^self new openInWorld
```

## Instance Methods

```
initialize
   changeCount := 0.
   text := 'Starting'
```

```
openInWorld
   "TextModelExample openInWorld"
   | window display |
   window := SystemWindow labelled: 'Text Example'.
   window model: self.
   display := PluggableTextMorph
      on: self
      text: #text
      accept: #text:
      readSelection: #selectTextInterval
      menu: nil.
   window
      addMorph: display
      frame: (0@0 extent: 1@1).
   window openInWorld.

selectTextInterval
   |headerSize |
   headerSize := changeCount printString size + 1.
   ^Interval
      from: headerSize + 1
      to: text size /2 + headerSize

text
   ^changeCount printString , ' ' , text

text: aString
   changeCount := changeCount + 1.
   text :=  aString , aString reversed.
   self changed: #text.
   ^true
```

# PluggableButtonMorph

Important PluggableButtonMorph state

label (label: aString)
Text displayed in the button

model (model: anObject)
When button has been triggered button requests action from model

actionSelector (action: aSymbol)
Method sent to the model when button is triggered

triggerOnMouseDown (triggerOnMouseDown: aBoolean)
If true, button is triggered when button is pressed
If false, button is triggered when button is pressed and released

feedbackColor (feedbackColor: aColor)
Border color when user presses button for feedback

getLabelSelector
Method sent to model to get label.
Allows button label to change

getMenuSelector
Method sent to model to get the menu for the button

getStateSelector
Method sent to get a boolean state of the model
Used to select onColor or offColor for button

askBeforeChanging (askBeforeChanging: aBoolean)
If true button sends #okToChange before actionSelector to model
Model returns true (default) if it is willing to change

# Text & Button Example
# Counter - No GUI

```
Object subclass: #Counter
   instanceVariableNames: 'count '
   classVariableNames: ''
   poolDictionaries: ''
   category: 'Whitney-Examples'
```

## Class Methods

```
new
   ^super new initialize
```

## Instance Methods

```
printOn: aStream
   aStream nextPutAll: count printString

add: amount
   count := count + amount.

decrease
   self add: -1

increase
   self add: 1

initialize
   count := 0
```

# Counter - With GUI

```
Object subclass: #Counter
   instanceVariableNames: 'count '
   classVariableNames: ''
   poolDictionaries: ''
   category: 'Whitney-Examples'
```

## Class Methods

```
new
   ^super new initialize

openInWorld
   ^self new openInWorld
```

## Instance Methods

```
printOn: aStream
   aStream nextPutAll: count printString

add: amount
   count := count + amount.
   self changed: #printString

count: aNumberOrText
   count := aNumberOrText asNumber.
   ^true

decrease
   self add: -1

increase
   self add: 1

initialize
   count := 0
```

# Instance Methods - Continued

```
openInWorld
  | window increaseButton decreaseButton display |
  window := SystemWindow labelled: 'Counter'.
  window model: self.
  increaseButton := PluggableButtonMorph new
    model: self;
    action: #increase;
    label: 'Increase';
    borderWidth: 1.
  decreaseButton := PluggableButtonMorph new
    model: self;
    action: #decrease;
    label: 'Decrease';
    borderWidth: 1.
  display := PluggableTextMorph
    on: self
    text: #printString
    accept: #count:.
  window
    addMorph: display
    frame: (0@0 extent: 1@0.5).
  window
    addMorph: increaseButton
    frame: (0@0.5 extent: 0.5@0.5).
  window
    addMorph: decreaseButton
    frame: (0.5@0.5 extent: 0.5@0.5).
  window openInWorld.
```