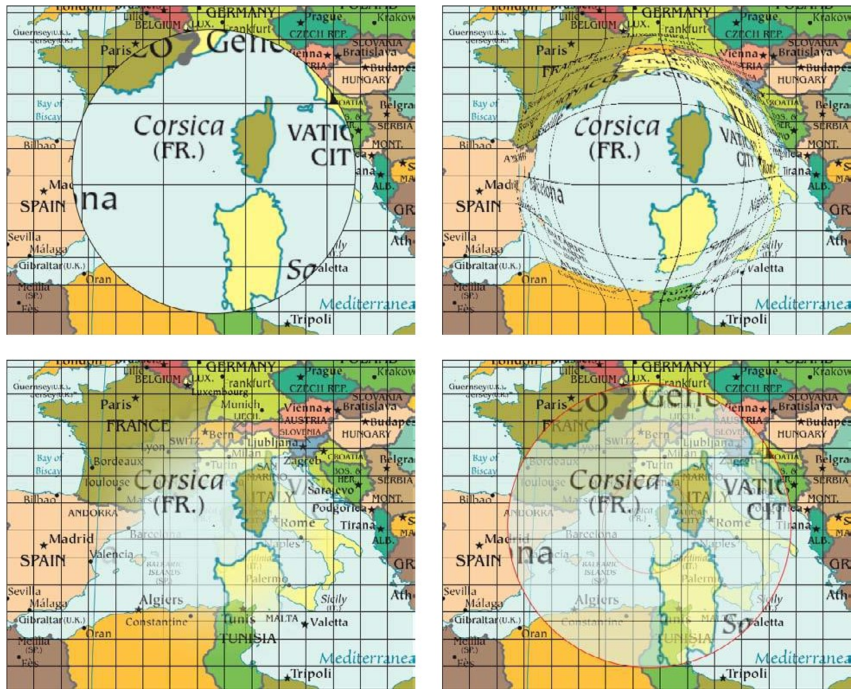
Thibault Raffaillac

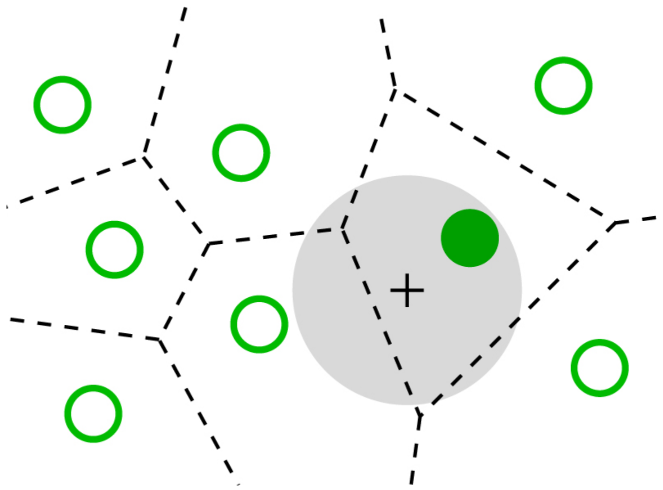
Language and System Support for Interaction

Supervised by Stéphane Huot and Stéphane Ducasse

Centre de Recherche en Informatique, Signal et Automatique de Lille

Prototyping interaction techniques


Sigma Lenses (Pietriga and Appert)

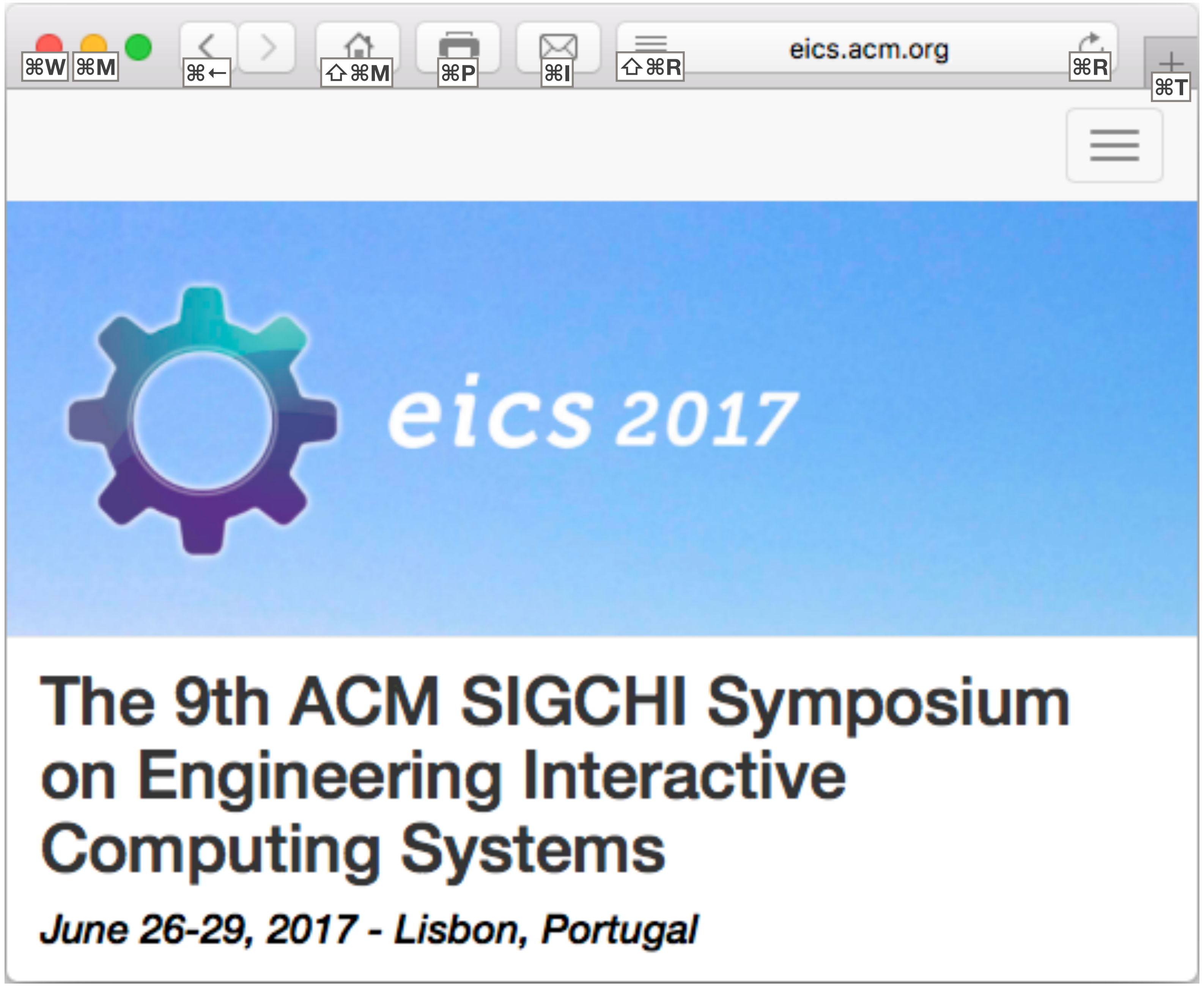

Bubble Cursor (Grossman and Balakrishnan)

CHI 2013
⌘W ⌘M
⌘←
⌘D
⌘⇧H
⌘T
⌘⌥2
chi2013
Welcome
Program
Attending
alt
Ctrl
ExposeHK (Malacria et. al)

Challenging the architectures of frameworks

Requiring developers to use non-standard code

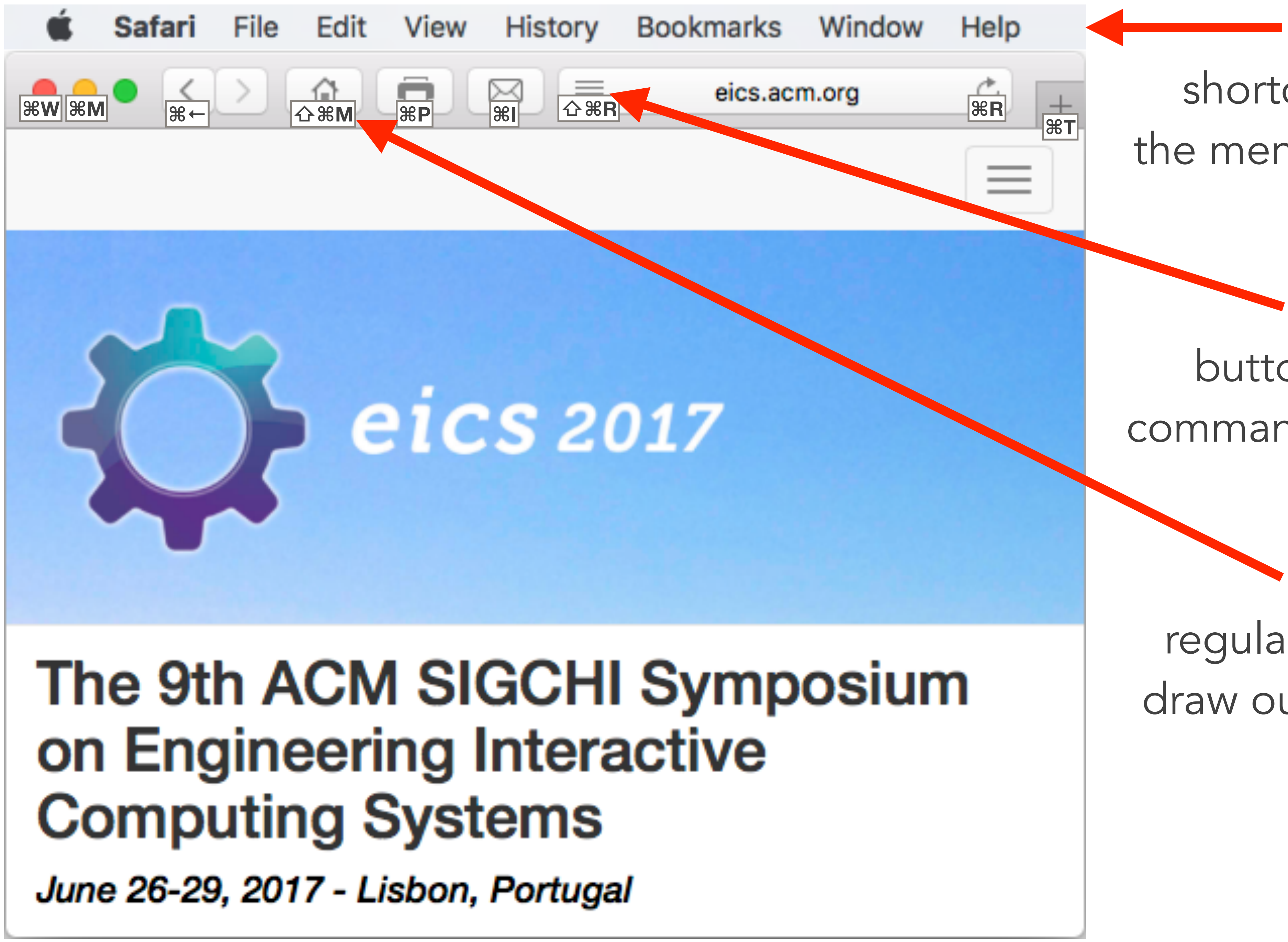
Prototyping interaction techniques


⌘W
⌘M
⌘←
⇧⌘M
⌘P
⌘I
⇧⌘R
eics.acm.org
⌘R
⌘T
eics 2017
The 9th ACM SIGCHI Symposium on Engineering Interactive Computing Systems
June 26-29, 2017 - Lisbon, Portugal

⌘W
⌘M
CHI 2013
⌘←
⌘D
⌘⇧H
⌘T
⌘⌥2
chi2013
Welcome
Program
Attending
alt
Ctrl

ExposeHK (Malacria et. al)

Prototyping interaction techniques



The 9th ACM SIGCHI Symposium on Engineering Interactive Computing Systems

June 26-29, 2017 - Lisbon, Portugal

Problem:
shortcuts are stored in the menus, not the buttons

Problem:
buttons may execute commands *after* some code

Problem:
regular buttons may not draw outside their bounds

Problems

Simple interaction ideas are *not* simple to implement.

Frameworks describe *interfaces* rather than *interaction*.

Lack of tools to introspect and edit live interfaces.

Lack of knowledge on how users hack these systems.

Thesis context: Pharo Smalltalk

Live programming environment, pure object-oriented

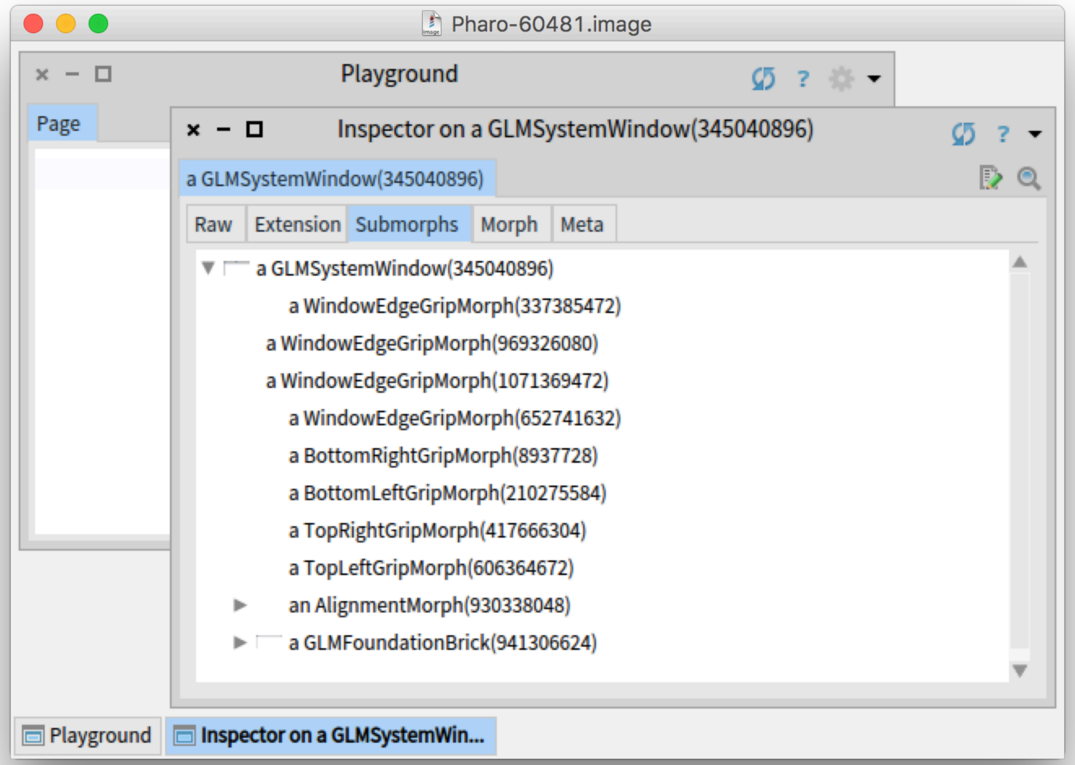
Supports prototyping with *introspection* (inspect objects)

Ageing interface (Morphic)

Successor (Bloc) being developed *elsewhere*

$\downarrow$

Contribute *indirectly*


Pharo-60481.image
Playground
Page
Inspector on a GLMSystemWindow(345040896)
a GLMSystemWindow(345040896)
Raw
Extension
Submorphs
Morph
Meta
Playground
Inspector on a GLMSystemWin...

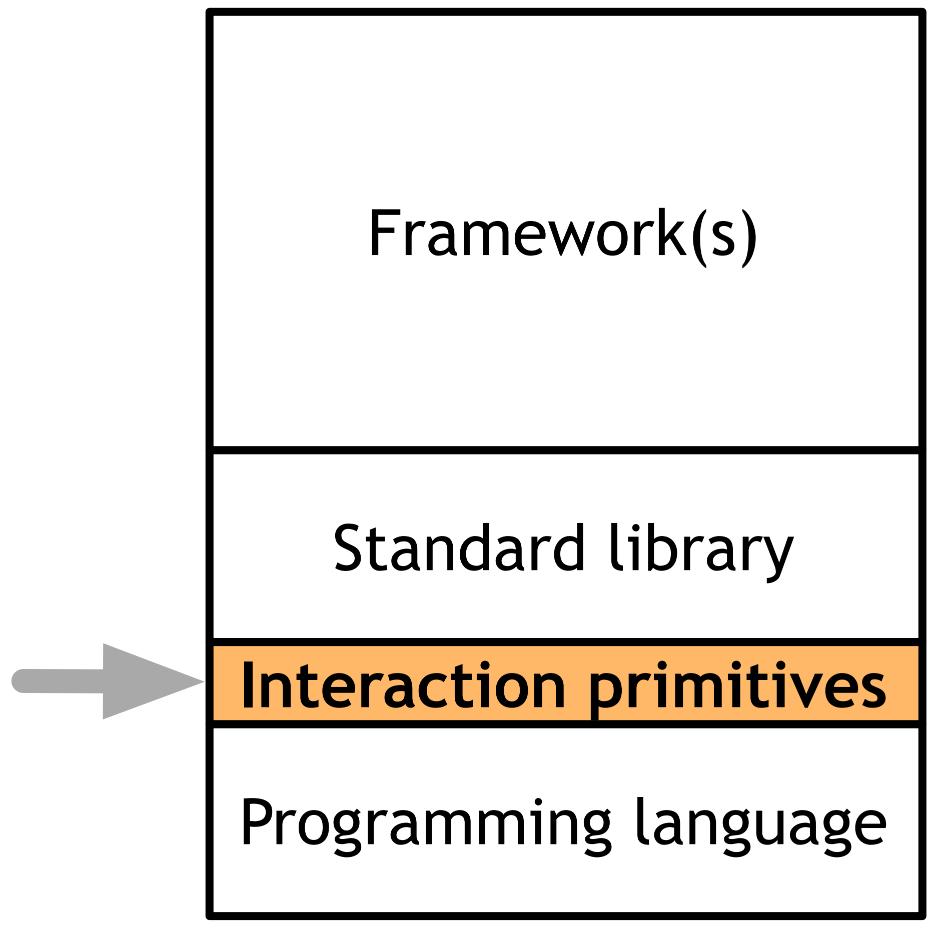
```
▼ a GLMSystemWindow(345040896)
    a WindowEdgeGripMorph(337385472)
  a WindowEdgeGripMorph(969326080)
  a WindowEdgeGripMorph(1071369472)
    a WindowEdgeGripMorph(652741632)
    a BottomRightGripMorph(8937728)
    a BottomLeftGripMorph(210275584)
    a TopRightGripMorph(417666304)
    a TopLeftGripMorph(606364672)
  ▶ an AlignmentMorph(930338048)
  ▶ a GLMFoundationBrick(941306624)
```

Thesis context: Pharo Smalltalk

Working on language and system below frameworks

Taking advantage of Pharo's **reflectivity** (access and modify language structures)

In close contact with its core developers



Goal:

Improve the flexibility of interaction frameworks, *indirectly*

Plan

1. What can we add to languages to support interaction?
2. How can we make interaction a 1st class object?
3. What do designers of new techniques need?

Plan

1. What can we add to languages to support interaction?
2. How can we make interaction a 1st class object?
3. What do designers of new techniques need?

Creating language primitives

Cement and simplify established practices

Reduce frameworks' complexity

Remain consistent across applications

Evolve languages towards interactivity

Generative (unexpected uses)

Creating language primitives

```
[object property: value] during: 2 seconds
```

➡ How do I validate it being "simpler"?

```
[mouse click] afterDo: [object doSomething]
```

➡ What is a standard way to design this properly?

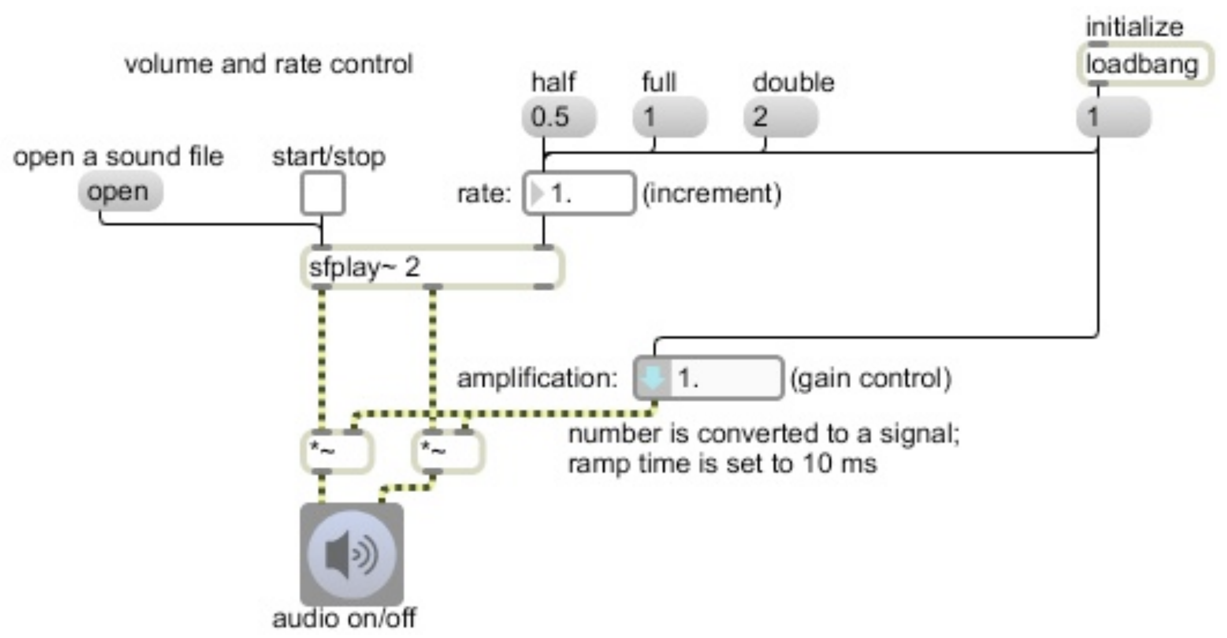
Plan

1. What can we add to languages to support interaction?
2. How can we make interaction a 1st class object?
3. What do designers of new techniques need?

First class objects

- In programming: first class citizens support common operations on variables (assignment, pass/return with function, modification)
- In HCI: open to interpretation

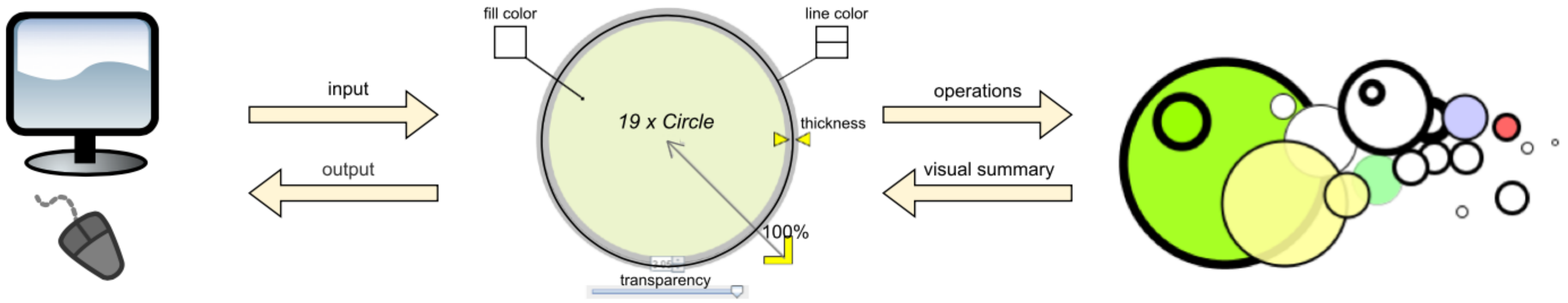
First class objects



Max/MSP

StickyLines (Ciolfi Felice et. al)

Google Spreadsheet



Surrogate Objects (Kwon et. al)

Characterizing “first-class”

Captures attention while in interaction

Revealed as normally invisible

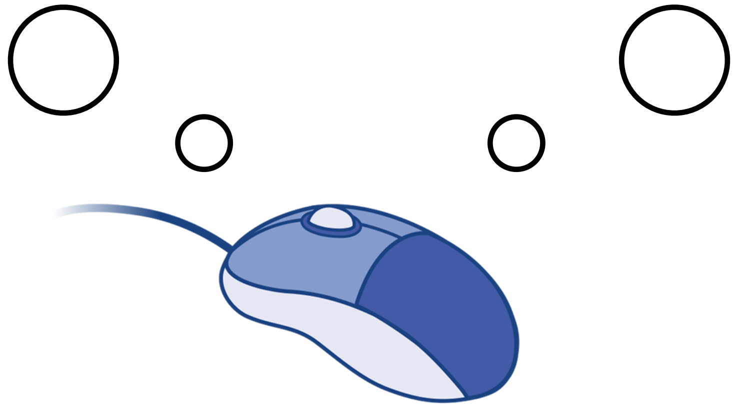
Available everywhere

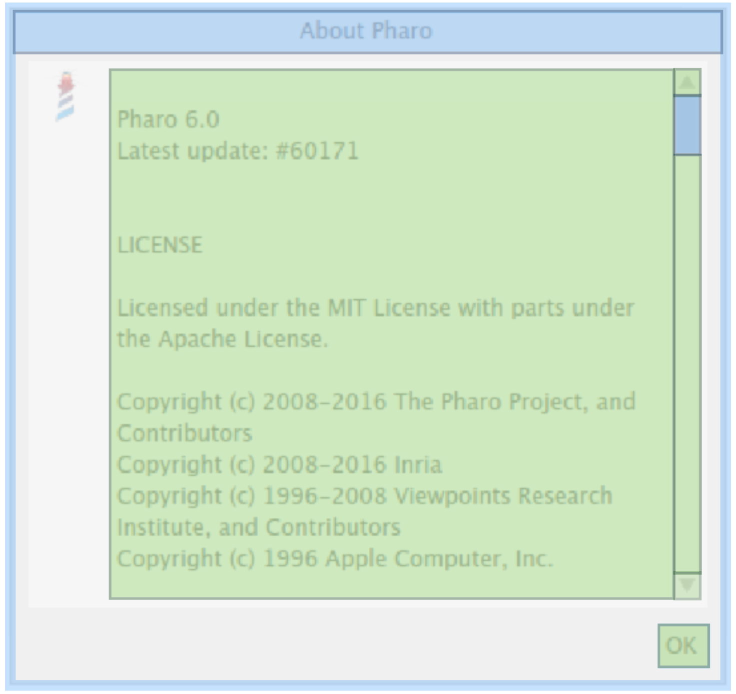
Shifts the point of view to itself

Provides advanced functions

When programming interaction

```
class Mouse {
  float dx, dy;
  bool[] buttons;
  float dpi;
};
```





About Pharo

Pharo 6.0
Latest update: #60171

LICENSE

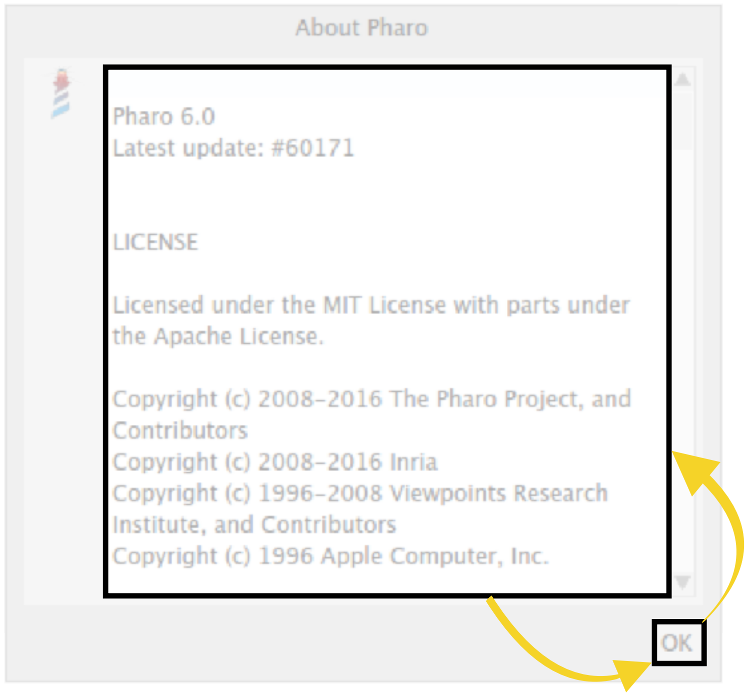
Licensed under the MIT License with parts under the Apache License.

Copyright (c) 2008-2016 The Pharo Project, and Contributors
Copyright (c) 2008-2016 Inria
Copyright (c) 1996-2008 Viewpoints Research Institute, and Contributors
Copyright (c) 1996 Apple Computer, Inc.

OK

```
⌘W, ⌘M, ⌘←, ...
```

```
class Keyboard {
  bool[] keys;
  bool[] modifiers;
  float backlight;
};
```



About Pharo

Pharo 6.0
Latest update: #60171

LICENSE

Licensed under the MIT License with parts under the Apache License.

Copyright (c) 2008-2016 The Pharo Project, and Contributors
Copyright (c) 2008-2016 Inria
Copyright (c) 1996-2008 Viewpoints Research Institute, and Contributors
Copyright (c) 1996 Apple Computer, Inc.

OK

Plan

1. What can we add to languages to support interaction?
2. How can we make interaction a 1st class object?
3. What do designers of new techniques need?

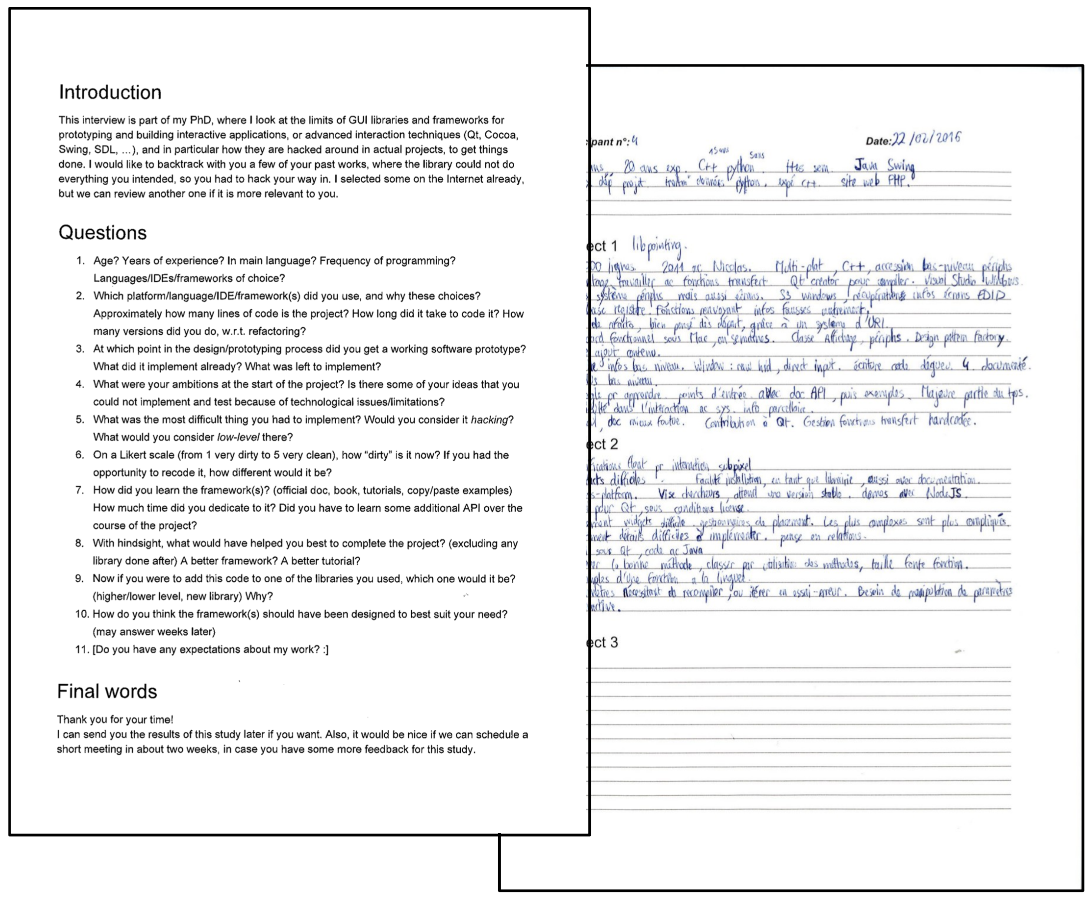
Interviews

8 semi-structured interviews of researchers who prototyped advanced interaction techniques

Feb - May 2016

8h of audio recordings

Questions seeking problems encountered at every stages of their projects



Introduction

This interview is part of my PhD, where I look at the limits of GUI libraries and frameworks for prototyping and building interactive applications, or advanced interaction techniques (Qt, Cocoa, Swing, SDL, ...), and in particular how they are hacked around in actual projects, to get things done. I would like to backtrack with you a few of your past works, where the library could not do everything you intended, so you had to hack your way in. I selected some on the Internet already, but we can review another one if it is more relevant to you.

Questions

1. Age? Years of experience? In main language? Frequency of programming? Languages/IDEs/frameworks of choice?
2. Which platform/language/IDE/framework(s) did you use, and why these choices? Approximately how many lines of code is the project? How long did it take to code it? How many versions did you do, w.r.t. refactoring?
3. At which point in the design/prototyping process did you get a working software prototype? What did it implement already? What was left to implement?
4. What were your ambitions at the start of the project? Is there some of your ideas that you could not implement and test because of technological issues/limitations?
5. What was the most difficult thing you had to implement? Would you consider it *hacking*? What would you consider *low-level* there?
6. On a Likert scale (from 1 very dirty to 5 very clean), how "dirty" is it now? If you had the opportunity to recode it, how different would it be?
7. How did you learn the framework(s)? (official doc, book, tutorials, copy/paste examples) How much time did you dedicate to it? Did you have to learn some additional API over the course of the project?
8. With hindsight, what would have helped you best to complete the project? (excluding any library done after) A better framework? A better tutorial?
9. Now if you were to add this code to one of the libraries you used, which one would it be? (higher/lower level, new library) Why?
10. How do you think the framework(s) should have been designed to best suit your need? (may answer weeks later)
11. [Do you have any expectations about my work? :]

Final words

Thank you for your time!
I can send you the results of this study later if you want. Also, it would be nice if we can schedule a short meeting in about two weeks, in case you have some more feedback for this study.

pant n°: 4 Date: 22/02/2016

[illegible]

ect 1 [illegible]

ect 2 [illegible]

ect 3

Interviews

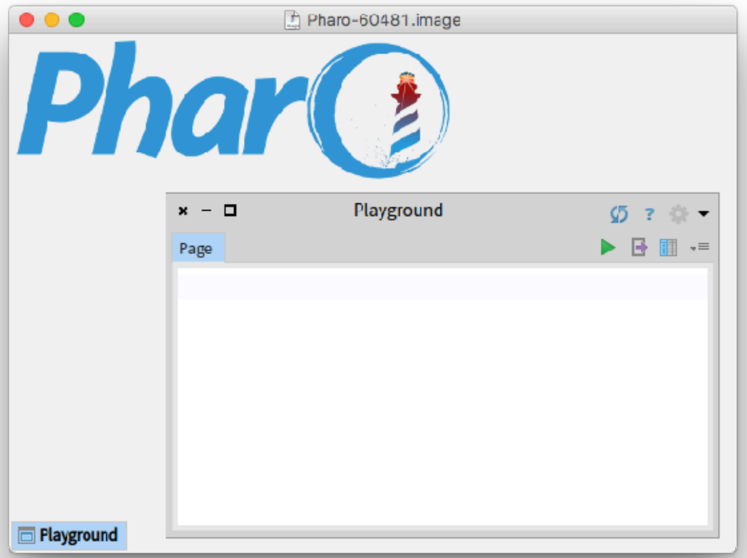
Initially could not make sense of the data (*What to look for?*) and gave up

After doing bibliography, committed to analyse along:

- Problems faced
- Explicit needs
- Tools deemed useful
- Hacking strategies

Course of action

- July - September: analysis of interviews
- October - December: Validation of delay operator
- October - April: Exploration of listener operator



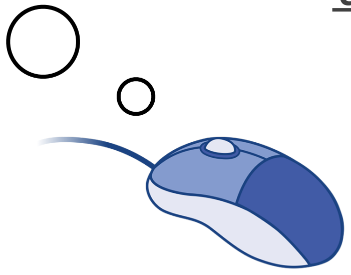
```
[object property: value] during: 2 seconds

[mouse click] afterDo: [object doSomething]
```

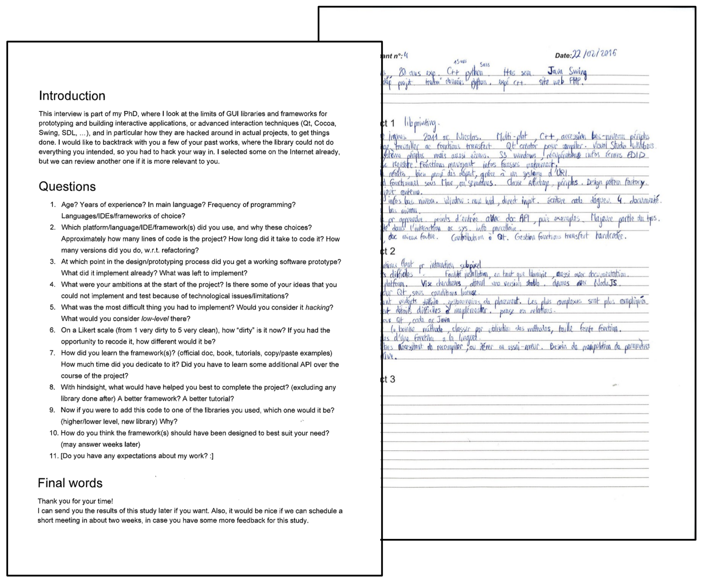
Thank you for your attention!

Thibault Raffaillac
thibault.raffaillac@inria.fr

```
class Mouse {
  float dx, dy;
  bool[] buttons;
  float dpi;
};
```



```
class Keyboard {
  bool[] keys;
  bool[] modifiers;
  float backlight;
};
```



Introduction

This interview is part of my PhD, where I look at the limits of GUI libraries and frameworks for prototyping and building interactive applications, or advanced interaction techniques (Qt, Cocoa, Swing, SDL, ...), and in particular how they are hacked around in actual projects, to get things done. I would like to backtrack with you a few of your past works, where the library could not do everything you intended, so you had to hack your way in. I selected some on the Internet already, but we can review another one if it is more relevant to you.

Questions

1. Age? Years of experience? In main language? Frequency of programming? Languages/IDEs/frameworks of choice?
2. Which platform/language/IDE/framework(s) did you use, and why these choices? Approximately how many lines of code is the project? How long did it take to code it? How many versions did you do, w.r.t. refactoring?
3. At which point in the design/prototyping process did you get a working software prototype? What did it implement already? What was left to implement?
4. What were your ambitions at the start of the project? Is there some of your ideas that you could not implement and test because of technological issues/limitations?
5. What was the most difficult thing you had to implement? Would you consider it *hacking*? What would you consider *low-level* there?
6. On a Likert scale (from 1 very dirty to 5 very clean), how "dirty" is it now? If you had the opportunity to recode it, how different would it be?
7. How did you learn the framework(s)? (official doc, book, tutorials, copy/paste examples) How much time did you dedicate to it? Did you have to learn some additional API over the course of the project?
8. With hindsight, what would have helped you best to complete the project? (excluding any library done after) A better framework? A better tutorial?
9. Now if you were to add this code to one of the libraries you used, which one would it be? (higher/lower level, new library) Why?
10. How do you think the framework(s) should have been designed to best suit your need? (may answer weeks later)
11. [Do you have any expectations about my work? :]

Final words

Thank you for your time!
I can send you the results of this study later if you want. Also, it would be nice if we can schedule a short meeting in about two weeks, in case you have some more feedback for this study.

[illegible]