Fun With Pharo

Stéphane Ducasse

Version of 2013-03-17

Contents

Chapter 1

Chasing a Shape on the Screen: a.k.a Baby clicking game

In this chapter we propose you to build a small game to train baby to use a mouse. The idea is to have a flashing shape moving on the screen and to click on it to change its direction.

This way we will show you how to change the behavior of a morph and how to add interaction with the morph. Pharo defines a simple circle Morph that we will refine and extent to a new one called FlasherMorph. Doing so you will learn how you can define a class by refining another one and extending its behavior. This example will be then analyzed in the next chapter to explain you what is inheritance, *i.e.,* how can we extend or refine the behavior of a class to obtain other classes with related behavior. The next chapter will answer all the questions that this chapter will raise. Just play the game and follow the instructions for now.

1.1 CircleMorph

We do not want to create a Morph from scratch because this is too complex. Indeed morphs know how to display themselves, get transformed, change color, to react to external events,... In fact what we will do is reuse a class that already implements what we need. For this purpose we will extend the CircleMorph class.

We will define FlasherMorph: a simple morph that flashes, *i.e.,* changes its color at constant rate. The script **??** shows how to create such a sim-

ple morph. Note that Morphs are created with a slightly different creation method. Indeed to create an instance of a Morph we should use the method newStandAlone. This is because Morph is a class that requires special treatment. The method new is the normally the method that should be sent to a class to create new instances.

```
CircleMorph new openInWorld
```

```
| c |
c := CircleMorph new.
c inspect.
c openInWorld
```

```
 | c |
c := CircleMorph new.
c borderWidth: 3.
c color: Color yellow.
c extent: 200@200.
c inspect.
c openInWorld.
```

```
CircleMorph subclass: #Flasher
    instanceVariableNames: ''
    classVariableNames: ''
    poolDictionaries: ''
    category: 'Flasher'
```

```
CircleMorph subclass: #Flasher
    instanceVariableNames: 'on'
    classVariableNames: ''
    poolDictionaries: ''
    category: 'Flasher'
```

```
| c |
c := Flasher new.
c borderWidth: 3.
c color: Color yellow.
c extent: 200@200.
c inspect.
c openInWorld.
```

```
step

    on
        ifTrue: [self color: Color black]
        ifFalse: [self color: Color white].
    on := on not.
```

```
step

    ('here: ', on printString) crLog.
    on
        ifTrue: [self color: Color black]
        ifFalse: [self color: Color white].
    on := on not.
```

```
step

    ('here: ', on printString) crLog.
    on
        ifTrue: [self color: Color black]
        ifFalse: [self color: Color white].
    self changed.
    on := on not.
```

```
step

    on
        ifTrue: [self color: Color black]
        ifFalse: [self color: Color white].
    self changed.
    on := on not.
```

```
stepTime

    ^ 500
```

```
Flasher subclass: #MultiColorFlasher
    instanceVariableNames: ''
    classVariableNames: ''
    poolDictionaries: ''
    category: 'Flasher'
```

```
MultiColorFlasher>>step

    color := Color random.
    self changed.
```

but we do not need on.

```
CircleMorph subclass: #Flasher
    instanceVariableNames: ''
    classVariableNames: ''
    poolDictionaries: ''
    category: 'Flasher'
```

```
Flasher>>initialize

    super initialize.
    self borderWidth: 3.
    self color: Color yellow.
    self extent: 100@100
```

```
stepTime

    ^ 500
```

```
Flasher subclass: #BinaryFlasher
    instanceVariableNames: 'on'
    classVariableNames: ''
    poolDictionaries: ''
    category: 'Flasher'
```

```
BinaryFlasher>>initialize

    super initialize.
    on := true
```

```
BinaryFlasher>>step

    ('here: ', on printString) crLog.
    on
        ifTrue: [self color: Color black]
        ifFalse: [self color: Color white].
    self changed.
    on := on not.
```

```
Flasher subclass: #MultiColorFlasher
    instanceVariableNames: ''
    classVariableNames: ''
    poolDictionaries: ''
    category: 'Flasher'
```

```
MultiColorFlasher>>step

    self color: Color random.
    self changed.
```

1.2 Non linear flashing

```
CircleMorph subclass: #Flasher
    instanceVariableNames: 'time'
    classVariableNames: ''
    poolDictionaries: ''
    category: 'Flasher'
```

```
Flasher>>initialize

    super initialize.
    self borderWidth: 3.
    self color: Color yellow.
    self extent: 100@100.
    time := 1000.
```

```
Flasher>>step

    time := time - 50.
    time < 10
        ifTrue: [time := 1000 ]
```

```
Flasher>>step

    time := time - 50.
    time < 10
        ifTrue: [time := 1000 ]
```

```
Flasher>>step

    time printString crLog.

    time := time - 50.
    (time between: 10 and: 200)
        ifTrue: [ time := time - 10].
    time < 10
        ifTrue: [time := 500]
```

```
Flasher>>step

    time printString crLog.

    time := time - increment.
    (time between: 10 and: 200)
        ifTrue: [ increment := 10].
    time < 10
        ifTrue: [
            time := 500.
```

```
        increment := 50.
        ]
```

Cleaning Logic a bit

```
Flasher>>initialize
    super initialize.
    self borderWidth: 3.
    self color: Color yellow.
    self extent: 100 @ 100.
    self initializeAnimationTime
```

```
Flasher>>initializeAnimationTime
    time := 600.
    increment := 50
```

```
Flasher>>step

    time printString crLog.

    time := time − increment.
    (time between: 10 and: 200)
        ifTrue: [ increment := 10].
    time < 10
        ifTrue: [ self initializeAnimationTime ]
```