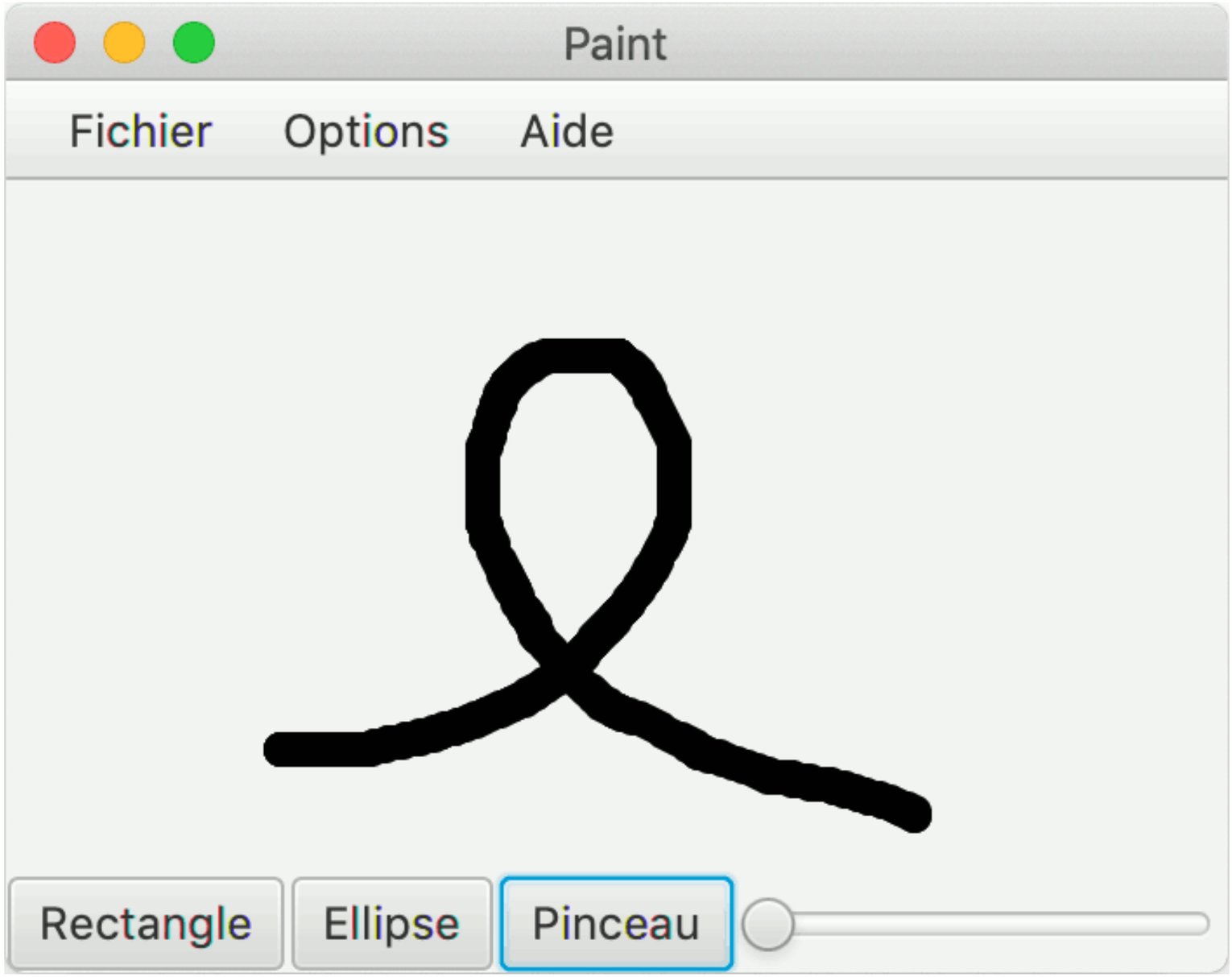
Améliorer les langages et bibliothèques logicielles pour programmer l'interaction

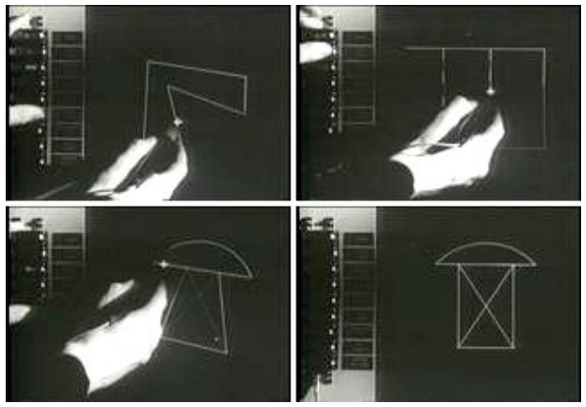
Thibault Raffaillac

Rapporteurs : Sophie Dupuy-Chessa (Université Grenoble Alpes
Éric Lecolinet (Télécom ParisTech)
Examinateurs : Stéphane Conversy (ÉNAC)
Laurence Duchien (Université de Lille)
Emmanuel Pietriga (Inria Saclay)
Directeur de thèse : Stéphane Huot (Inria Lille)

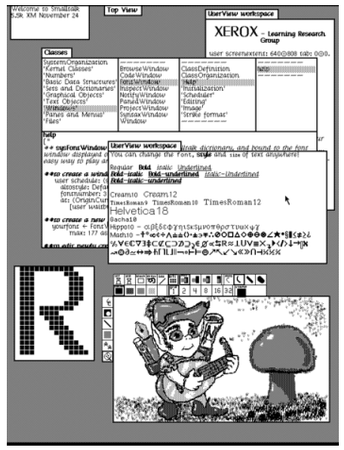
Les interfaces graphiques



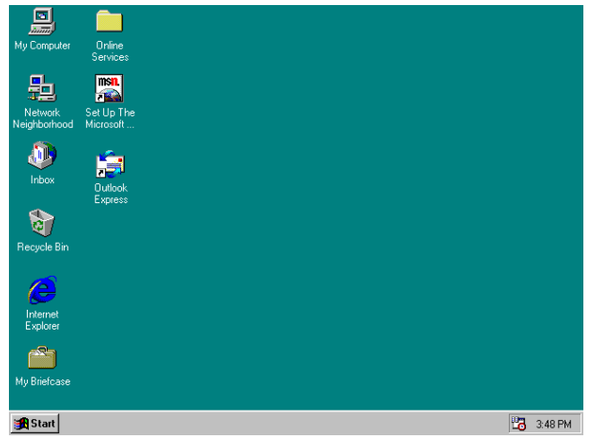
Histoire des interfaces graphiques



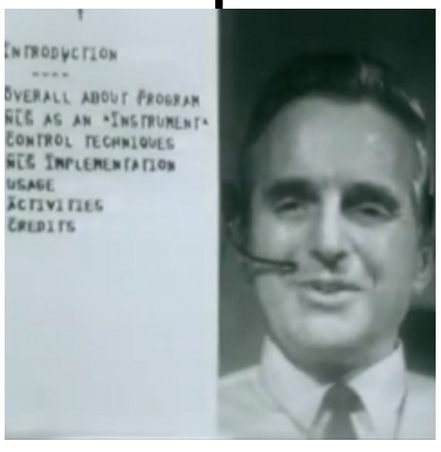
Sketchpad (1963)



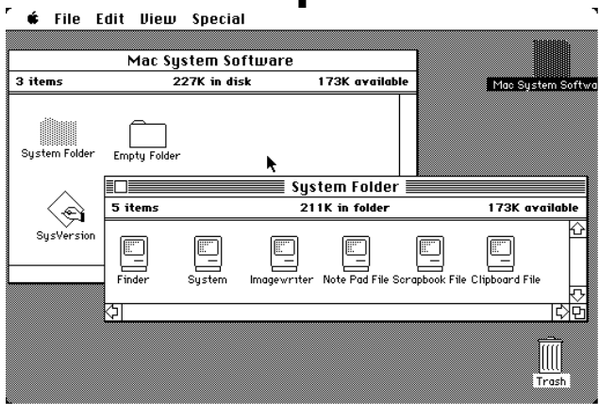
Xerox Alto (1972)
Métaphore bureau



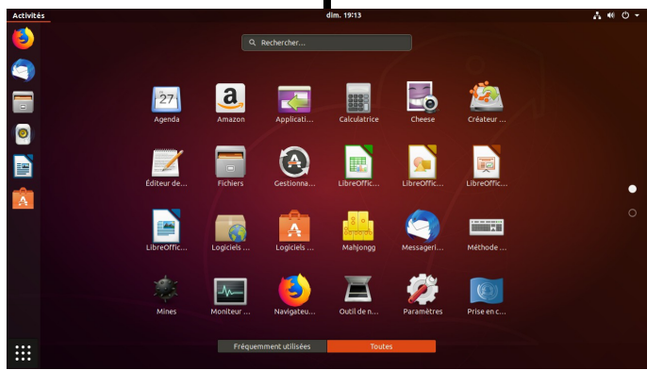
Microsoft Windows 95



Mother of All Demos (1968)
Fenêtres, souris, visio, ...



Apple Macintosh (1984)

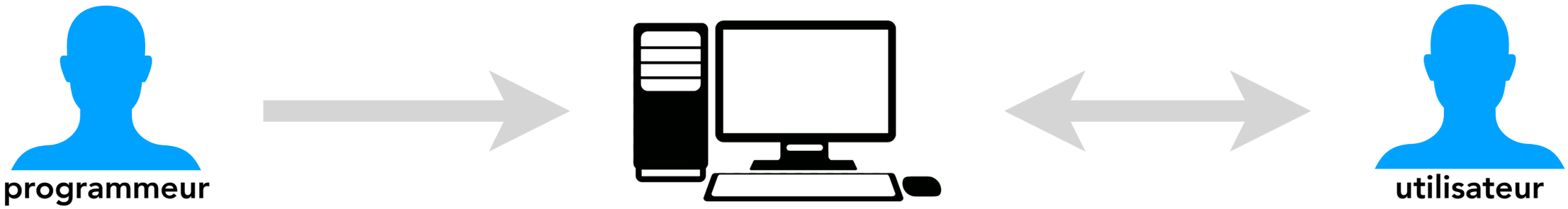


Linux Ubuntu (2004)

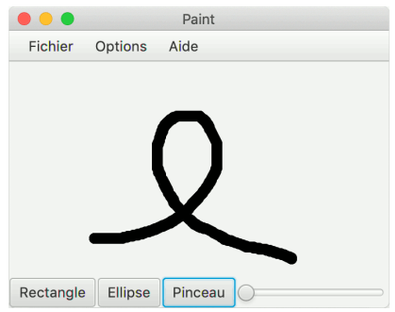
Des interfaces à l'interaction

Exploration

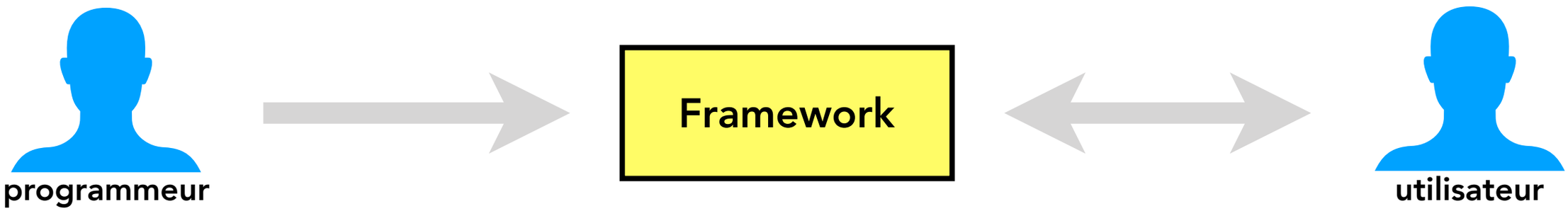
Programmation d'interfaces

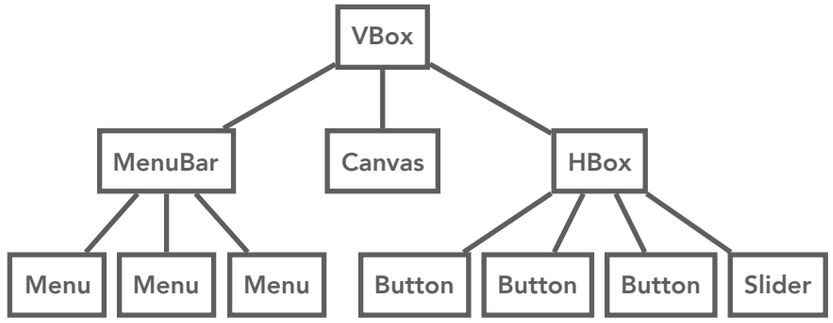


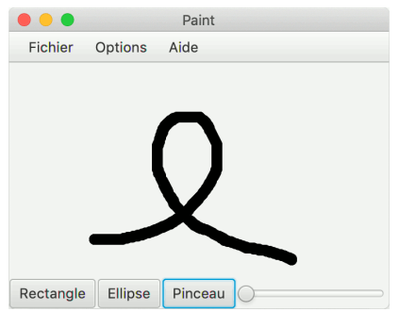

```java
import javafx.application.Application;
import javafx.geometry.Pos;
import javafx.scene.Scene;
import javafx.scene.canvas.Canvas;
import javafx.scene.canvas.GraphicsContext;
import javafx.scene.control.Button;
import javafx.scene.control.Menu;
import javafx.scene.control.MenuBar;
import javafx.scene.control.Slider;
import javafx.scene.layout.VBox;
import javafx.scene.layout.HBox;
import javafx.scene.shape.StrokeLineCap;
import javafx.stage.Stage;

public class Exemple extends Application {
   public void start(Stage stage) {
      Canvas canvas = new Canvas(270, 200);
      GraphicsContext gc = canvas.getGraphicsContext2D();
      gc.setLineWidth(10.0);
      gc.setLineCap(StrokeLineCap.ROUND);
      canvas.setOnMouseDragged(e -> {
         gc.fillOval(e.getX(), e.getY(), 10, 10);
      });

      HBox bottom = new HBox(2,
         new Button("Rectangle"),
         new Button("Ellipse"),
         new Button("Pinceau"),
         new Slider());
      bottom.setAlignment(Pos.CENTER);

      stage.setScene(new Scene(new VBox(
         new MenuBar(
            new Menu("Fichier"),
            new Menu("Options"),
            new Menu("Aide")),
         canvas,
         bottom)));
      stage.setTitle("Paint");
      stage.show();
   }
}
```



Programmation d'interfaces

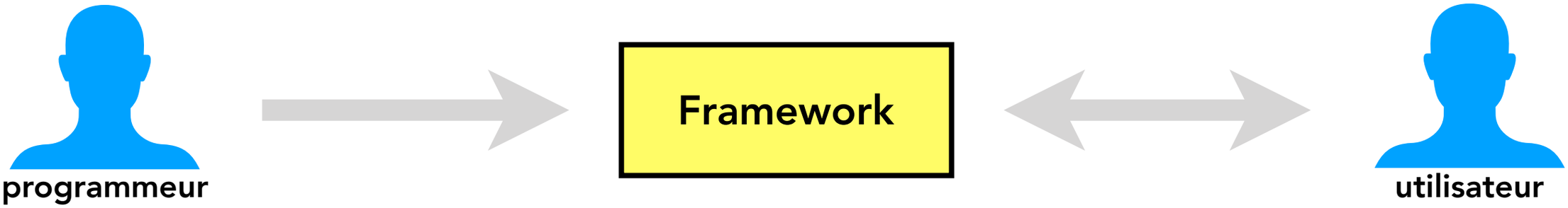


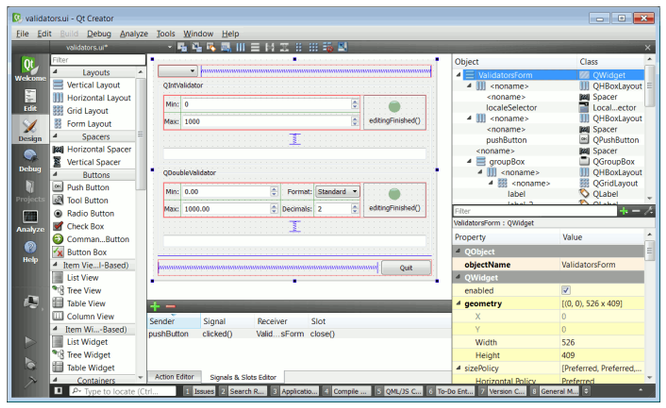




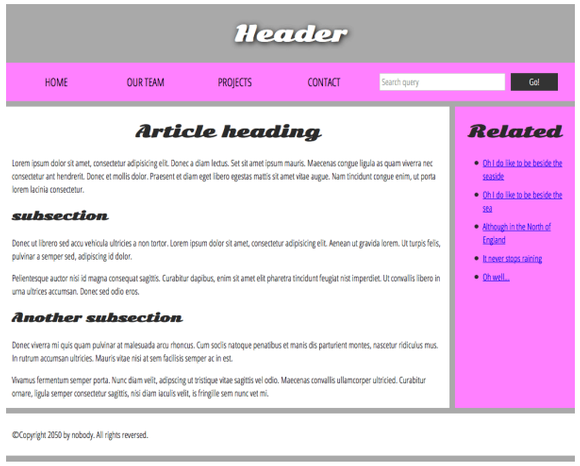

```
/**
 * Simple HelloButton() method.
 * @version 1.0
 * @author john doe <doe.j@example.com>
 */
HelloButton()
{
  JButton hello = new JButton( "Hello, wor
  hello.addActionListener( new HelloBtnList

  // use the JFrame type until support for t
  // new component is finished
  JFrame frame = new JFrame( "Hello Button"
  Container pane = frame.getContentPane();
  pane.add( hello );
  frame.pack();
  frame.show();           // display the fra
}
```

Programmation d'interfaces

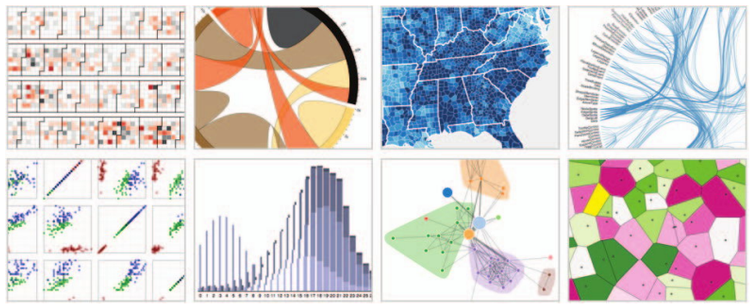




Qt



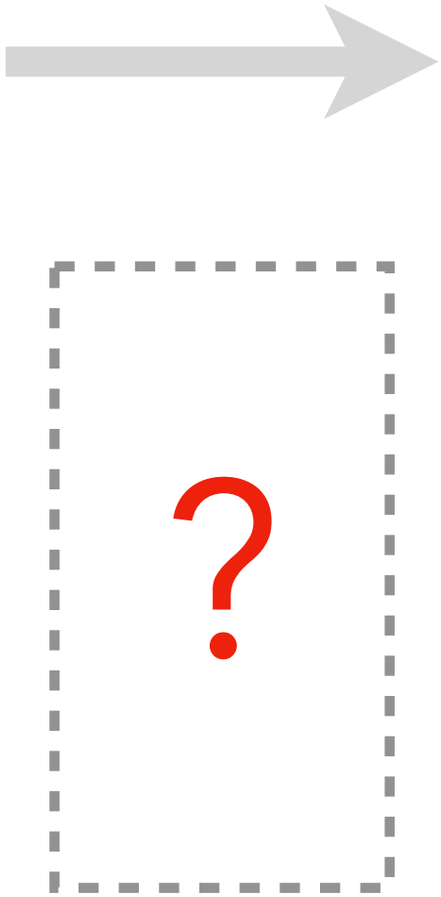
HTML/CSS

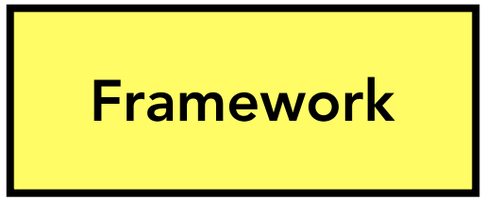


D3

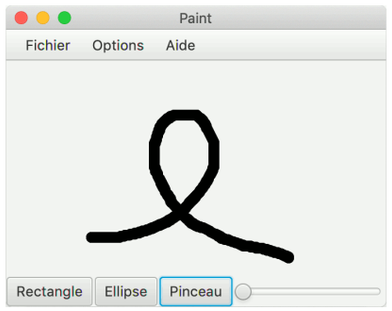
Programmation d'interactions

programmeur

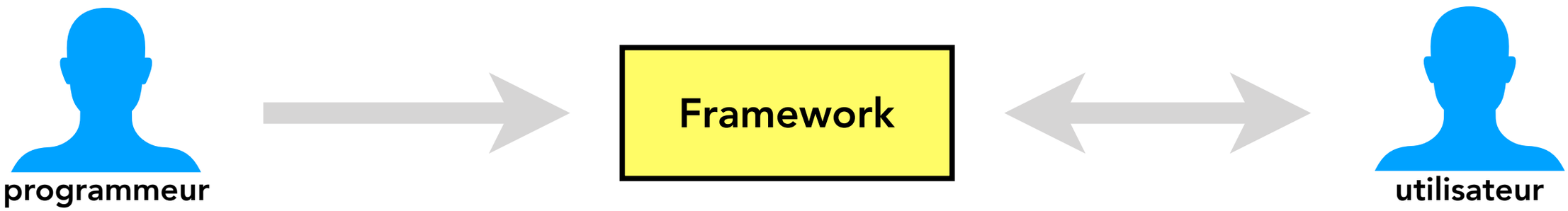


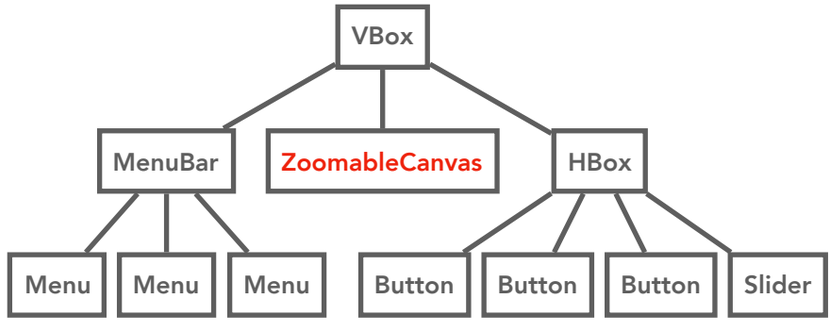


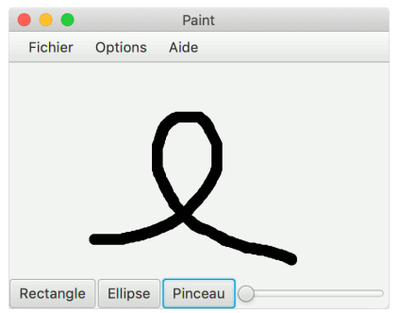
utilisateur

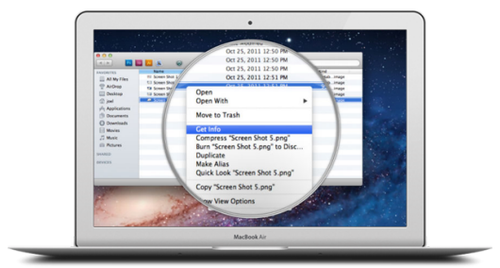


Extension de framework

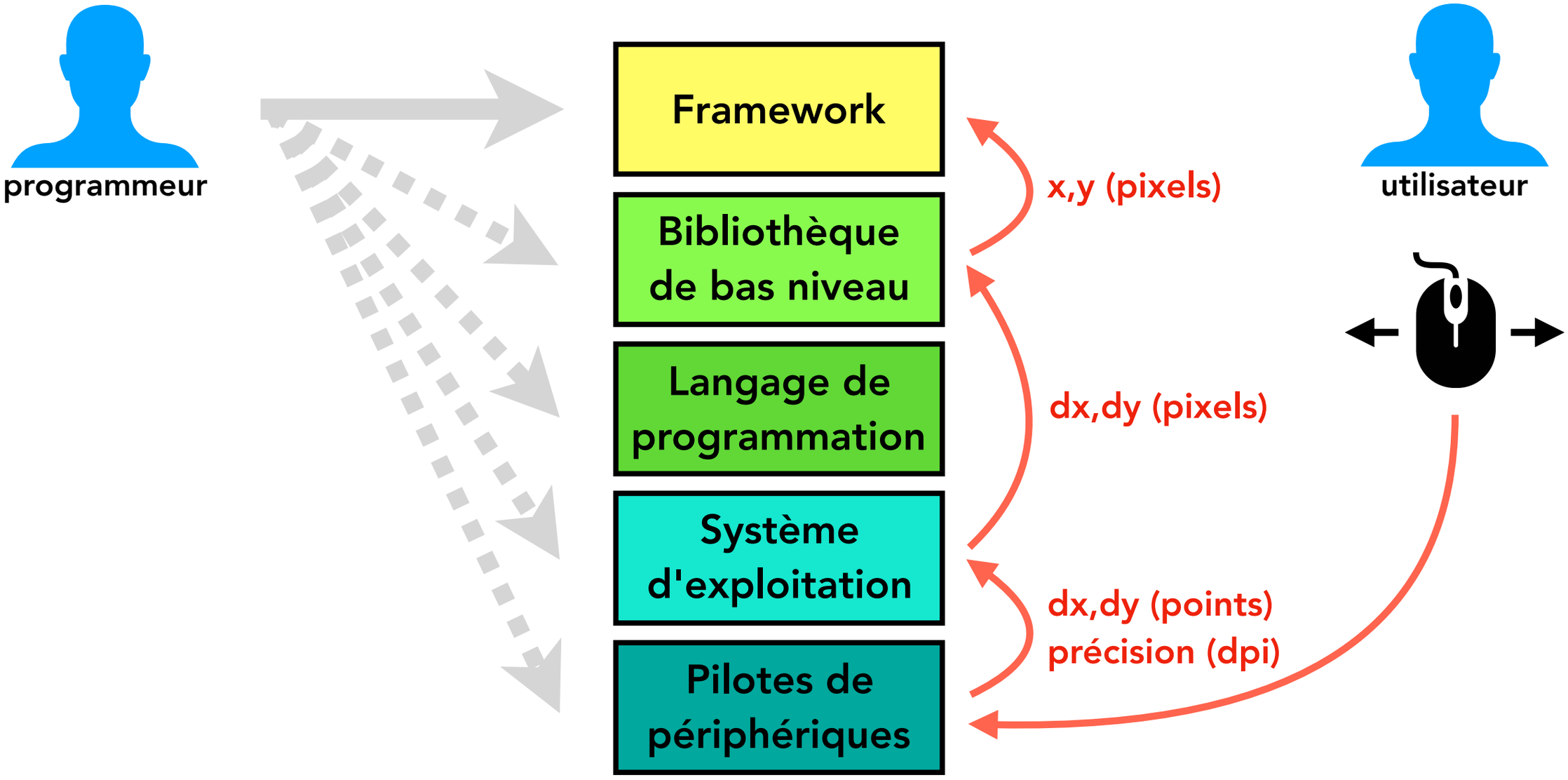




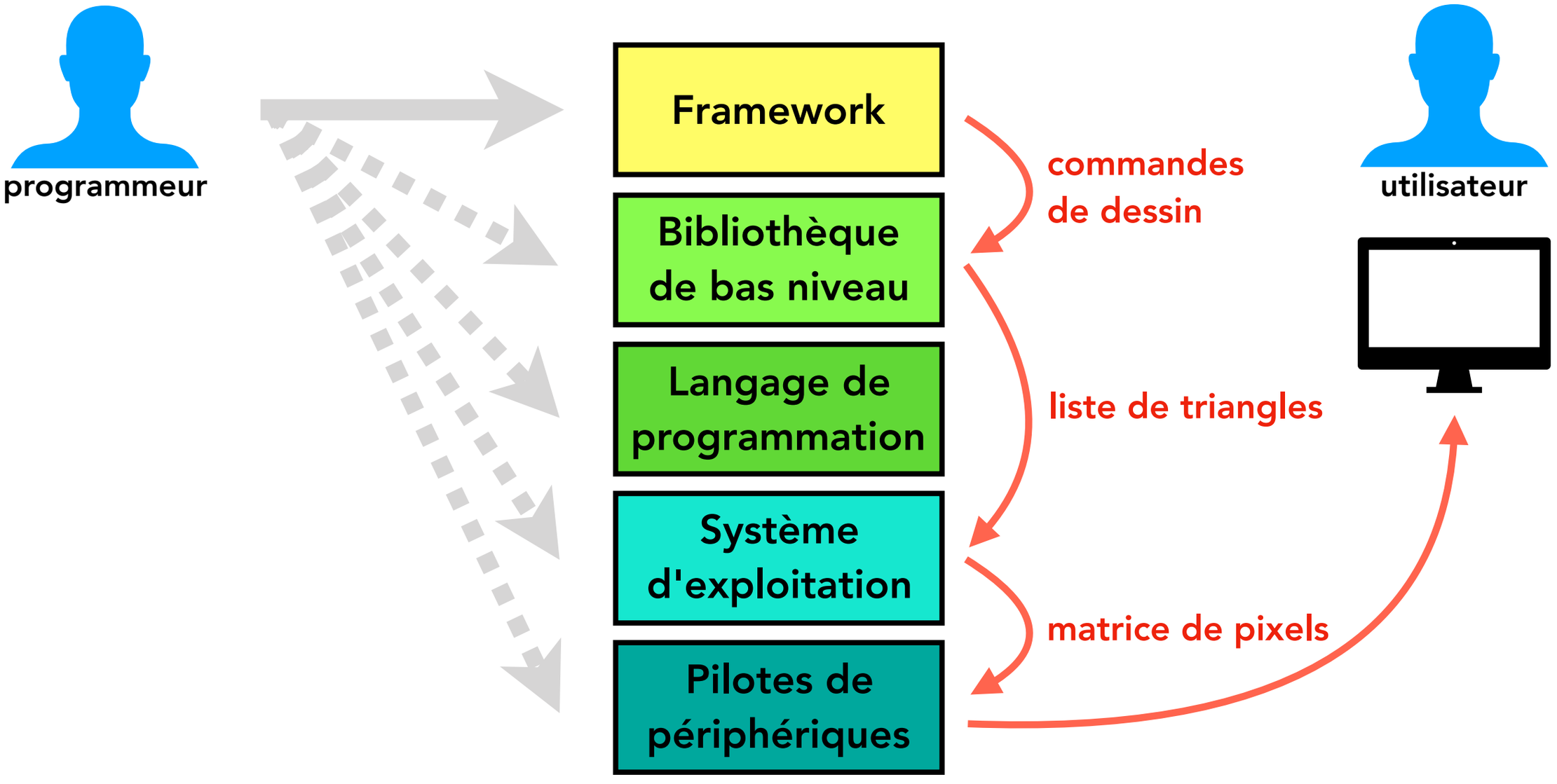




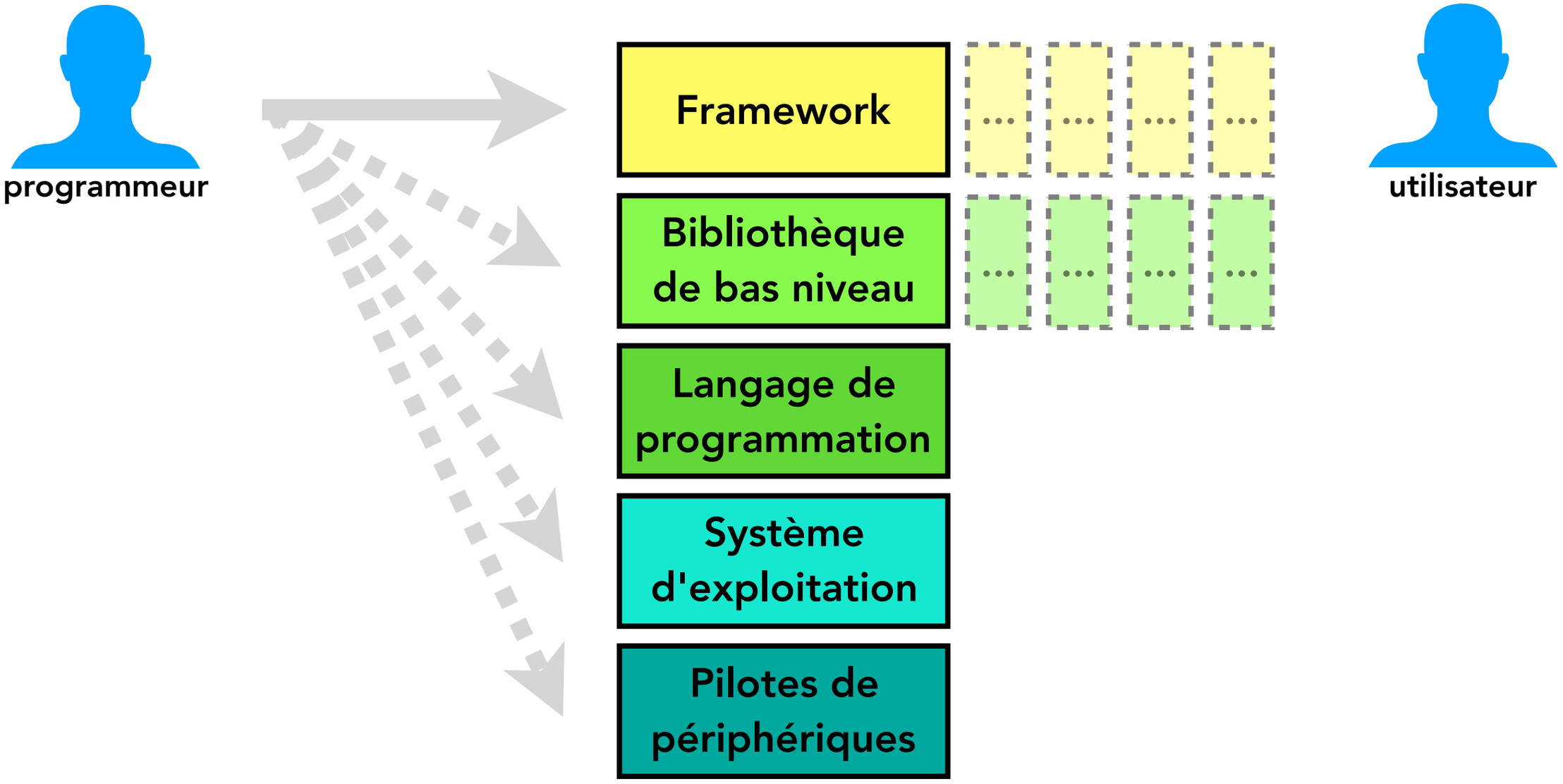
Structure en couches



Structure en couches

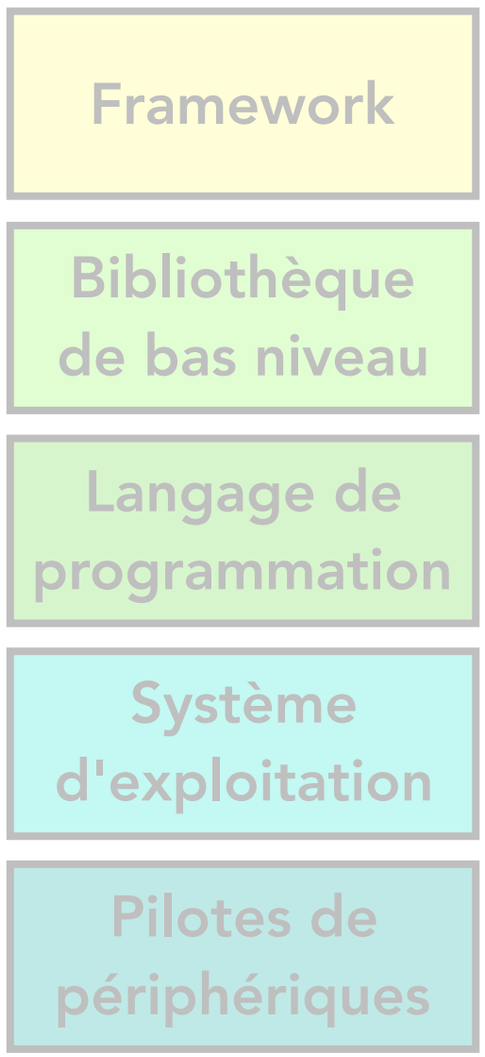


Alternatives multiples



Contributions et plan

1. Études préliminaires



3. Le framework Polyphony

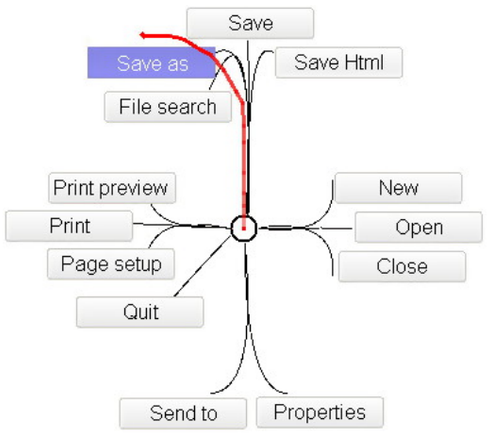
2. Animation de fonctions

Études préliminaires

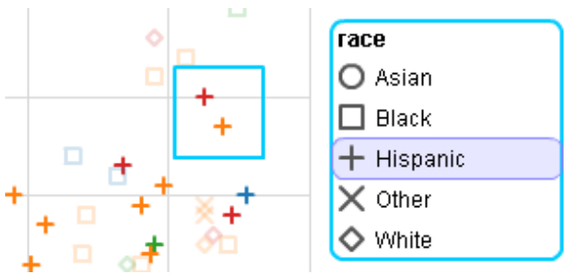
Population : chercheurs en Interactions Homme-Machine

Objet : développement de nouvelles *techniques d'interaction* (combinaison d'éléments logiciels et matériels permettant aux utilisateurs d'accomplir une certaine tâche)

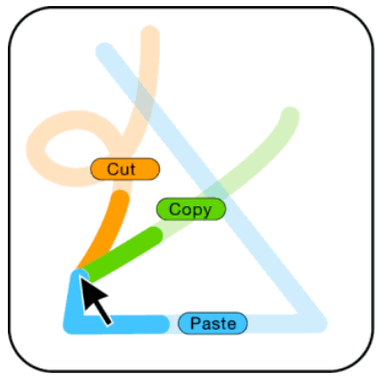
pointage



menus



sélection



apprentissage

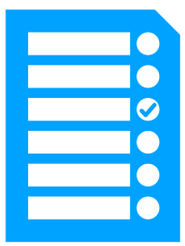
Interviews de chercheurs

Q1 : Comment développent-ils une nouvelle technique d'interaction ?

Q2 : Quels problèmes ont-ils avec les bibliothèques de programmation ?

Q3 : Comment contribuer au mieux à leur travail ?

9 participants

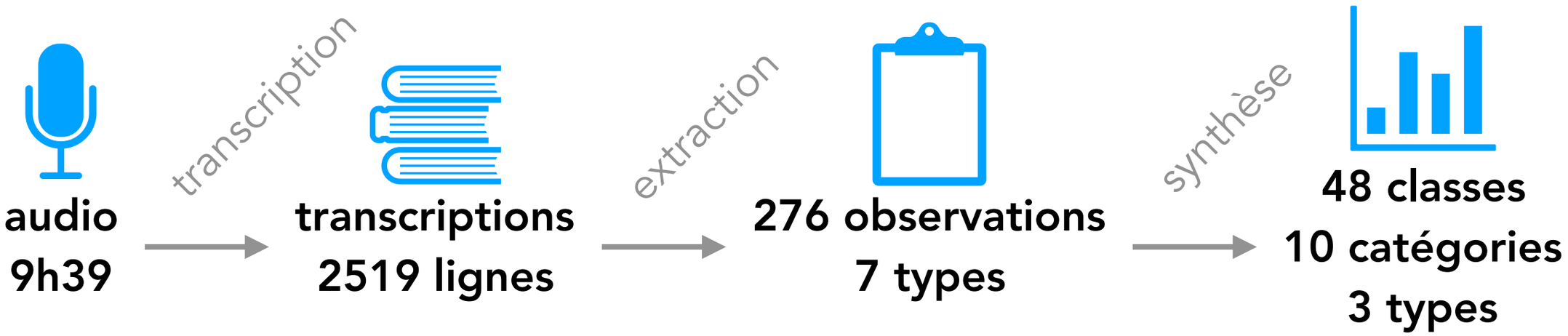


interviews semi-structurées

2~3 projets

problèmes et outils

Analyse des résultats



Les outils issus de la recherche sont peu représentés

Beaucoup d'efforts (et difficultés) de documentation

Combinaisons d'outils non anticipés

Détournements de conventions et principes génériques

Questionnaire en ligne

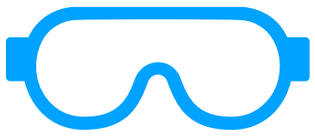
Q1' : Quels sont les critères de choix de bibliothèques ?

Q2 : Quels problèmes les chercheurs rencontrent-ils ?

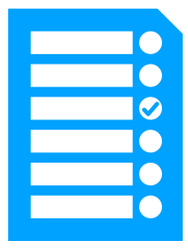
Q2' : Quelles sont leurs stratégies de résolutions ?

Q3 : Comment contribuer au mieux à leur travail ?

32 participants (AFIHM et CHI)



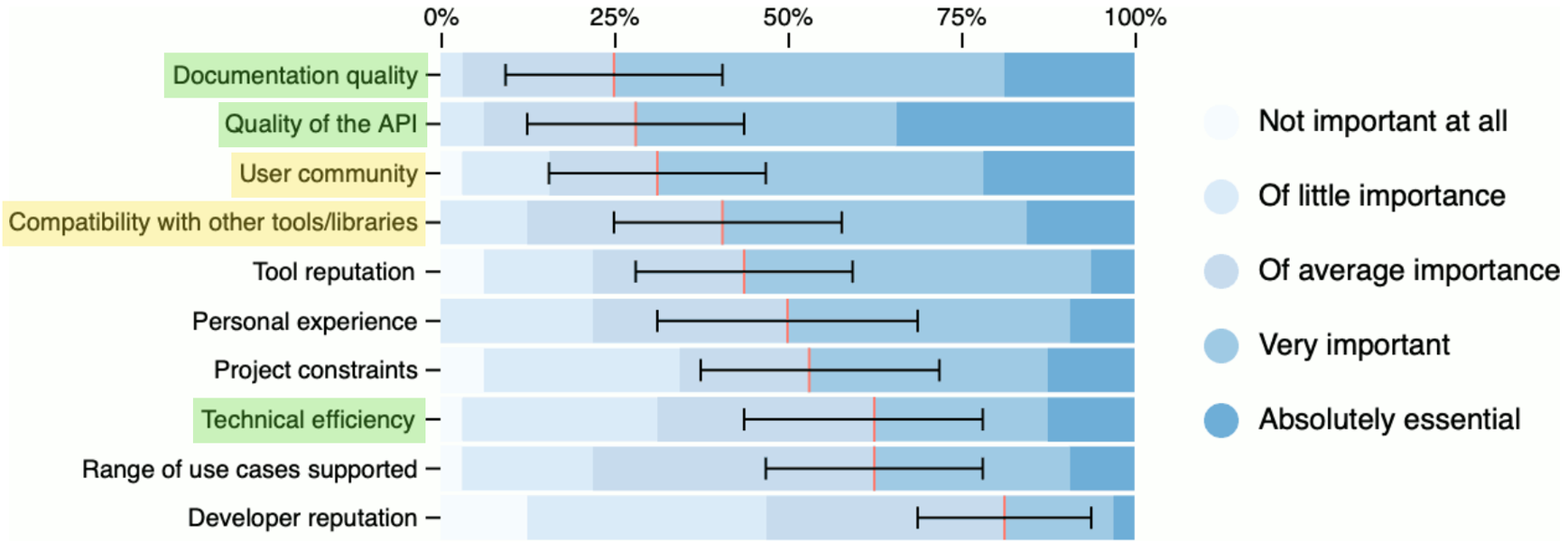
2/3 de niveaux avancés ou experts



4 parties

QCM 5~6 choix

Critères de choix des bibliothèques (Q1')

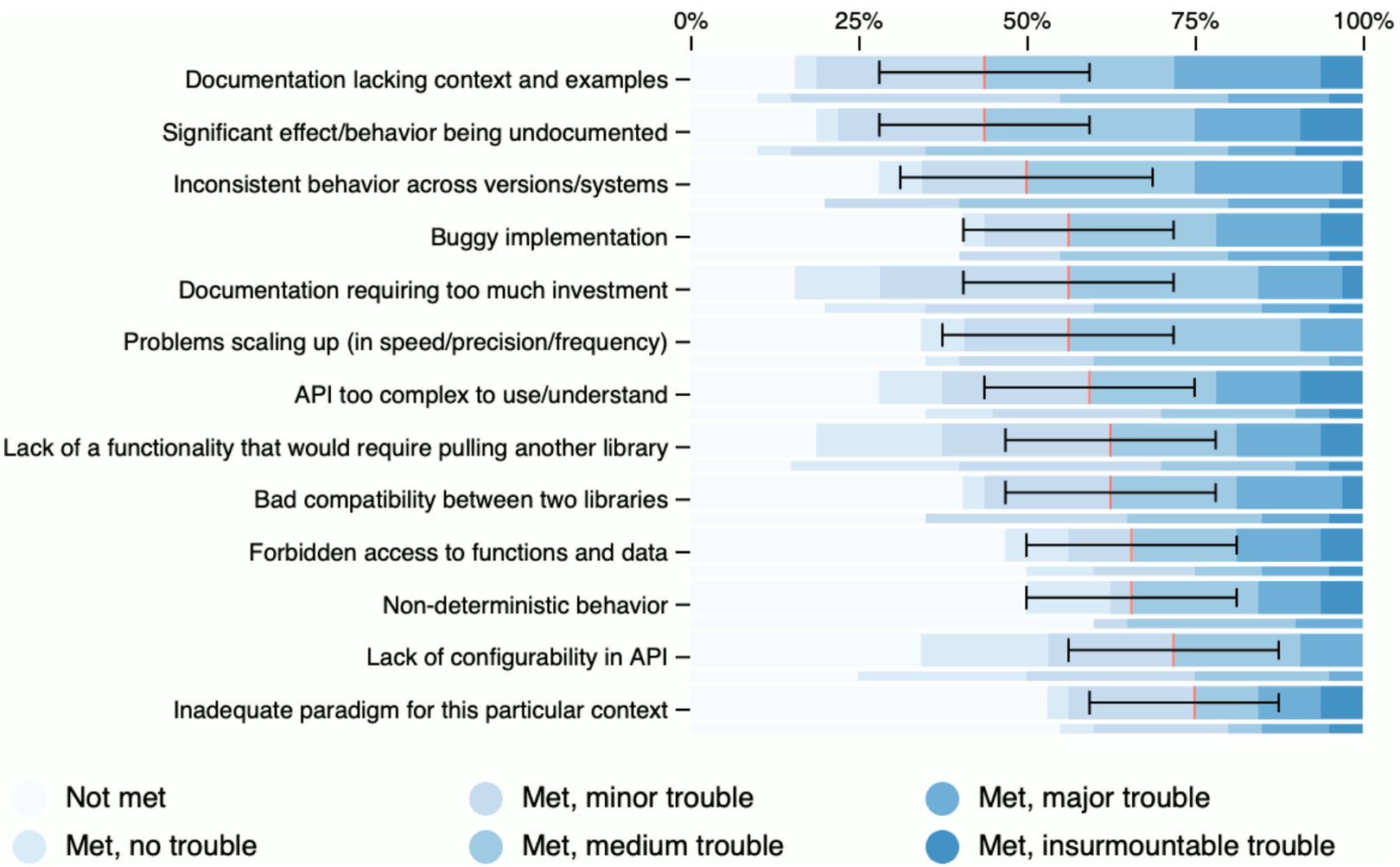


Prédominance de la *facilité d'apprentissage*

Présence de critères *mesurables*

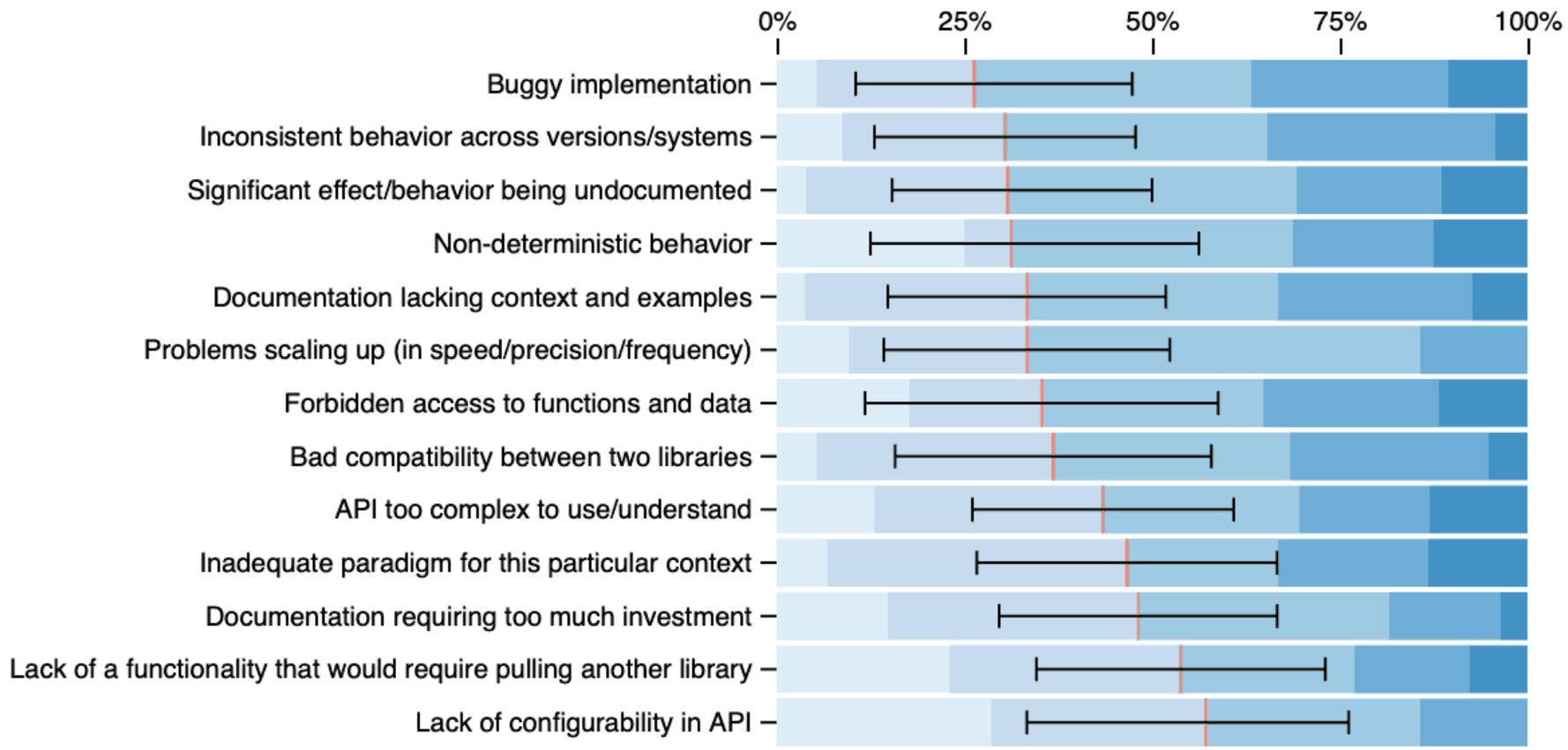
Outils cités : 7,6% issus de recherche, 37,5% Web

Problèmes rencontrés par les participants (Q2)



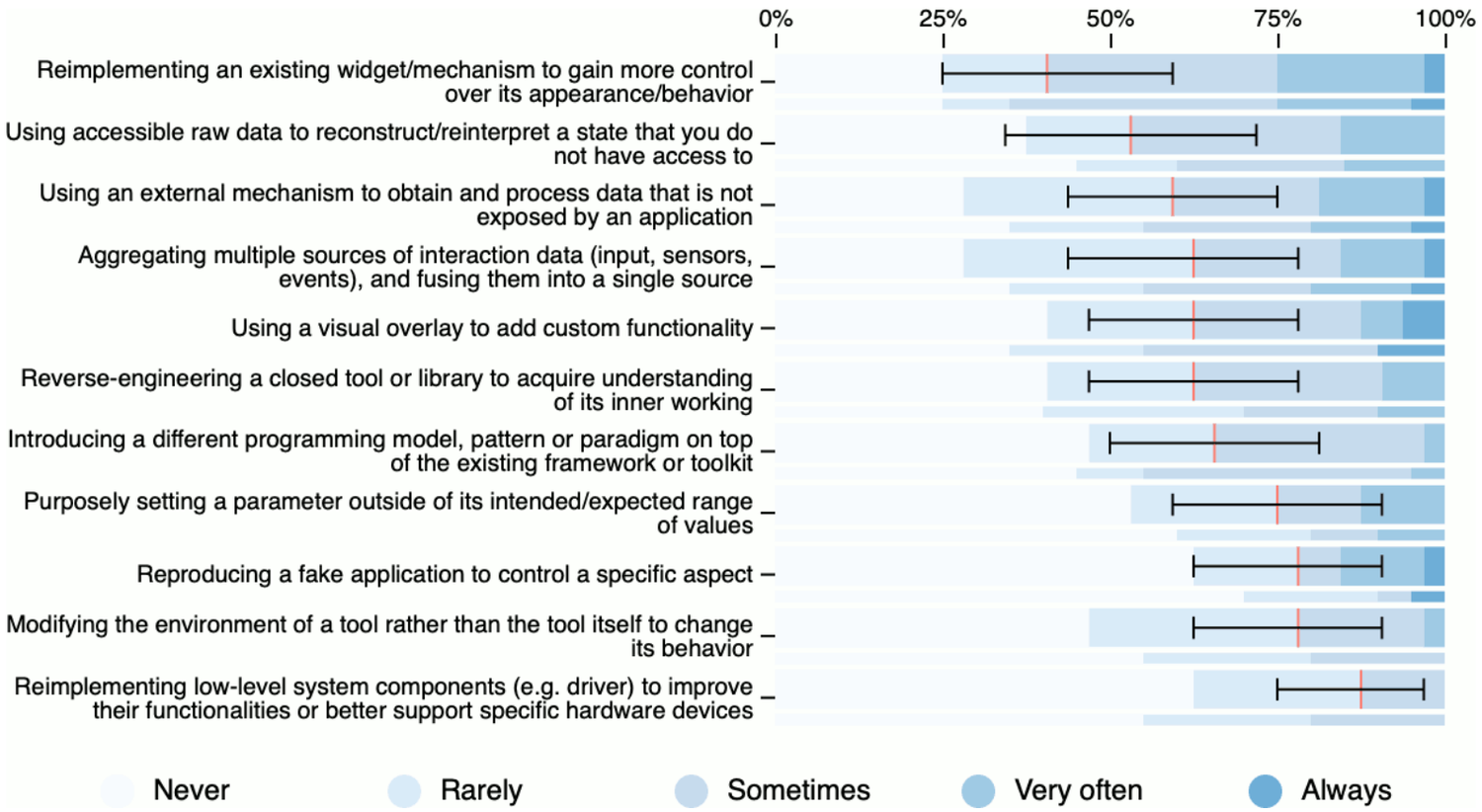
Catégorie d'utilisateurs minoritaires

Problèmes rencontrés par les participants (Q2)



Bibliothèques insuffisamment testées

Stratégies de résolution (Q2')



Manque de contrôle avec les éléments réutilisables

Mauvais choix de conception des données exposées

Importance des mécanismes d'*extension*, *réutilisation* et *transparence*

Pistes de contributions (Q3)

Faciliter la **réplicabilité** des travaux par d'autres

Animation de fonctions, Polyphony

Soutenir l'utilisation de technologies **Web** pour le bureau

Polyphony

Permettre l'ajout de nouveaux contrôles par **réutilisation** de comportements

Polyphony

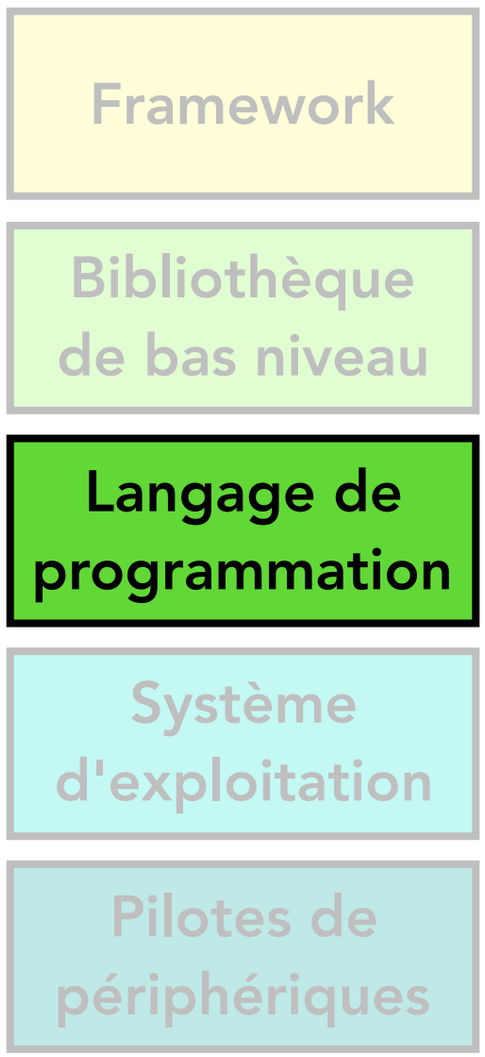
Complémenter les frameworks plutôt que les supplanter

Animation de fonctions

Plan

programmeur

1. Études préliminaires



3. Le framework Polyphony

support des animations

2. Animation de fonctions

Introduction

Quoi : expression d'animation par fonction+durée

```
object.setProperty(target) during 3s
```

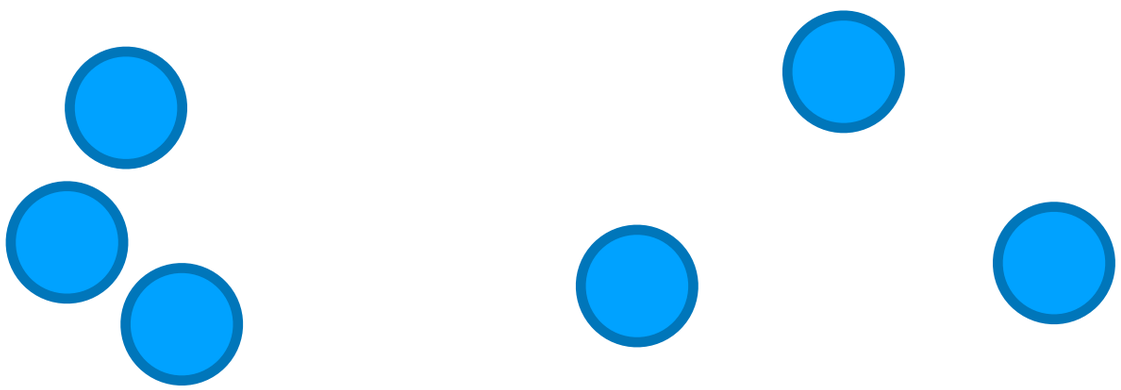
Pourquoi : syntaxe rapide à écrire et facile à mémoriser

Démarche : illustrer le support de l'interaction dans un langage, contribuer aux frameworks sans les modifier

Public visé : concepteurs de langages et de frameworks

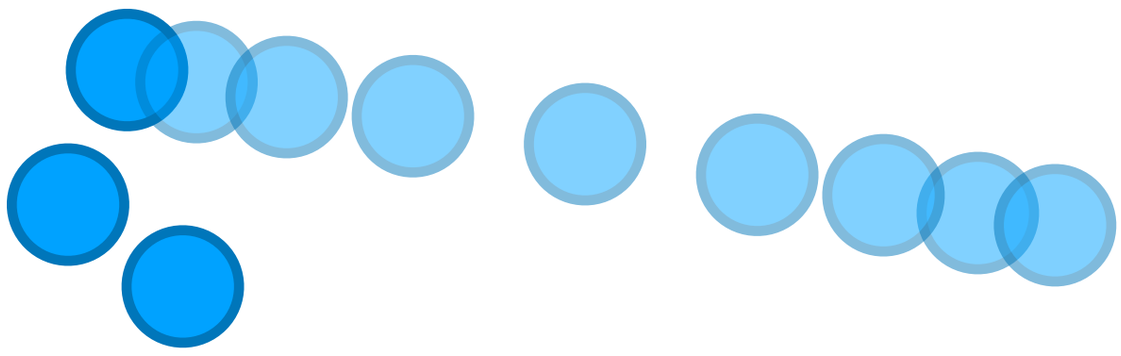
Les transitions animées

Transition instantanée



```
ball.setGeometry(QRect(100, -75, 30, 30));
```

Transition animée



```
QPropertyAnimation a(ball, "geometry");
a.setDuration(3000);
a.setEndValue(QRect(100, -75, 30, 30));
a.start();
```

Problèmes

Syntaxe d'animation lourde

Non cohérente avec la transition instantanée (mémorisation)

Exploration difficile avec/sans animations

Ensemble prédéfini de propriétés “animables”

État de l'art

```
object.transition()
   .duration(2000)
   .attr("property", target);
```

D3 (Javascript)

```
TweenLite.to(object, 2, {property: target});
```

GSAP (Javascript)

```
object.property = target
```

Core Animation (Swift)

État de l'art

```
object.transition()
   .duration(2000)
   .attr("property", target);
```

uniquement certaines fonctions

```
TweenLite.to(object, 2, {property: target});
```

mémorisation, tout sauf *certaines fonctions*

```
object.property = target
```

expressivité limitée

Proposition

```
object.setProperty(target) during 3s
```

Réification d'appel de fonction

```
object
"setProperty"
    target
```

Récupération de l'état initial

```
origin = getProperty()
```

Polymorphisme

```
origin*(1-t) + target*t
```

Planification d'appels

```
object.setProperty(...)
object.setProperty(...)
object.setProperty(...)
object.setProperty(...)
object.setProperty(...)
object.setProperty(...)
```

Implémentation

Pharo Smalltalk : orienté objet, dynamique, réflexif (et aidé par un développeur du noyau)

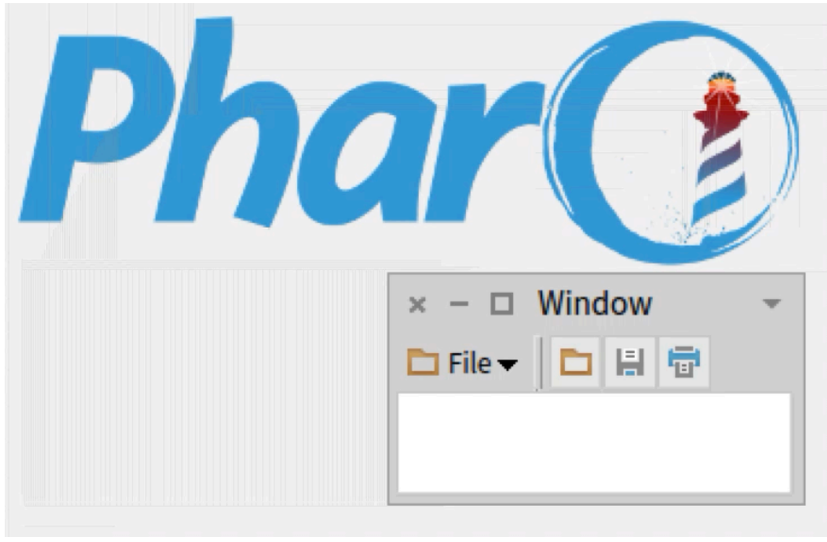
Syntaxe de base : `object property: value`

Prototype 1 : `object property: (value during: 3 seconds)`

Prototype 2 : `[object property: value] during: 3 seconds`

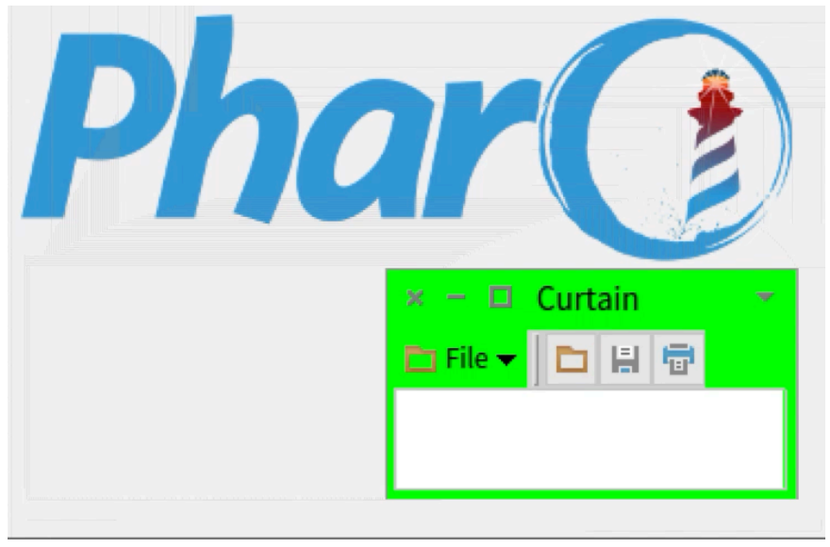
Résultats


```
[window position: 170@120] during: 3 seconds
```




```
[window color: Color green] during: 3 seconds
```

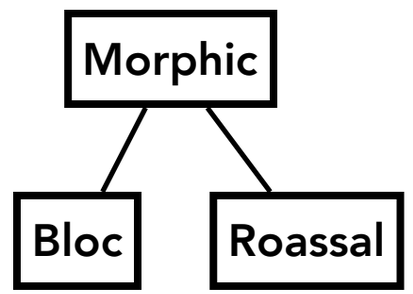

```
[window title: 'Curtain'] during: 3 seconds
```




```
a := [window position: 90@0] during: 3 seconds.
[a to: 10@120] during: 3 seconds
```

Limites

Planification des appels : dépendance d'un framework (dont les autres dépendent)



Violations de conventions : nommage (`translateTo:/position`), types d'entrée/sortie (`color:/color`)

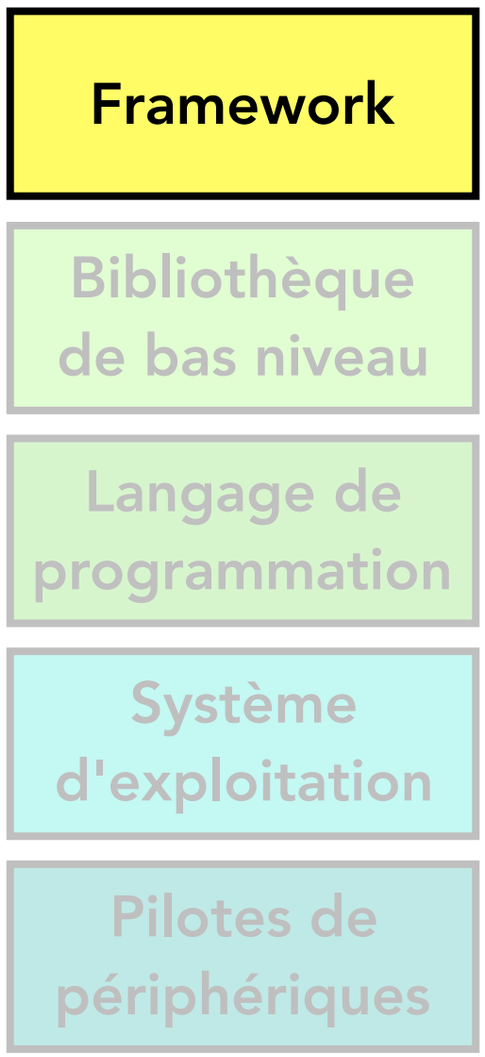
Types non interpolables : `*` et `+` non présents ou incompatibles

Bugs : propriétés non modifiables après affichage, pas de rafraîchissement automatique

Adoption : démonstration spectaculaire mais faible intérêt, mauvaise adéquation population/besoin

Plan

1. Études préliminaires



3. Le framework Polyphony

2. Animation de fonctions

Polyphony

Quoi : programmation d'interfaces graphiques avec le modèle Entité-Composant-Système (ECS)

Pourquoi : réutilisation par composition, accès unifié aux données de différentes couches

Démarche : contribuer à un modèle de programmation émergent, étudier les différences avec les jeux vidéo

Public visé : développeurs de frameworks ou de jeux vidéos

Historique

Thief: The Dark Project (1998)
séparation code/éléments

Operation Flashpoint: Dragon Rising (2007)
optimisation mémoire

Unity (2019)
parallélisme

Dungeon Siege (2002)
composition

Overwatch (2016)

Besoins identifiés

https://forum.unity.com/threads/gui-in-pure-ecs-projects.530578/

I would guess that most of us are just doing very bare bones "detecting clicks and touches on sprites" at the moment, but has anyone come across a true UI framework for ECS yet? Or have any ideas about where you intend to go with this? Are you writing your own input fields that handle mobile keyboards and the whole nine yards? Writing everything from square 1 seems a daunting and wasteful task, but is it unavoidable?

https://love2d.org/forums/viewtopi...

ECS for GUI

Sat Oct 10, 2015 10:21 am

So guys I have been thinking about GUIs recently. I have seen tutorials for Unity's GUI system from 4.6+ and it reminds me of an ECS. In reality it is probably done with classes and not entities but considering lua is better fit for ECS rather than OOP maybe an ECS based GUI would be a good thing? One benefit of an ECS based GUI would be that the output of a hypothetical GUI editor would be pretty easy to read and make. But what are yall opinions on a ECS based GUI?

gamedev.net

Paragon123 620

Posted July 26, 2016

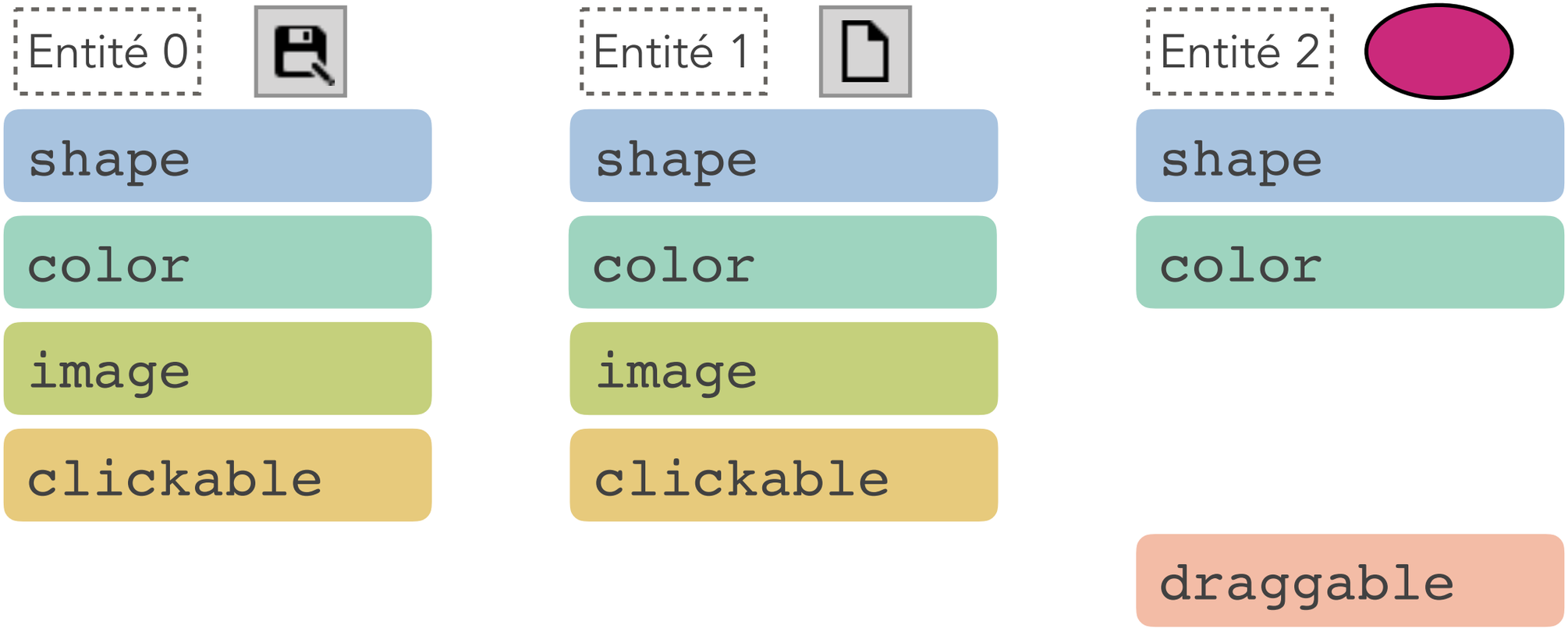
Does anyone have any tips, or articles about creating GUIs in an ECS system?

https://forum.unity.com/threads/gui-in-pure-ecs-projects.530578/

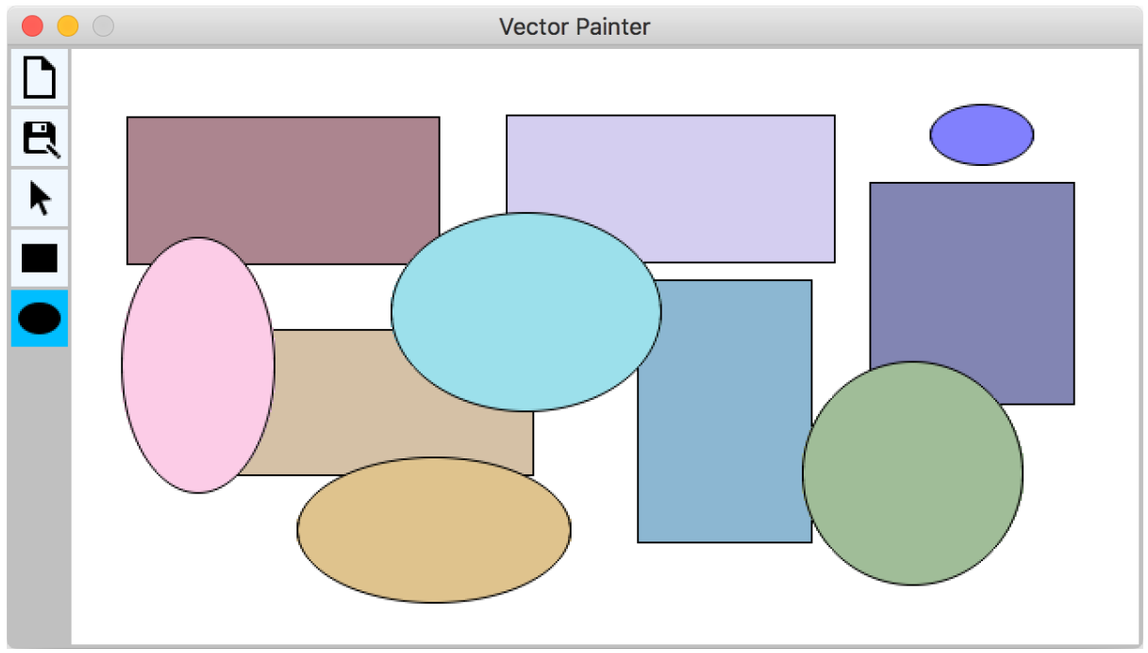
It is possible to make UI in ECS. Still I dont think anybody will do it anytime soon.

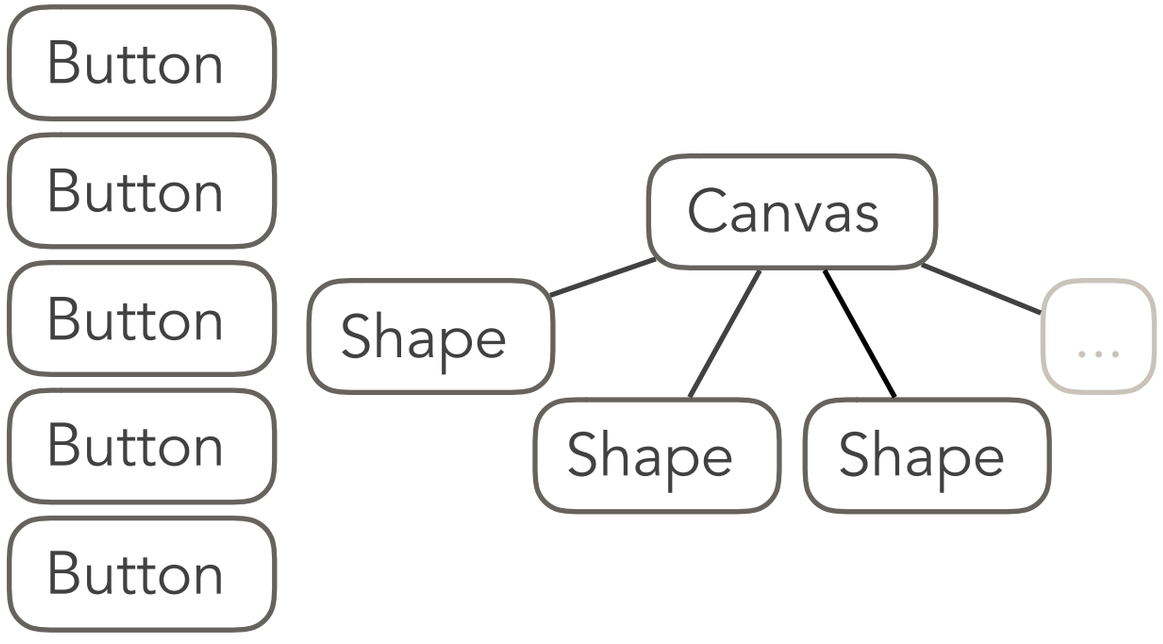
Apr 1, 2019

Entité-Composant-Système



Exemple d'interface





La chaîne de Systèmes

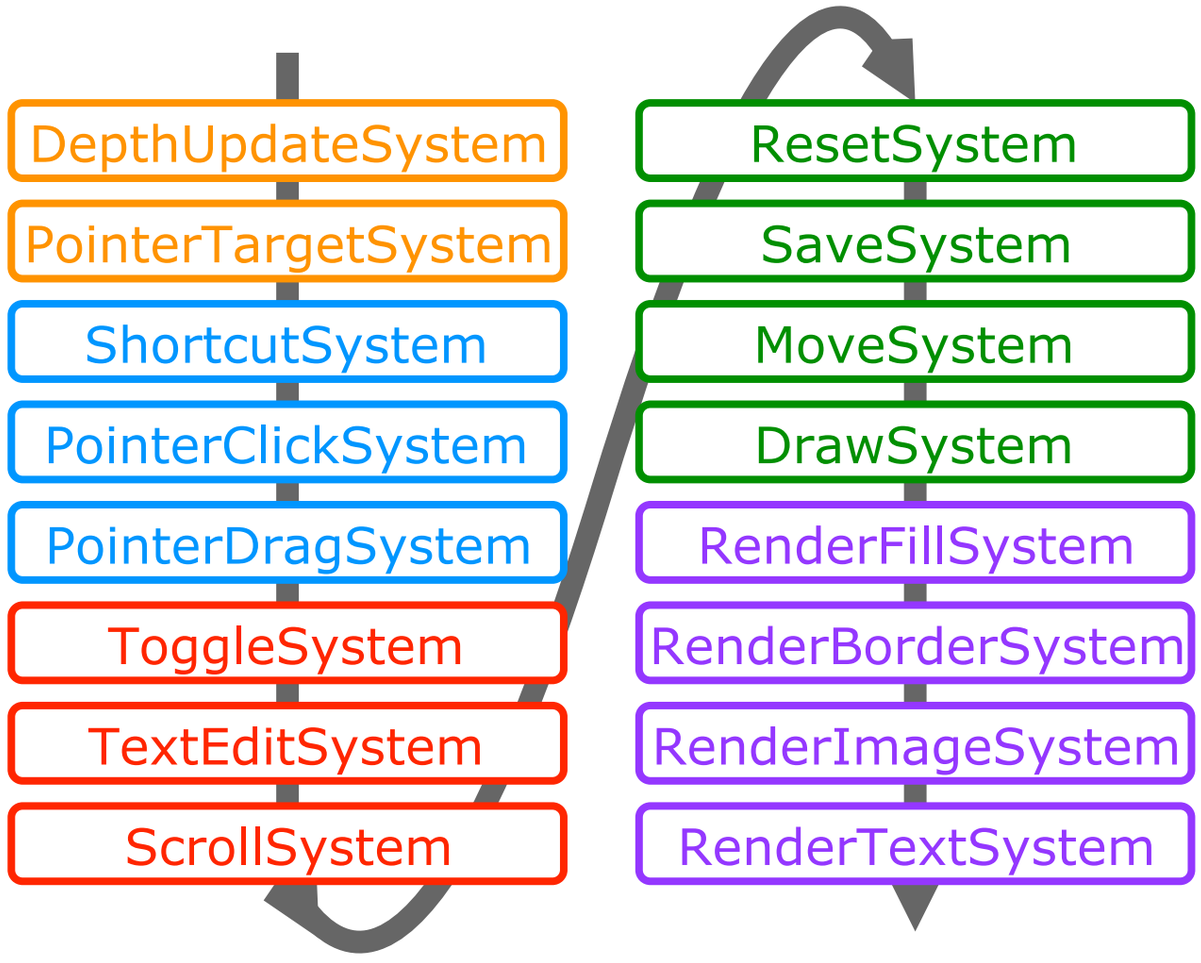
Gestion des entrées

Techniques d'interaction

Traitements des widgets

Traitements applicatifs

Rendu des sorties



Périphériques = Entités

Pourquoi : multiples périphériques, branchement à chaud, compatibilité sans abstraction (ex. touchpad → souris)

`cursorPosition`

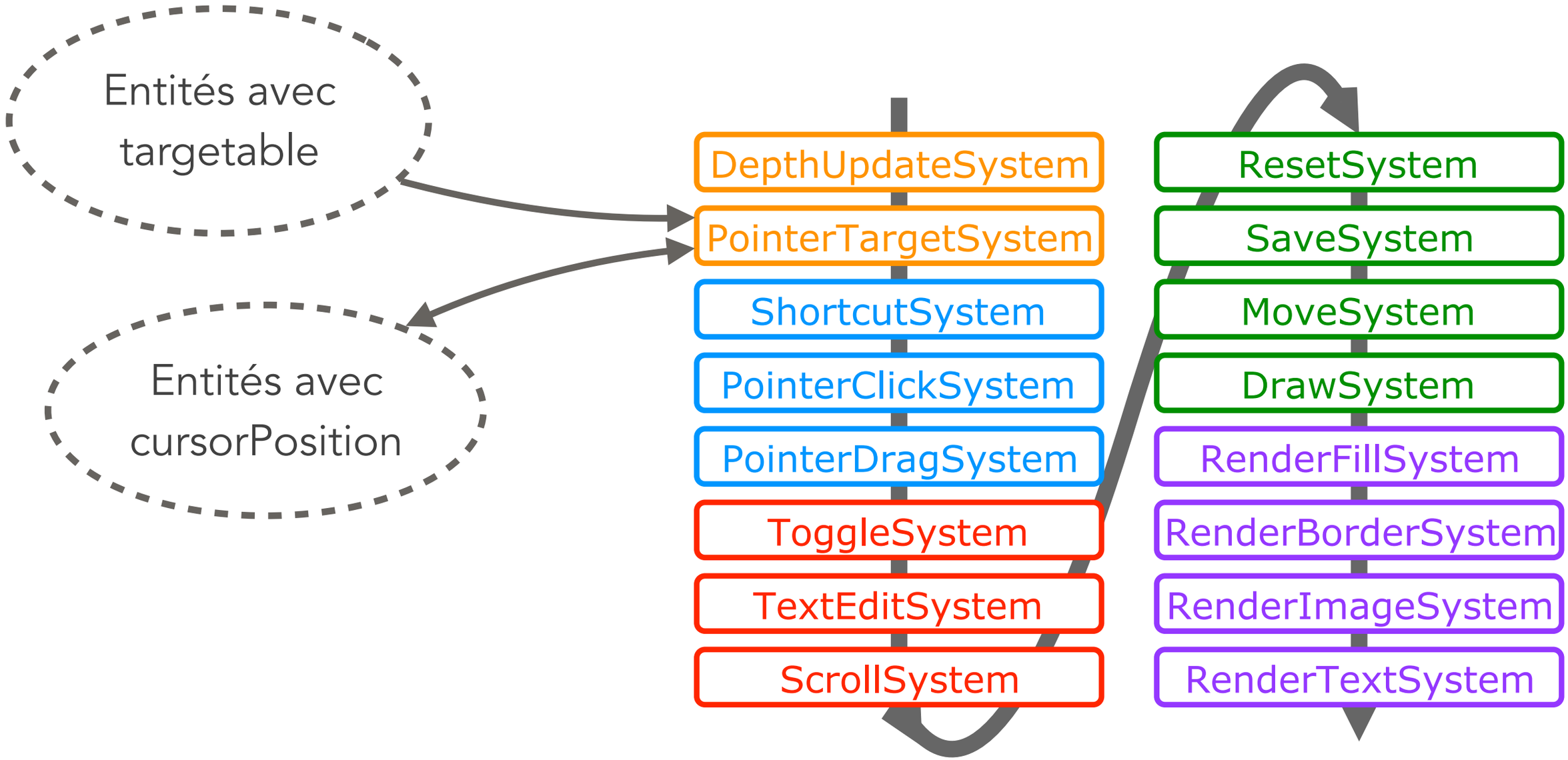
`buttons`

`keyStates`

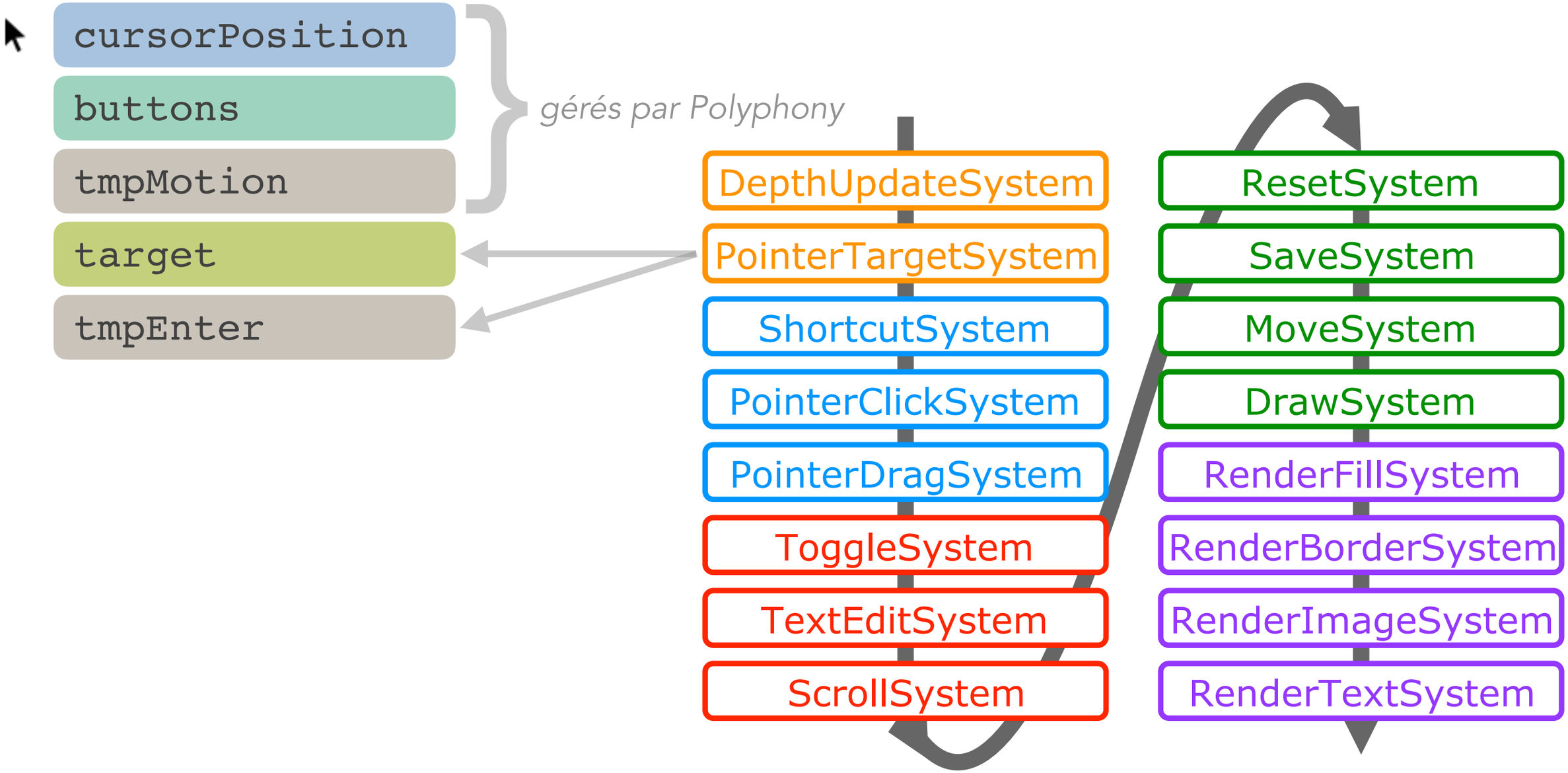
`bounds`

`origin`

Exemple de séquence



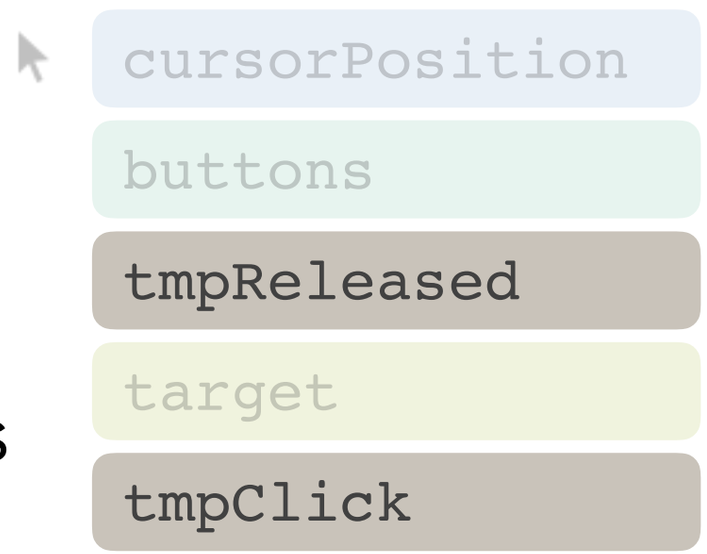
Exemple de séquence



Accumulation des données d'interaction

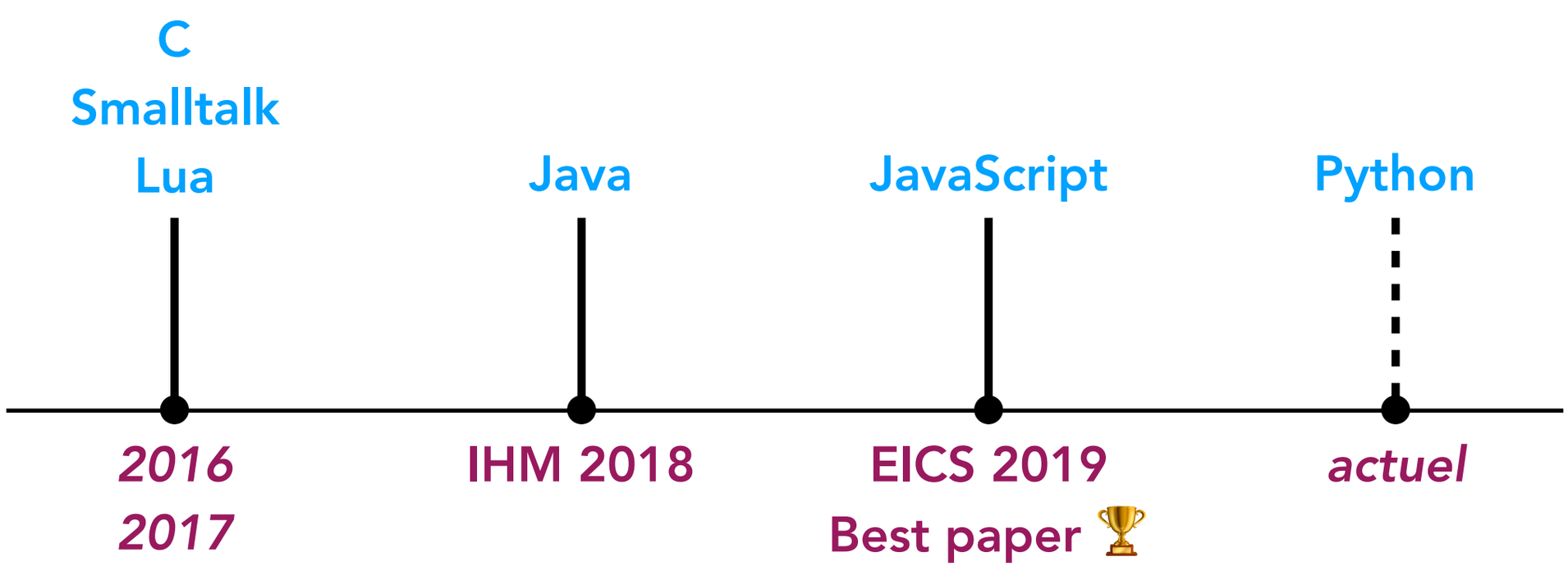
Composants temporaires : supprimés automatiquement à la fin des Systèmes

Avantages : historique des changements préservé, accès à toutes les données sans couches, plus de *listeners*



Limites : manque de lisibilité si beaucoup de Composants, pas de centralisation des évènements

Implémentations



Difficultés : structures de données (C), métaprogrammation (Smalltalk, Java), intégration de bibliothèques de bas niveau (Lua, JavaScript)

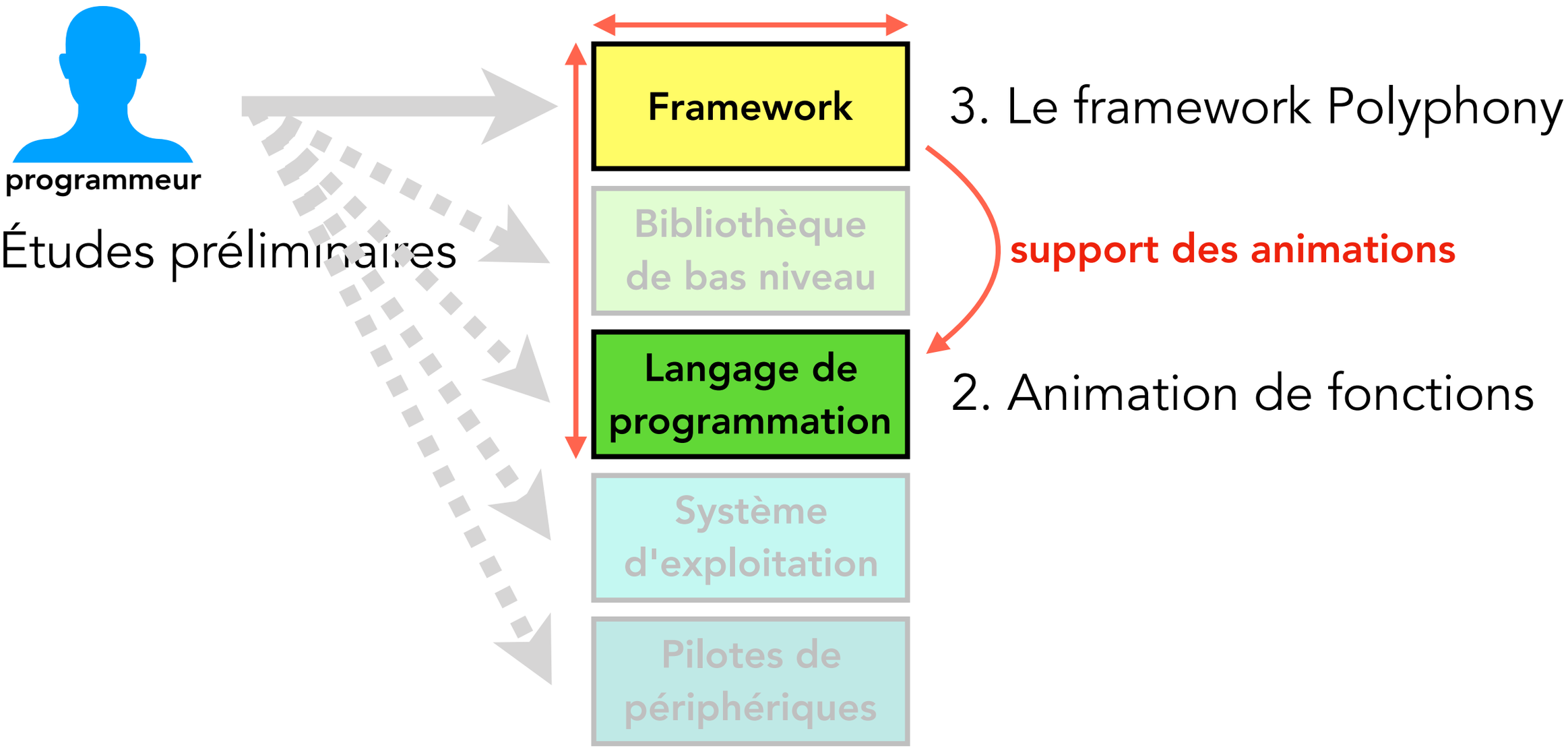
Suites de ce travail

Démonstration d'interactions : choisir un meilleur langage, raffiner le choix des Composants/Systèmes

Illustration dans un jeu vidéo : intérêt de la communauté, interfaces innovantes, fort attachement aux objets

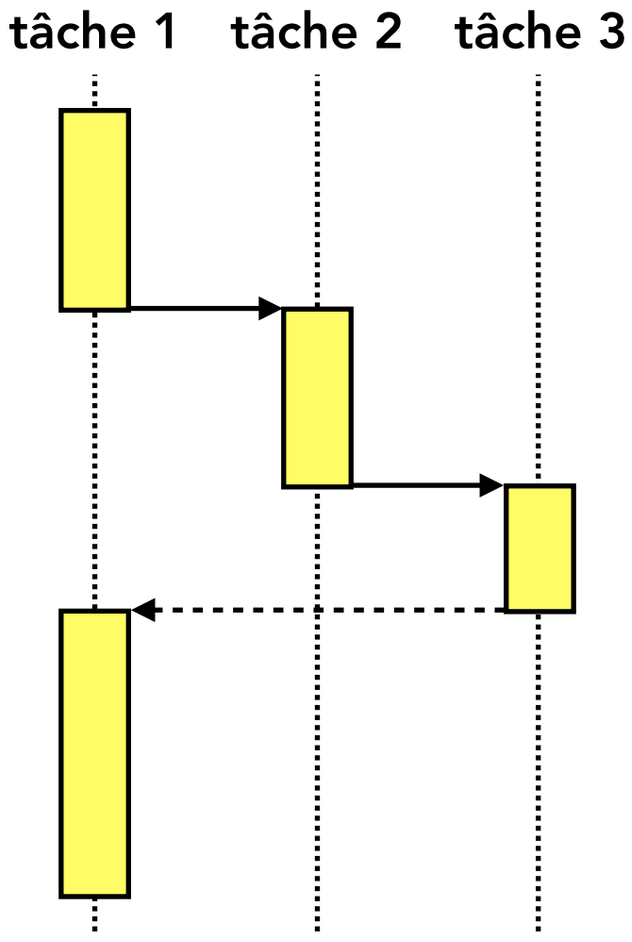
Langage ECS : implémentations basées/influencées sur les objets, syntaxes verbeuses, extensions précompilées

Vers des Essentiels d'Interaction

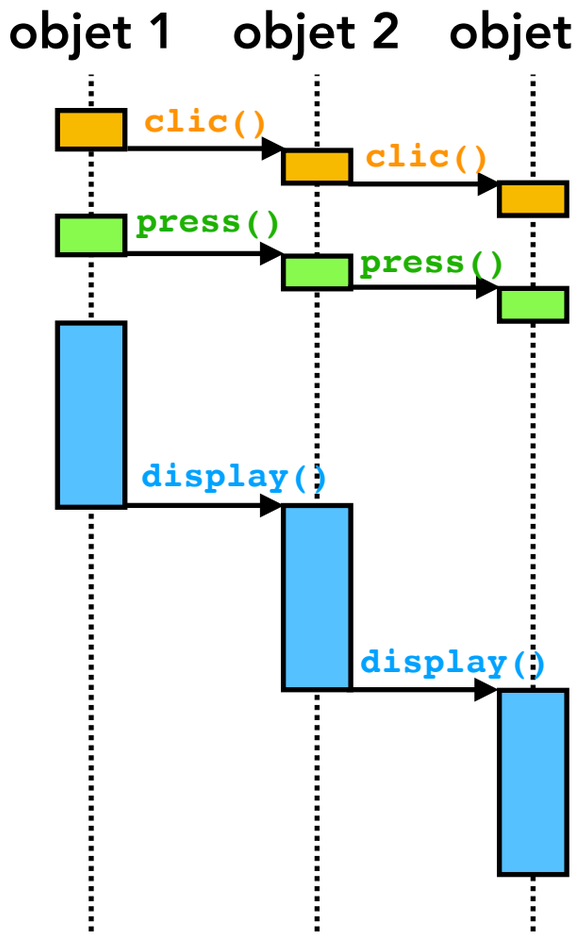


Orchestrer l'exécution

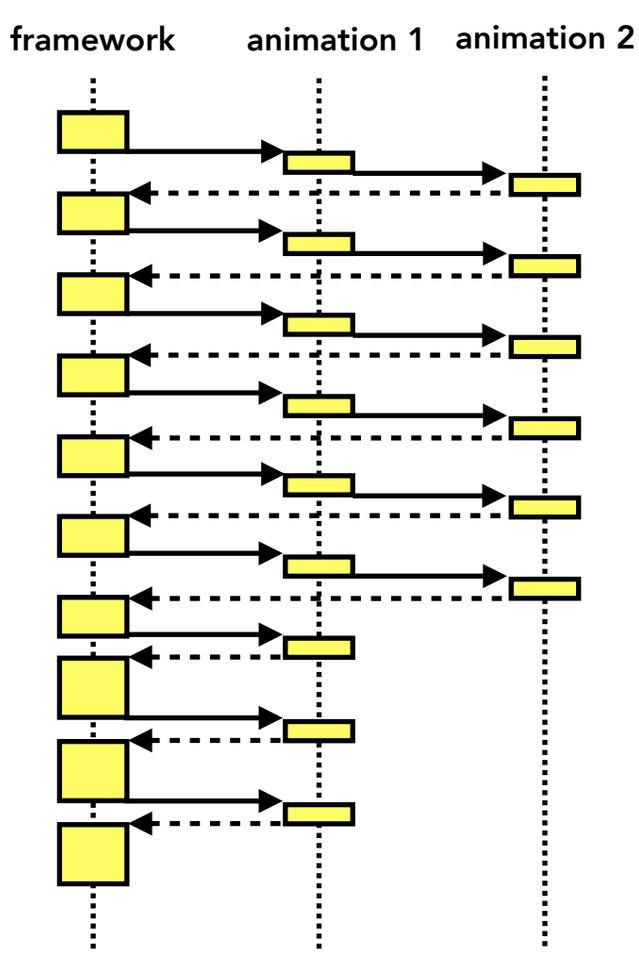
Calcul



Interface à objets



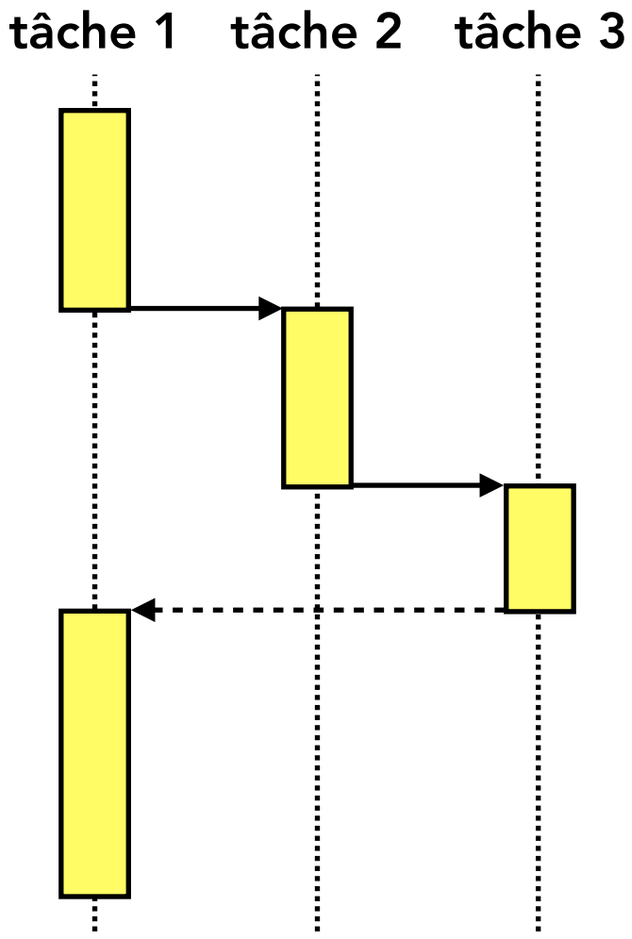
Animations



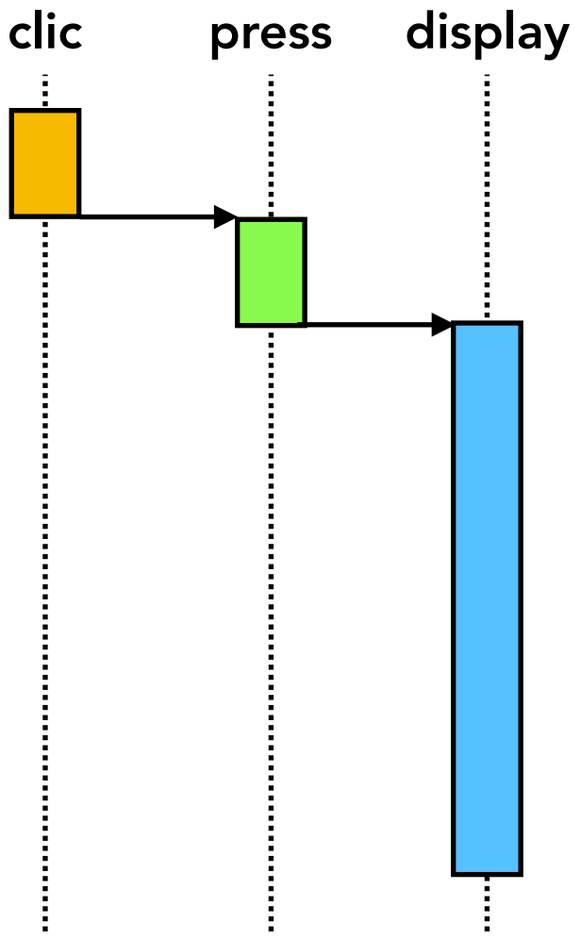
Difficultés : représentation visuelle, débogage, optimisation

Orchestrer l'exécution

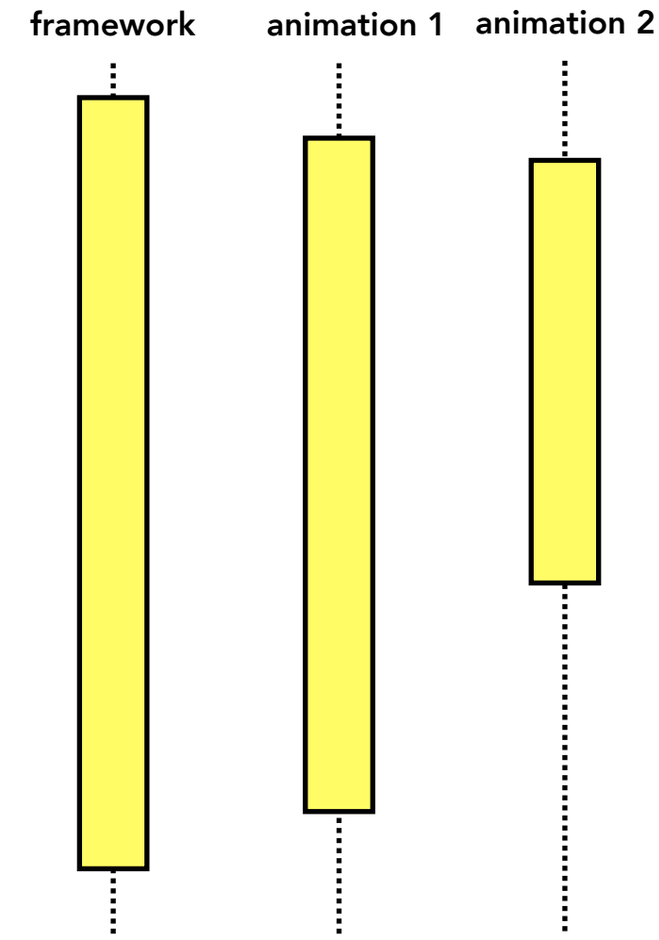
Calcul



Polyphony



Animation intégrée

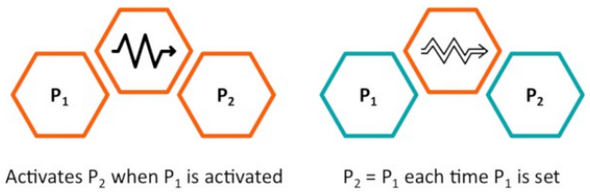


Démarche : réduction des "sauts" du flux d'exécution

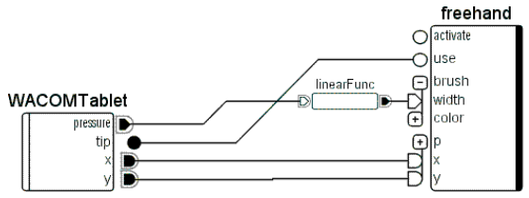
Orchestrer l'exécution

Orchestrer : arranger, ordonner et synchroniser des processus interactifs

1. déclencheurs/dépendances des blocs de code



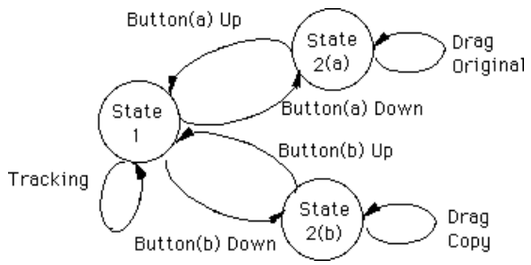
djnn/Smala (Chatty et. al)



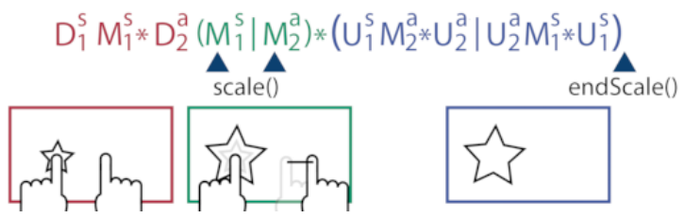
ICon (Dragicevic)

Hop.js (Serrano et. al)

2. code séquentiel pour interaction fragmentée



machine à états (Buxton)



Proton (Kin et. al)

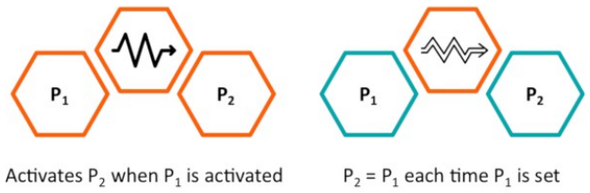
Seaside (Pharo)

Fournir un environnement d'interaction

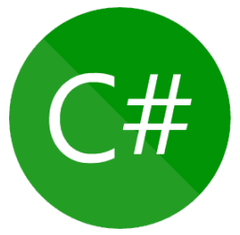
Environnement d'interaction : fonctions et structures de données pour programmer les périphériques d'interaction

Exemples : souris/clavier/écran initialisés, bibliothèques essentielles pré-installées, syntaxe adaptée

Processing (Fry et Reas)



djnn/Smala (Chatty et. al)

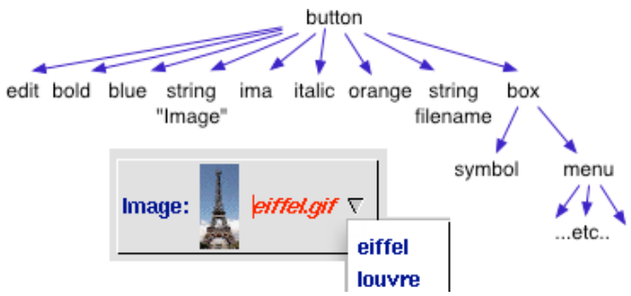


C# (Microsoft)

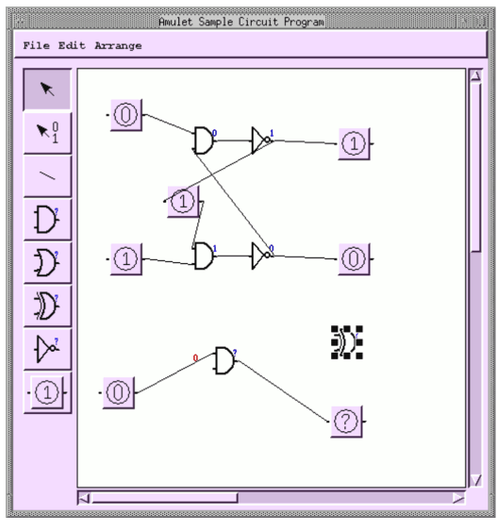
Étendre les langages

Métaprogrammation : manipulation d'éléments du langage ou programme comme des données

Intérêt : adapter la syntaxe du langage pour accompagner de nouveaux concepts, créer des concepts *générateurs*



UBit (Lecolinet)



Amulet (Myers et. al)

Contributions

1. Étude d'observation de chercheurs ayant prototypé de nouvelles techniques d'interaction

2. Description et démonstration du concept d'animation de fonction

3. Architecture d'interfaces graphiques sans callbacks, et stockage unifié des données d'interaction