

PROGRAMMER'S LEARNING MACHINE

MATTHIEU NICOLAS

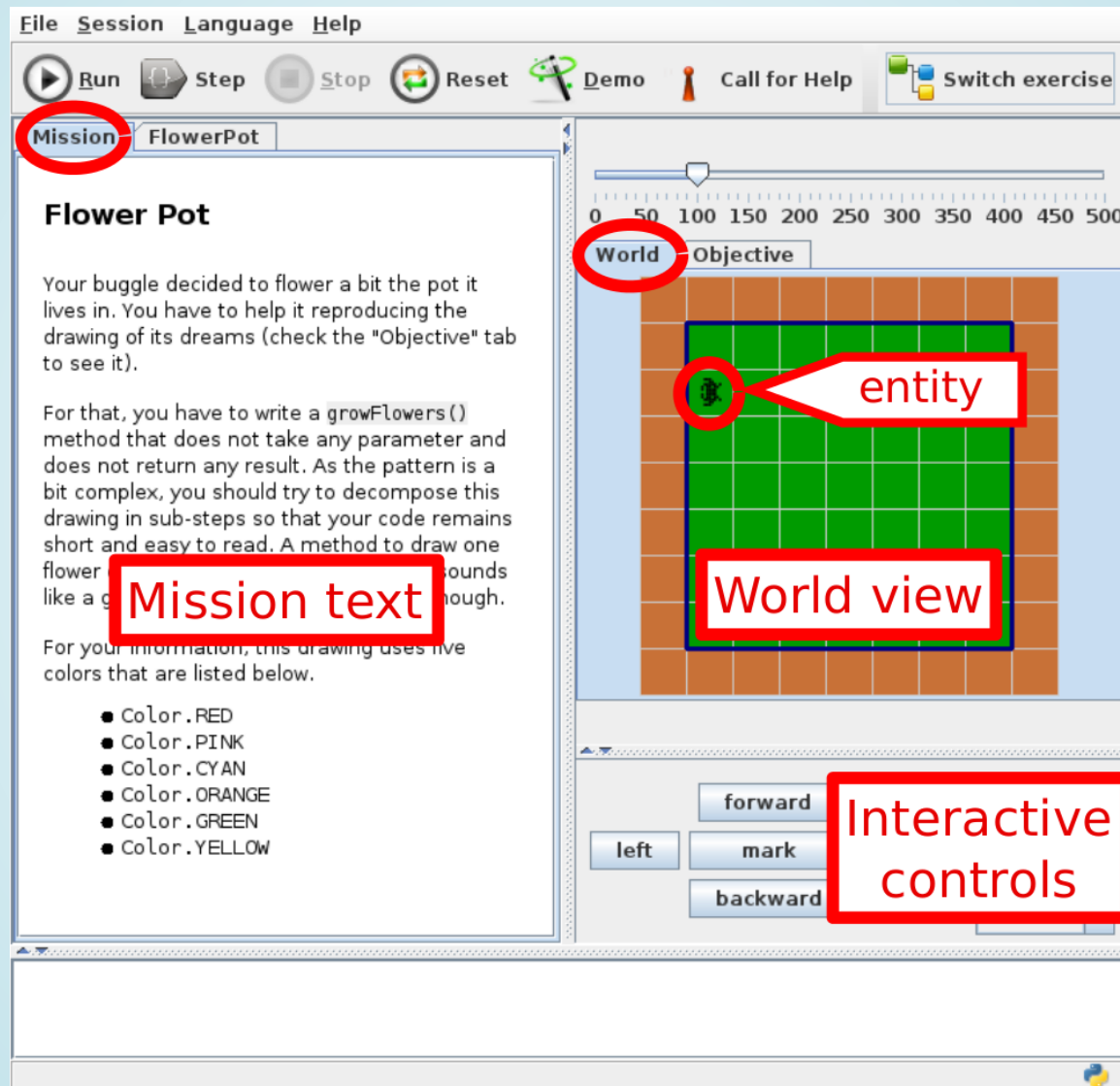
IJD SEMINAR - MARCH 03, 2015

WHAT IS PLM

- Software to learn programming
- Allows students to progress at their own speed...
- ... while the teacher helps the ones having trouble
- Used at *TELECOM Nancy* since 2008

STATE BEFORE
ADT

OVERVIEW



OVERVIEW

File Session Language Help

Run Step Stop Reset Demo Call for Help Switch exercise

Mission **FlowerPot**

Flower Pot

Your buggle decided to flower a bit the pot it lives in. You have to help it reproducing the drawing of its dreams (check the "Objective" tab to see it).

For that, you have to write a `growFlowers()` method that does not take any parameter and does not return any result. As the pattern is a bit complex, you should try to decompose this drawing in sub-steps so that your code remains short and easy to read. A method to draw one flower of the color passed in parameter sounds like a good start, but it is probably not enough.

For your information, this drawing uses five colors that are listed below.

- `Color.RED`
- `Color.PINK`
- `Color.CYAN`
- `Color.ORANGE`
- `Color.GREEN`
- `Color.YELLOW`

World **Objective**

0 50 100 150 200 250 300 350 400 450 500

forward
left mark right
backward

Brush Color
Buggle Color

OVERVIEW

The screenshot displays the Brave programming environment interface. At the top, a menu bar includes 'File', 'Session', 'Language', and 'Help'. Below it is a toolbar with icons for 'Run', 'Next', 'Stop', 'Reset', 'Demo', 'Call for Help', and 'Switch exercise'. The 'Next' button is highlighted with a red circle. The main window is divided into three sections. The left section, titled 'Mission: FlowerPot', contains a code editor with the following Python code:

```
1 def makeFlower(c):
2     setBrushColor(c)
3     brushDown()
4     forward(2)
5     backward()
6     left()
7     forward()
8     backward(2)
9     forward()
10    setBrushColor(Color.YELLOW)
11    brushDown()
12    right()
13
14 def line(c):
15     makeFlower(c1)
16     forward(3)
17     makeFlower(c2)
18     backward(5)
19 def halfLine(c):
20     forward(2)
21     makeFlower(c)
22     backward(3)
23
24 def growFlowers():
25     line(Color.RED, Color.CYAN)
26
27     right()
28     forward(2)
29     left()
30     halfLine(Color.ORANGE)
31     right()
32     forward(2)
33     left()
```

The 'Code area' is highlighted with a red box. The right section shows a 'World' view with a grid of colored squares (green, red, yellow, cyan) and a small bug icon. The 'ongoing execution' is highlighted with a red box. The bottom section is a 'console logs' area, highlighted with a red box, which currently shows 'Running Brave new world'. The status bar at the bottom indicates 'Running Brave new world'.

200 EXERCISES

FichierExerciceSessionLangageAide

▶ Exécuter

◀ Un pas

■ Stop

↺ Réinitialiser

🔄 Démo

🔗 Appeler à l'aide

📁 Changer d'exercice

MissionSourceCode

Bem-vindo ao mundo dos Buggles

Você acaba de iniciar o Programmer's Learning Machine. Este é um sistema de gestão de aprendizagem para ensinar a arte de programação de computadores através de exercícios interativos. É formado por um conjunto de exercícios agrupados por lições, para que você possa aprender no seu tempo. Atualmente, o ambiente está configurado para ser programado na linguagem de programação Java, mas você pode mudar isto no menu de Idioma/linguagem se você quiser, ou clicando no ícone Java na direita da barra de status.

Nesta primeira lição, os buggles vão guiar seus primeiros passos na programação.

Os buggles? O que é isso??

Os buggles são pequenos animais que obedecem a qualquer ordem que você dá a eles. Em cada exercício, você tem que fornecer a eles as instruções corretas para que o mundo se torne o objetivo do exercício. Por exemplo, neste exercício você deve instruir seu buggle a avançar um passo. Você pode ver isto verificando a diferença entre a visão *Mundo* e a visão *Objetivo*. Dependendo das lições (e suas configurações no menu Idioma/linguagem), seu código deve ser escrito em C, Java, Python ou Scala.

Ambiente de trabalho

Antes de seguir adiante, se acostume com o ambiente de trabalho. Dê uma olhada nos vários elementos que compõe a janela principal, mova seu mouse sobre eles para ver as dicas, e experimente os elementos para ver o que eles fazem. A área branca abaixo é o console: é onde erros e mensagens aparecem.

Se seu código tem erros (e qualquer código sempre vai ter em algum momento), o computador vai exibir mensagens de erro no console. Você obviamente terá que consertar os erros para passar nos exercícios. As mensagens que são mostradas podem soar assustadoras à primeira vista, mas não entre em pânico. O compilador é meio limitado nas suas capacidades de comunicação, mas ele não faz de má-vontade. Se você olhar de perto, a solução para o problema está escrita no meio daquelas mensagens ilegíveis. Você vai notar que com um pouco de hábito a gente se acostuma.

O que eu devo fazer?

É chegada a hora de construir seu primeiro programa. Simplesmente peça a seu buggle para dar um passo para frente usando o painel de Código Fonte. Para isto, simplesmente escreva o seguinte código (clicar nos controles iterativos não é o suficiente: você tem que escrever o código e depois experimentar

050100150200250300350400450500

MondeObjectif

avance

gauche

marquer

recule

droite

Brush Color

pt_br

200 EXERCISES

FileExerciseSessionLanguageHelp

Run

Step

Stop

Reset

Demo

Call for Help

Switch exercise

MissionHanoiBoard

Tower of Hanoi

The Tower of Hanoi or Towers of Hanoi, also called the Tower of Brahma or Towers of Brahma, is a mathematical game or puzzle. It consists of three pegs, and a number of disks of different sizes which can slide onto any peg. The puzzle starts with the disks in a neat stack in ascending order of size on one peg, the smallest at the top, thus forming a pyramid.

The objective of the puzzle is to move the entire stack to another peg, obeying the following rules:

- Only one disk may be moved at a time.
- Each move consists of taking the upper disk from one of the pegs and sliding it onto another peg, on top of the other disks that may already be present on that peg.
- No disk may be placed on top of a smaller disk.

Goal of this exercise

Write the core of the method: `void hanoi(int height, int src, int other, int dst)`

This method will recursively solve the presented problem. The first parameter named `height` is the height of the tower to move; The second parameter `src` is the index of the initial tower, the third parameter `other` is the index of the unused peg while the fourth parameter `dst` is the index of the expected final tower.

A key to solving this puzzle is to recognize that it can be solved by breaking the problem down into a collection of smaller problems and further breaking those problems down into even smaller problems until a solution is reached.

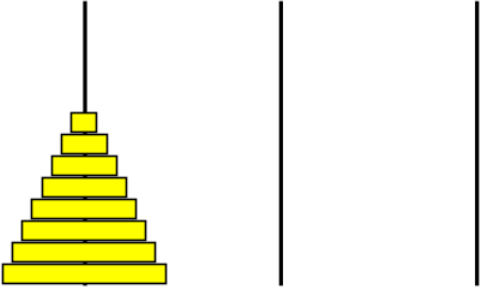
[I don't get it, please give me some extra indications](#)

solve(0,1,2)

050100150200250300350400450500

WorldObjective

0 moves



move

0

0

Validate

en

200 EXERCISES

Fichier Exercice Session Langage Aide

▶ Exécuter

◀ Un pas

■ Stop

↺ Réinitialiser

🔄 Démo

📞 Appeler à l'aide

📁 Changer d'exercice

Mission Lander

Se poser pour les nuls

Bravo ! Vous avez gagné dans une boîte de céréales un voyage (aller simple) pour Mars. La NASA vous a même fourni un module d'excursion lunaire pour l'occasion, et c'est donc avec ce véhicule que vous voyagerez.

Avant le décollage, vous devez vous entraîner sur simulateur aux bases de l'alunissage. Si tout se passe comme prévu, vous ne devriez pas avoir à voler en mode manuel, mais mieux vaut prévenir que guérir, comme ils disent.

Votre mission est de **poser le module avec une vitesse verticale inférieure à 10 m/s**. Il y a deux façons d'influer sur la trajectoire du module : en modifiant son angle, ou en réglant la poussée de son moteur. Dans cette simulation, nous ne nous intéressons qu'à la poussée, exprimée en m/s^2 .

Vous devez écrire le code qui sera appelé tous les dixièmes de seconde pendant la simulation. Assurez vous que votre code ne dure pas trop longtemps, et évitez en particulier les boucles `while`. Il faut juste ajuster la poussée du moteur en utilisant la fonction `void setPousseeDesiree(int poussee)` où `poussee` est un entier entre 0 et 4 (représentant une poussée entre 0 m/s^2 et 4 m/s^2). Vous ne pouvez incrémenter ou décrémenter la poussée que de 1 entre deux pas de la simulation. Ainsi, si la poussée du moteur est actuellement de 2 et que vous demandez 4, vous n'aurez que 3 à la prochaine étape. Si vous demandez une valeur inférieure à 0 (ou supérieure à 4), tout se passera comme si vous aviez demandé 0 (ou 4).

Afin de prendre des décisions informées, vous pouvez demander des informations sur l'état actuel du module. Dans cette simulation, vous ne serez probablement intéressé que par sa position verticale (accessible grâce à `double getY()`) et sa vitesse verticale (accessible grâce à `double getVitesseY()`). Vous pouvez également demander la poussée actuelle du moteur (avec `int getPoussee()`) si vous avez oublié ce que vous avez demandé. Rappelez vous que la gravité martienne est de 3.711 m/s^2 , cela peut vous être utile.

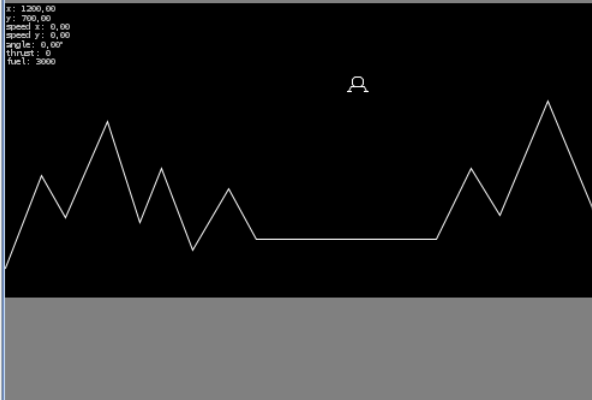
Une dernière chose : le module est plutôt petit, et vous n'avez pas tant de carburant que cela. À chaque pas de simulation, il consomme autant d'unité de carburant que la poussée actuelle du moteur. Si vous tombez en panne sèche, vous serez en chute libre alors attention ! Vous pouvez consulter le niveau du réservoir avec `int getFuel()`.

Bon courage !

0 50 100 150 200 250 300 350 400 450 500

Monde Objectif

x: 1200.00
y: 700.00
speed x: 0.00
speed y: 0.00
angle: 0.00°
thrust: 0
fuel: 3000



fr

PROGRAMMING LANGUAGES



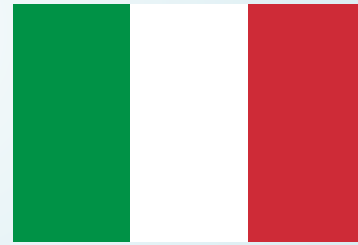
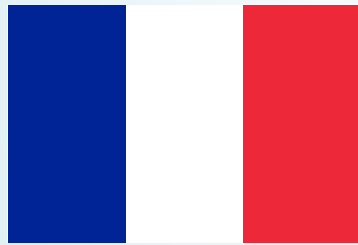
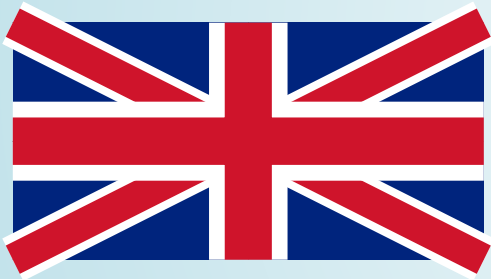
Java™



python™



LANGUAGES



USER TRACKING

- Keep the students' sessions
- Track the students' progress

USER TRACKING

- **git** used in order to version the student's code
- Local repository and anonymous branch
- Data pushed to a **GitHub** repository



USER TRACKING

- User's actions stored as commits

```
{  
  kind:"executed",  
  lang:"Java",  
  exo:"welcome.lessons.welcome.instructions.Instructions",  
  passedtests:"1",  
  totaltests:"1",  
  outcome:"pass"  
}
```

ADT'S GOALS

IMPROVE THE SOFTWARE'S
QUALITY

ATTRACT STUDENTS

- More teaching content
- Webification
- Gamification

AND TEACHERS

- Keep track of the students' progress
- Adapt content to their needs
- Able to add their own exercises

AND RESEARCHERS

- To an experimental teaching platform
- How to detect students having difficulties?
- What are the most common errors?

WORK DONE

FIRST STEPS

- Back-to-school season
 - Got the students' feedback...
 - ... and 170 bug reports
- Solved minor issues

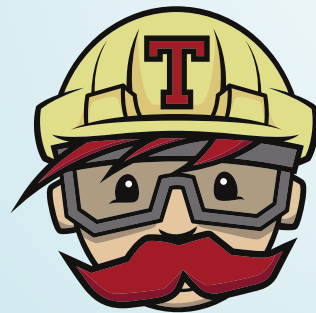
DEBUG USER TRACKING

- Lost data
- Fixed and refactored the code

UNIT TESTING

- Need to ensure the critical parts
 - **git**
 - exercises' solutions
 - lessons

CONTINUOUS INTEGRATION



Travis CI

CONTINUOUS INTEGRATION

- Execute tests automatically when commits are pushed
- Notify us as soon as a build failed

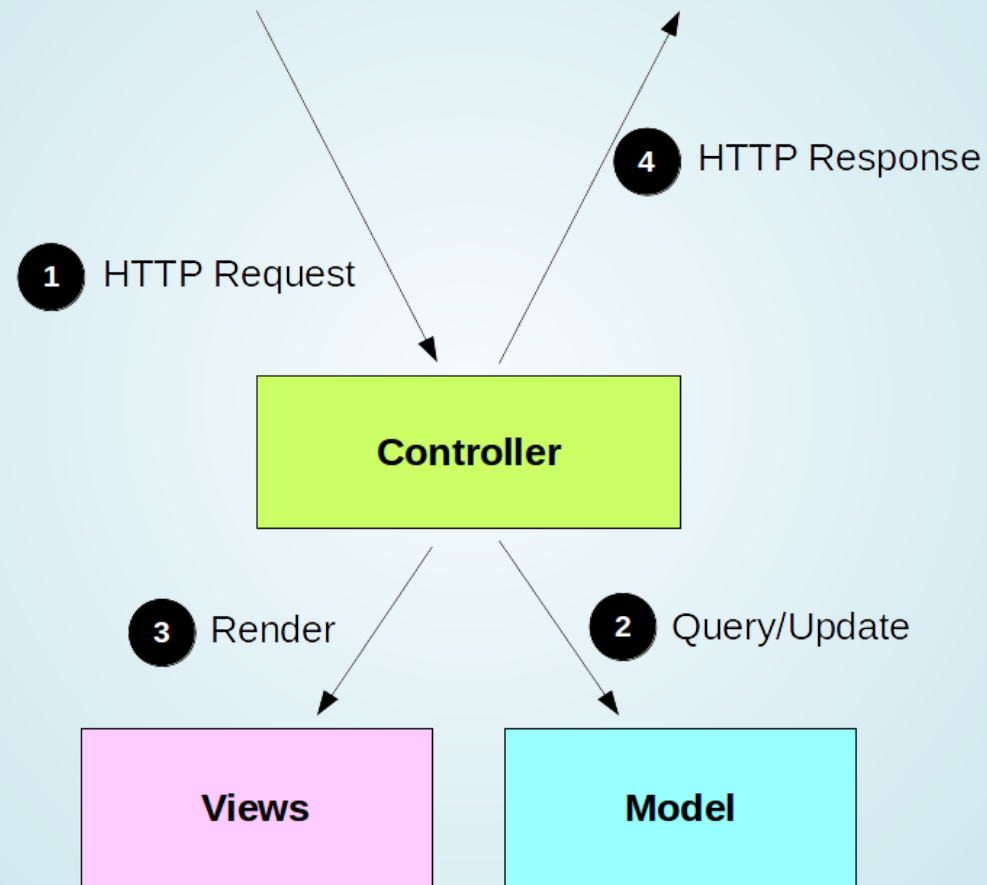
TO A WEB APP

PLAY FRAMEWORK

- Java and Scala web application framework
- Allow to set up application server

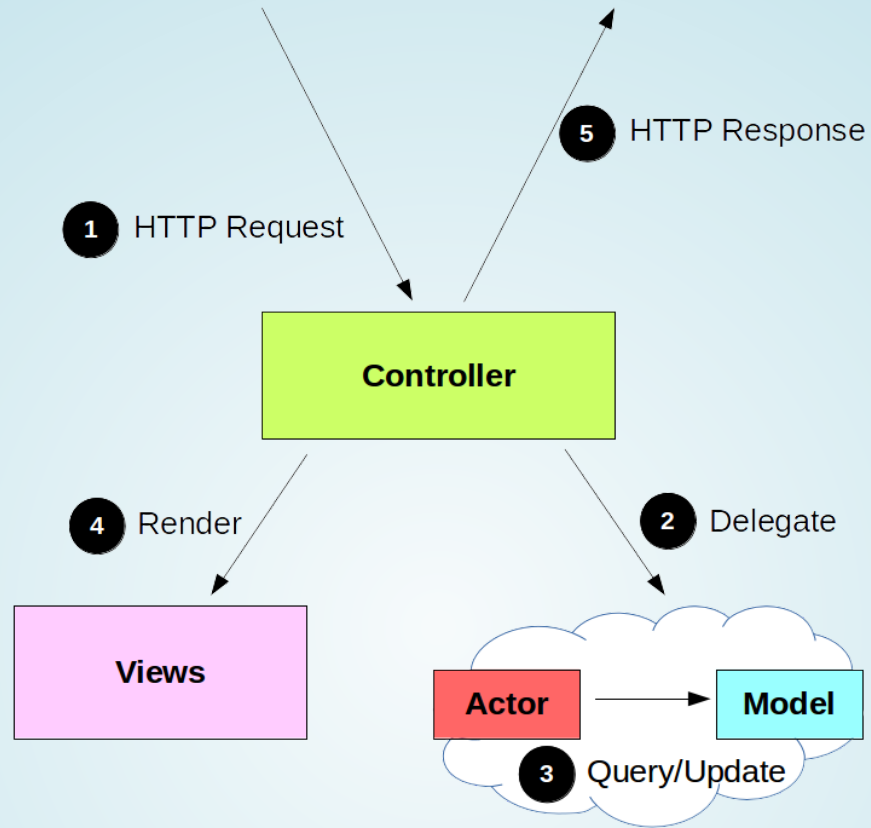


MVC PATTERN



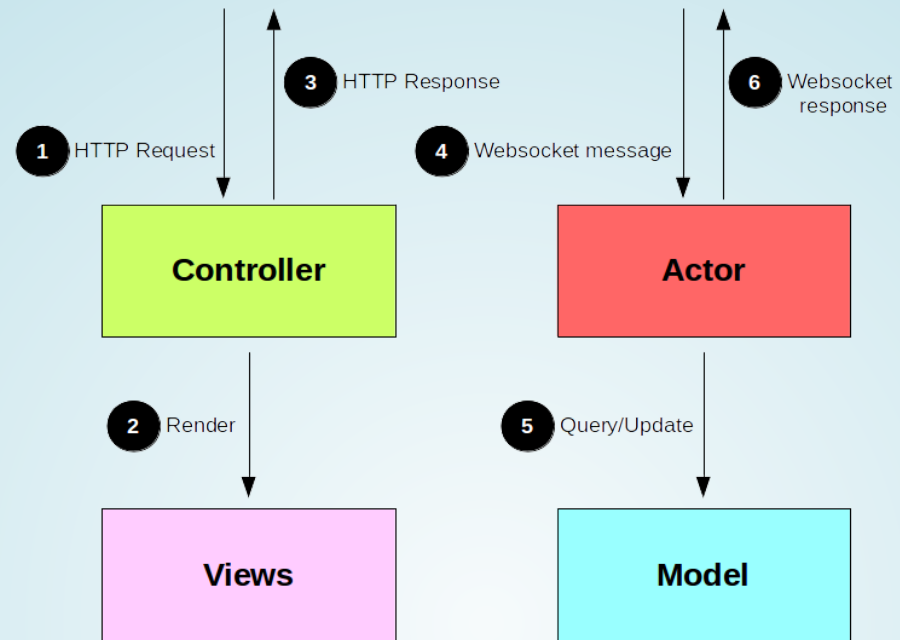
ACTORS

- Concurrent and scalable
- *'Let it crash'*



WORKING WITH WEBSOCKETS

- Controller only needed to render views
- One actor per websocket



REFACTOR PLM

- To be used as a library
- Remove the current UI
- Keep track of the world's evolution

REFACTOR PLM

- Has to update the server's and the client's world model
- Each time the world is modified...

```
setBugglePosition(newX, newY)
```

- ... the corresponding operation is generated

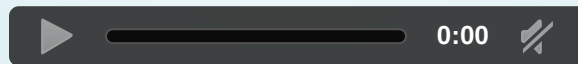
```
//Operation stored then send to the client  
moveBuggleOperation(oldX, oldY, newX, newY)
```

CLIENT-SIDE

- One-page application built with **AngularJS**
- UI made with **Foundation**



CLIENT-SIDE



RESULTS

- Main functionalities implemented
- First universe adapted
- Only as a local server

NEXT STEPS

- Convert other universes
- Embed debugging tools
- Support **C** language
- To a web server

THANKS FOR LISTENING

Do you have any questions?