ooc

un langage de programmation minimal, objet, flexible, portable

Amos Wenger

www.ooc-lang.org

« programmer »

l'art de se faire comprendre d'une machine

Amos Wenger

« langage »

moyen de communication

« défis »

complexe
performant
fiable
coût

« complexe »

champ visuel
neurones
calcul & mémoire

« fiable »

tests unitaires
déverminage

« coût »

temps
argent
muscles

« performant »

mille feuilles
échelle variable
optimiser

« solutions »

minimalisme
abstraction
logiciel libre

ooc

traduit en pur C99
lo | Scala | Python | Ruby
Mars 2009 | C objet
BSD | @GitHub

Amos Wenger

www.ooc-lang.org

ooc

classes, polymorphisme
surcharge d'opérateurs
ramasse-miettes (garbage collector)
génériques

Amos Wenger

www.ooc-lang.org

ooc

modules
collections
types primitifs
ubiquité

Amos Wenger

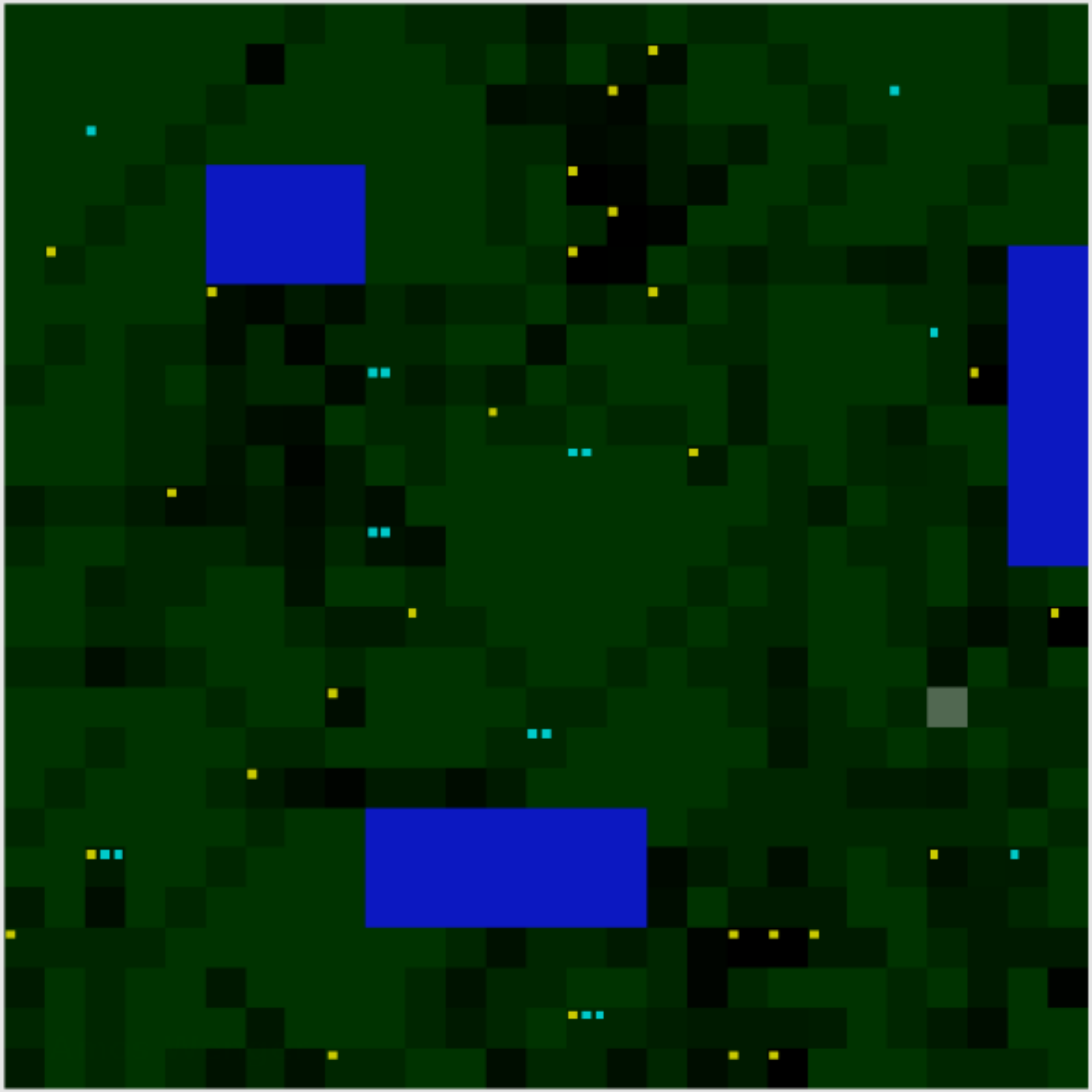
www.ooc-lang.org

ooc

GTK | SDL | OpenGL | SFML
C | C++ | Python | etc.
web | 3D | bureau | utilitaires

Amos Wenger

www.ooc-lang.org



Simulation control

Start

Sub-step

Ouvrir

Cell information

Cell at (23, 17)

Veg Level = 3,10

0 preys

0 predators

Land

World information

28 **preys**

18 **predators**

Preys' energy: 7

Predators' energy

Time: 0

View control

Zoom arrière

Quitter


Teeworlds
2:24
(1)[b1tchs

```
// appel de la fonction println(String)
println("Hello, world!")

// appel de la fonction membre String.println()
"Hello, world!" println()
```

```
ajouter: func(a, b: Int) -> Int {
    a + b
}
```

```
import structs/ArrayList

liste := ArrayList<Int> new()
liste add(1) .add(2) .add(3)

for (i in 0..liste size()) {
    liste[i] toString() println()
}
```

```
import io/FileReader

main: func {
  fr := FileReader new("/etc/hosts")

  while (fr hasNext())
    fr read() print()

}
```

```
for(i in 0..10) {
    // on peut utiliser printf aussi
    printf("%d\n", 0)
}
```

```
for(element in liste) {
    // affiche l'élément
    element println()
}
```

```
Animal: class {
    nom: String

    init: func (=nom)
}

anim := Animal new("Marsupilami")
("Mon animal s'appelle " + anim) println()
```

```
Animal: abstract class {
    crier: abstract func (message: String)
}

Chien: class extends Animal {
    crier: func (message: String) {
        printf("Woof woof, %s, woof woof !\n", message)
    }
}

Chien new() crier("E = MC^2")
```

```
Int: cover from int {
    negate: func -> This {
        -this
    }
}

a := 42
b := a negate()
printf("a = %d, b = %d\n", a, b)
```

```
convertir: func (entree: String) -> Int {
    return match entree {
        case "un"       => 1
        case "deux"     => 2
        case "trois"    => 3
        case =>
            Exception new("Je ne sais compter que jusqu'à tr
    }
}
```

```
commenter: func (note: Int) {
    println(match {
        case note < 10   => "Dommage.."
        case note < 15   => "Pas mal"
        case note < 17   => "Bien"
        case note <= 20  => "Excellent!"
        case             => "Tricheur!"
    })
}
```

```
use gtk
import gtk/[Gtk, Window]
exit: extern func

main: func {
    w := Window new("Hi, world")
    w setUSize(800, 600) .connect("destroy", exit) .showAll()
    Gtk main()
}
```

ooc

ce qui manque
ce qui est prévu
@ooc_lang | #ooc-lang

Amos Wenger

www.ooc-lang.org

questions ?