Corewar

Alexander Camatchy
Alix Danvy
Mateo Delerue
Abdoulaye Fofana

Université de Caen Normandie

26 Avril 2022

1. Corewar
2. Algorithme Génétique
   - Fonctionnement
   - Parrallélisme
   - Résultats
3. Interface Graphique
4. Conclusion

Introduction Corewar

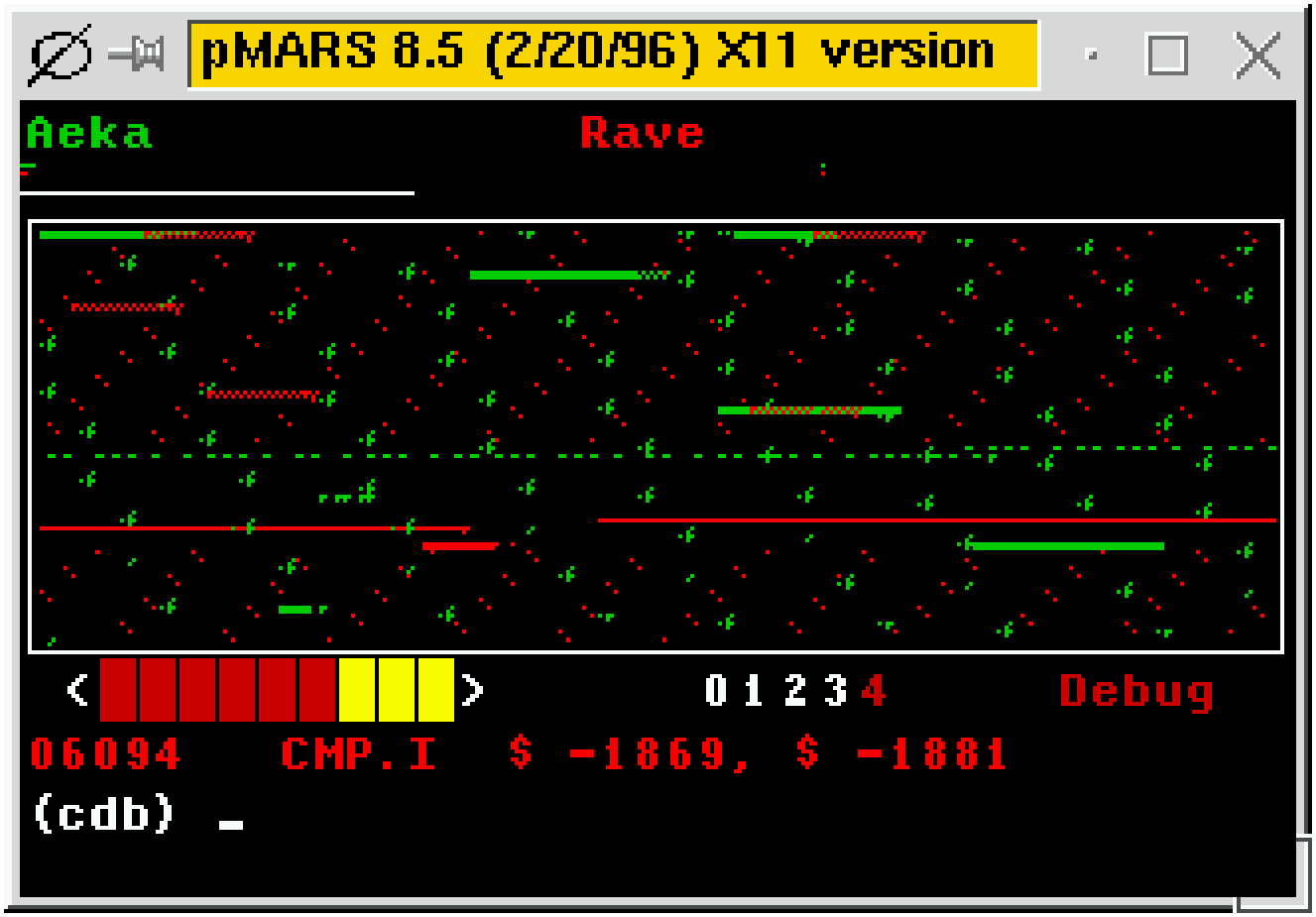

Figure: pMars Interpreter

RedCode

Instruction	Action
MOV A B	move A to B
ADD A B	add A to B
SUB A B	substract A from B
JMP A B	jump to A
JMZ A B	jump to A if B is zero
JMN A B	jump to A if B is not zero
CMP A B	if A equals B then skip the next instruction
SLT A B	if A is less than B then skip the next instruction
DJN A B	decrement B then if B is not zero jump to A
SPL A B	create a new process at A
DAT A B	the process dies

RedCode

Notation	Mode d'addressage	Action
A	Direct	Accède à la case A relativement à la case courante
#A	Octothorpe	Donne la valeur A
@A	Indirect	Accède à la case ayant l'addresse contenue dans la case A
<A	Pre-decremente	Décrémente la valeur contenue dans la case A puis accède à la case correspondant à cette addresse
>A	Pre-incremente	Incrémente la valeur cont enue dans la case A puis accède à la case correspondant à cette addresse

RedCode

```
ca      equ  9
cb      equ  -960

        mov  5, <-1
a       add  @2, 3
        sub  >ca+1, 10
        jmp  aa, #ca
aa      jmz  -9*-2, #+3
        jmn  -a, @2
        cmp  @-9, <8
aaa     slt  1, >0+2
        djn  325644521, a-1
        spl  @34*31, @cb/2
```

Figure: Exemple de code source RedCode en norme ICWS-88

Architecture

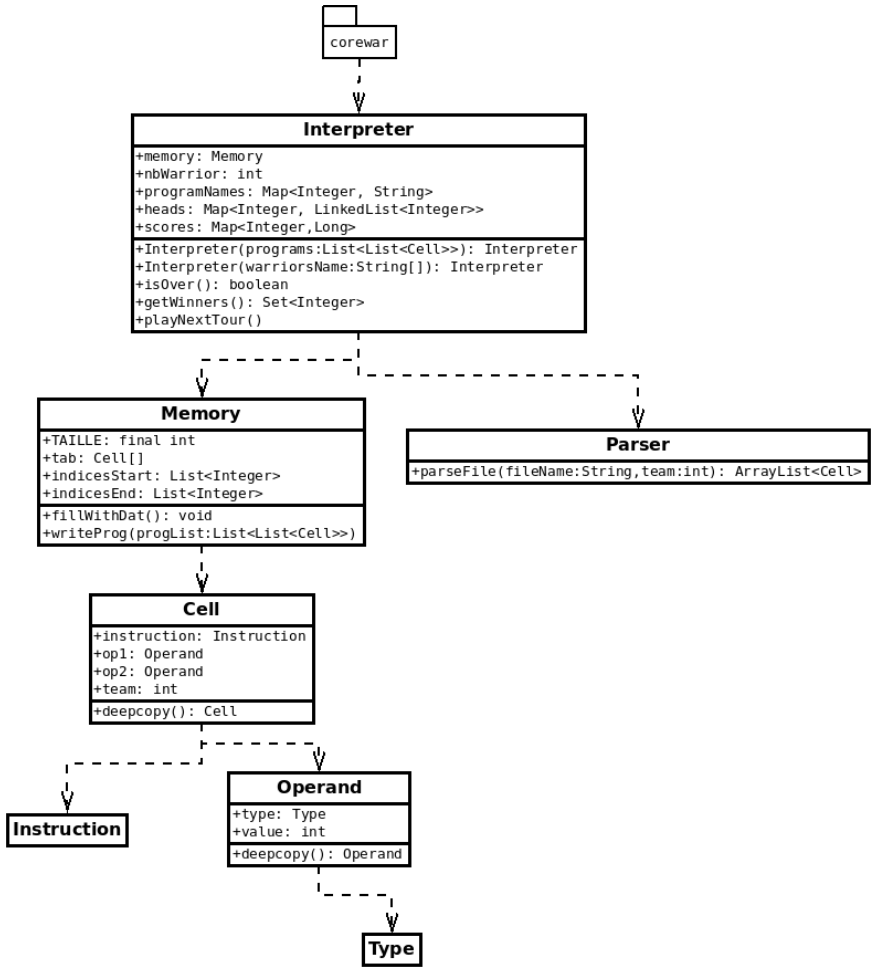

Figure: Diagramme de classe du projet

Fonctionnement

Algorithme 1 : Pseudo-algorithme du déroulement d'une partie

```
1  programs ← Parser.parseFiles(args)
2  memory.writePrograms(programs)
3  tant que !interpreter.isOver() faire
4      pour program ∈ interpreter.programs faire
5          head ← program.popProcess()
6          nextHead ← interpreter.execute(memory.fetch(head))
7          si nextHead != null alors // Tête toujours en vie
8              program.pushProcess(nextHead)
9          fin
10     fin
11 fin
```

Algorithme Génétique

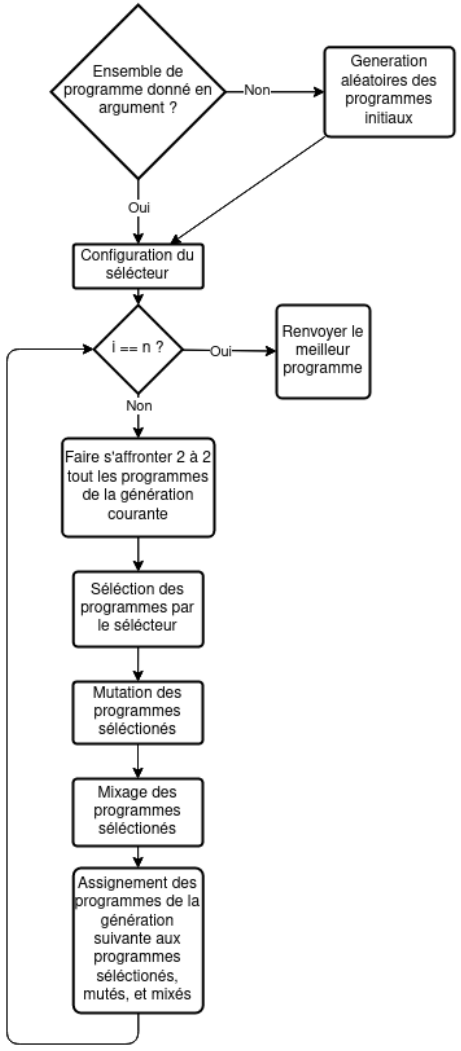

Figure: Schéma du fonctionnement de l'algorithme génétique

Architecture

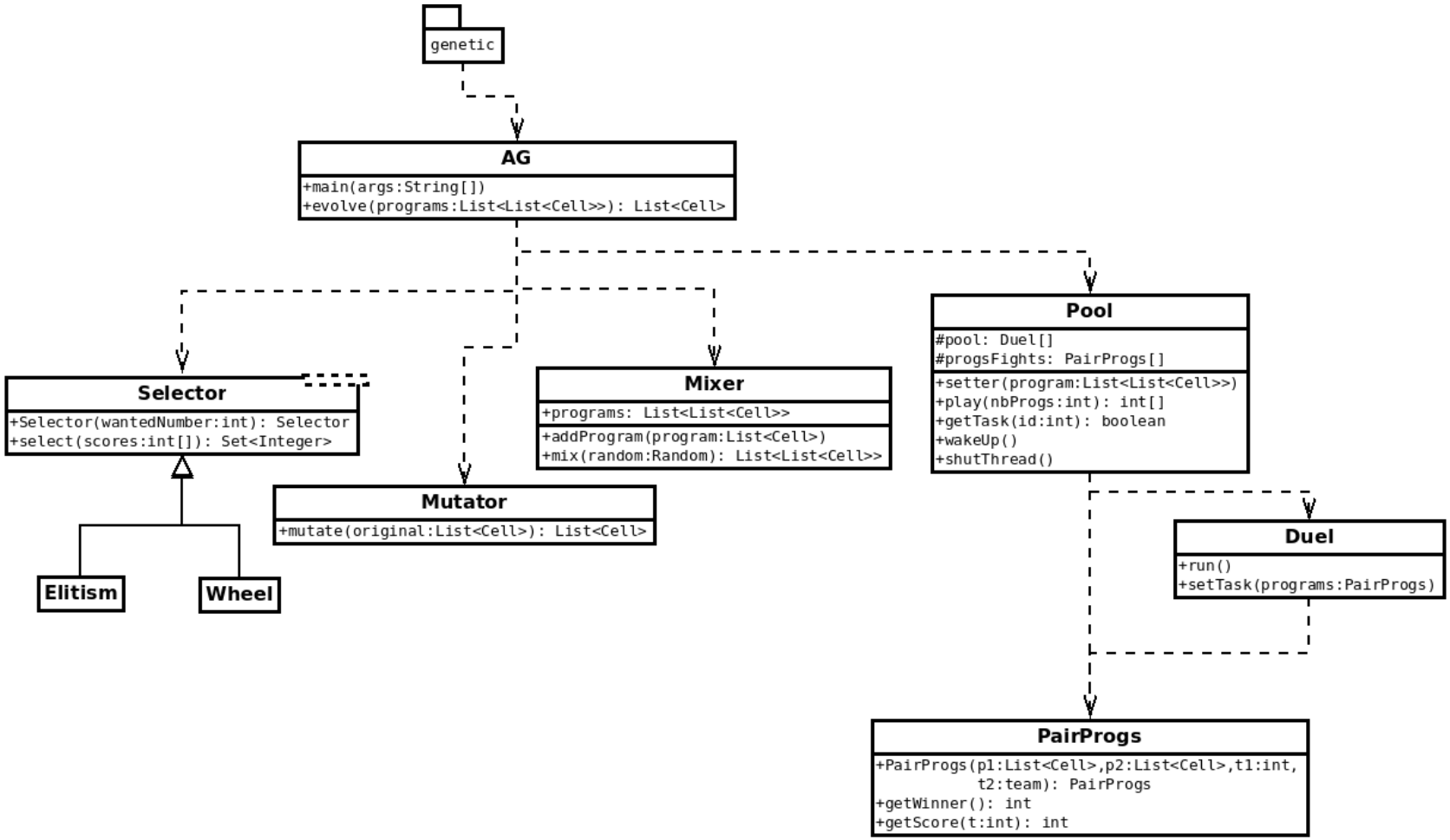

Figure: Diagramme de classe du package de l'algorithme génétique

Parrallélisme

Pool

- Initalisation:
  - Création des Threads (4 par défaut).
  - Lancement des Threads.
- Préparation:
  - Récupération des programmes à évaluer.
  - Préparation des tâches pour les Threads.
- Évaluation:
  - Réveil des Threads.
  - Attente des Threads.
  - Évaluation des résultats.
- Fin: Arrêt des Threads.

Parrallélisme

Threads

- Lancement:
  - Mise en attente.
  - Attente d'un signal de la Pool.
  - Demande de tâche à la Pool :
    - Tâche disponible: attribution et traitement, retour demande de tâche.
    - Plus de tâche disponibles: retour mise en attente.
- Fin: Arrêt du fonctionnement.

Résultats

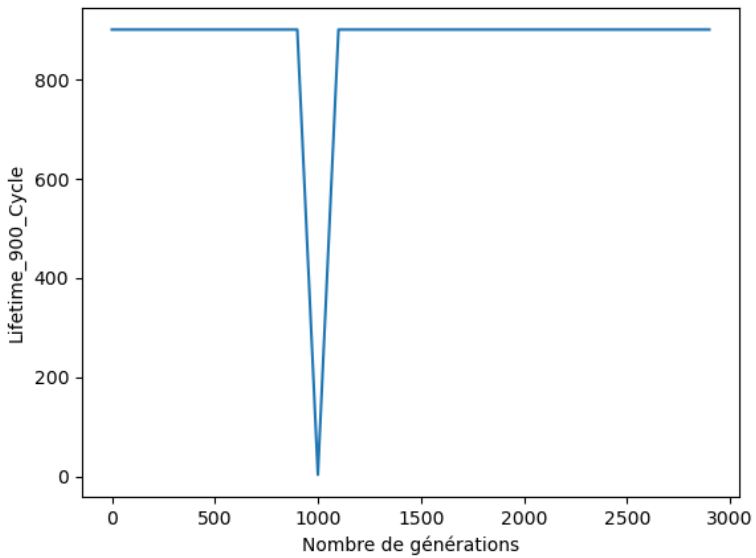

Figure: Best program lifetime over 900 cycle.

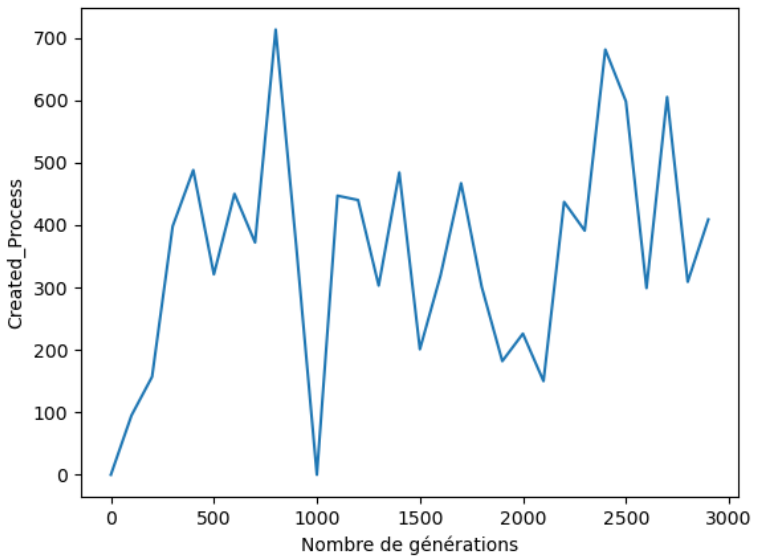

Figure: Number of created process over 900 cycle.

Résultats

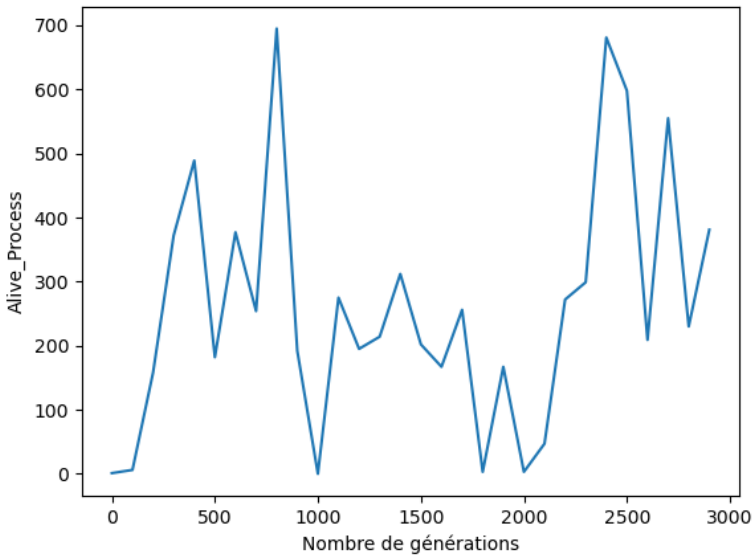

Figure: Number of alive process after 900 cycle.

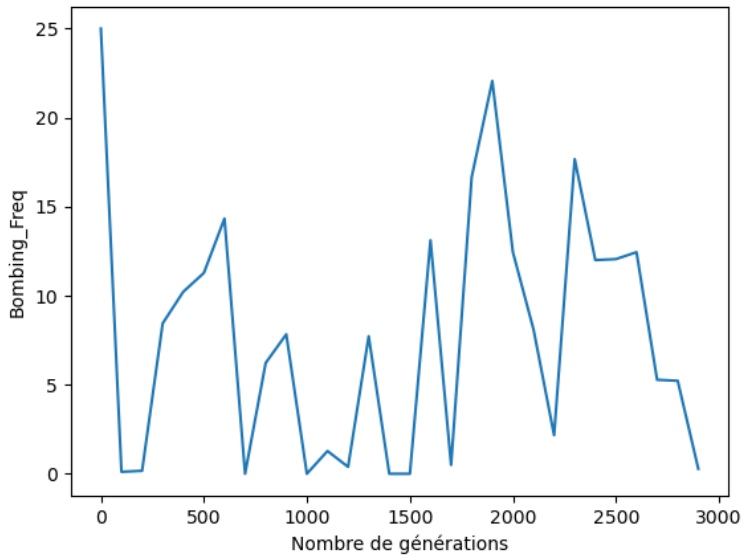

Figure: Bombing frequence of the program (average bombing over 50 cycle).

Résultats

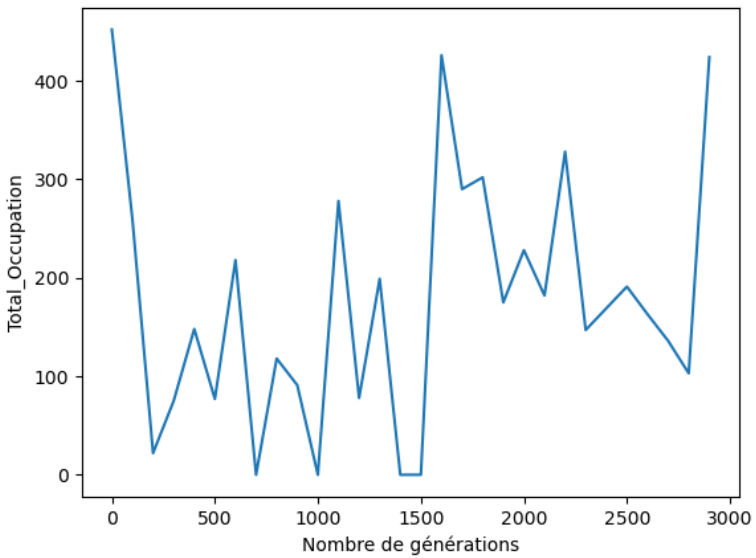

Figure: Number of written cell over 900 cycle.

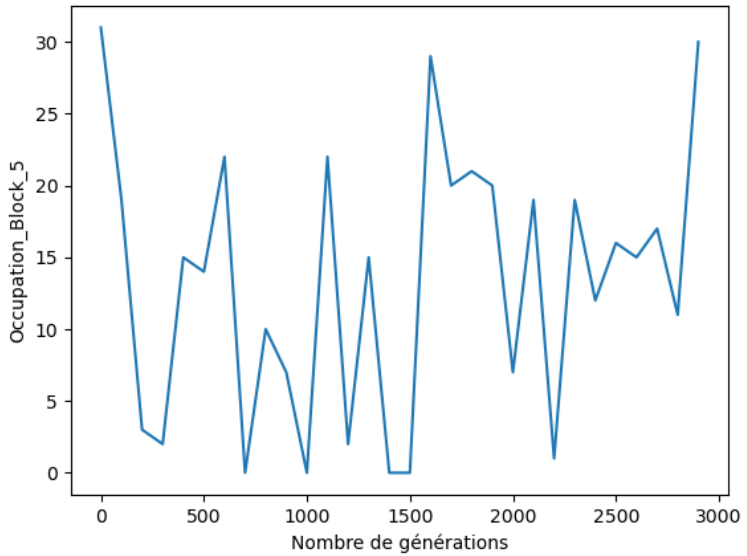

Figure: Number of block of five cell that have at least one cell written by the program on over 900 cycle.

Interface Graphique

Figure: Menu principal où l'on choisit les programmes

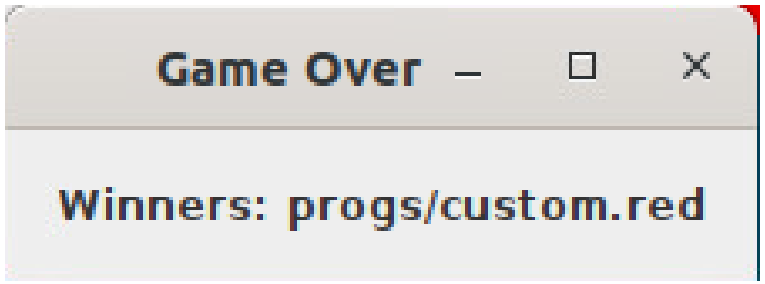

Figure: Pop-up Qui affiche le programme gagnant

Interface Graphique

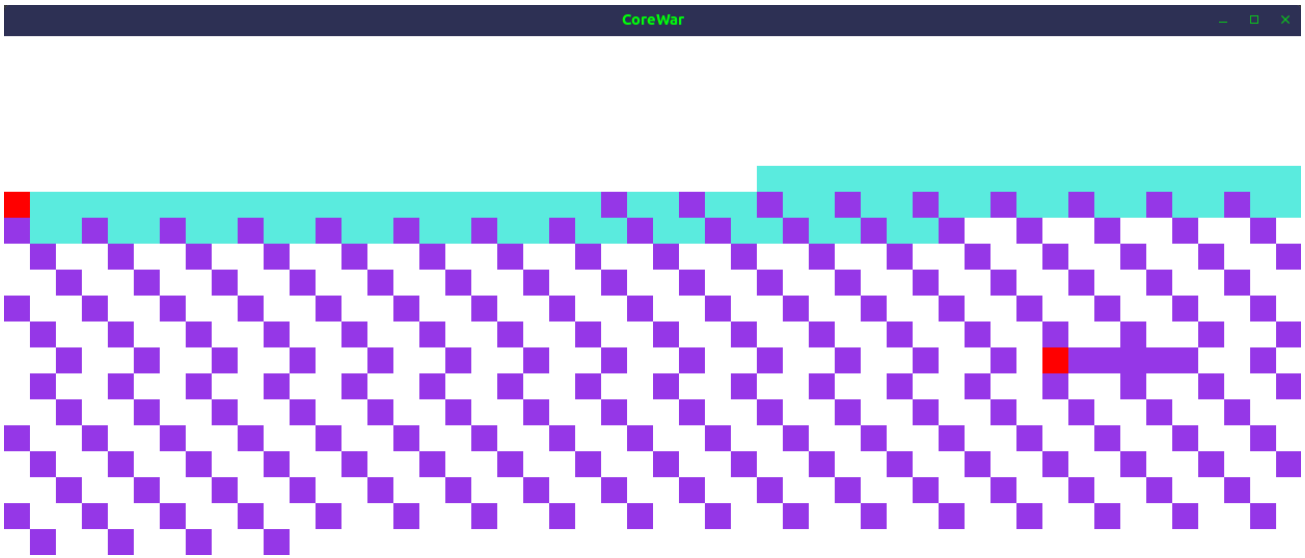

Figure: Jeu principal

Conclusion

- Objectifs atteints.
- Ce qu'on a appris.
- Pistes d'améliorations.
- Avenir du projet.