Krzysztof Dąbrowski i Jakub Bogusz

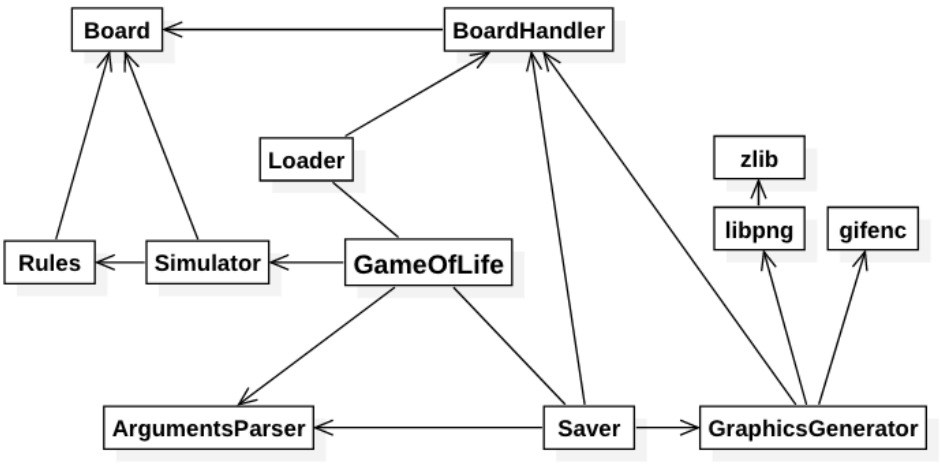
Raport końcowy – Gra w życie

9 kwietnia 2019

Spis treści

1. Ostateczny projekt modułów

Zmodyfikowany diagram modułów:



2. Opis modyfikacji

3. Prezentacja działania

Przykłady wywołania programu z różnymi flagami.

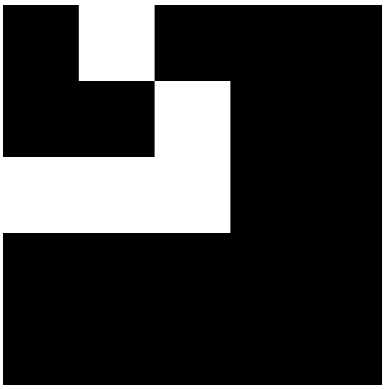
3.1. Generacja obrazów z wczytaniem stanu z pliku

Wywołanie programu:
`./game-of-life -f input.txt -t png -o raport -n 6`
Argumenty:

- `-f input.txt` – Stan początkowy jest wczytany z pliku „input.txt”,
- `-t png` – Zostanie wygenerowana seria plików .png z kolejnymi pokoleniami,
- `-o raport` – Wygenerowane pliki zostaną zapisane do folderu „raport”,
- `-n 6` – Zostanie wygenerowanych 6 następnych pokoleń .png (ze stanem początkowym 7 obrazków).

3.2. Początkowy stan

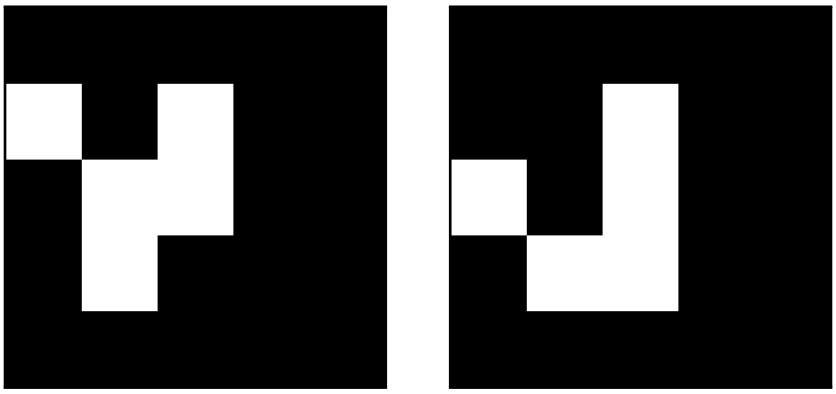
Stan wczytany z pliku został również zamieniony na obraz. Wygląda on następująco:



Rysunek 1. Stan wczytany z pliku

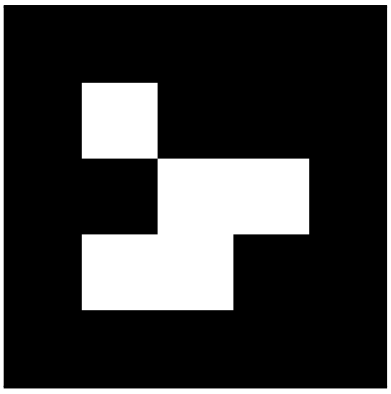
3.2.1. Powstałe obrazy

Program wygenerował 6 następnych pokoleń. Przedstawiają ruch stuktóry szybowca.

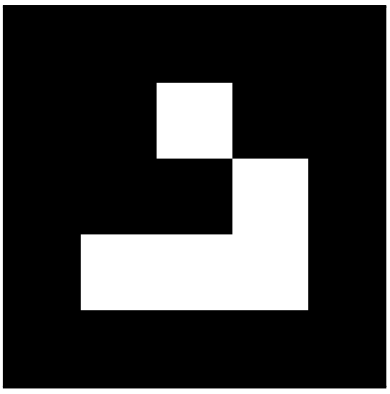


(a) Pokolenie 1

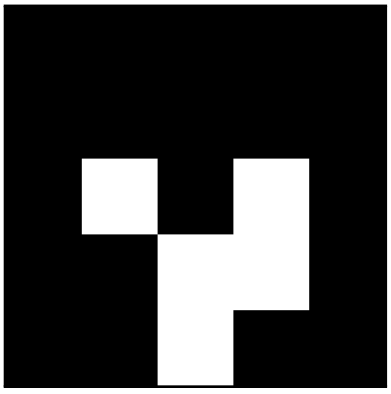
(b) Pokolenie 2



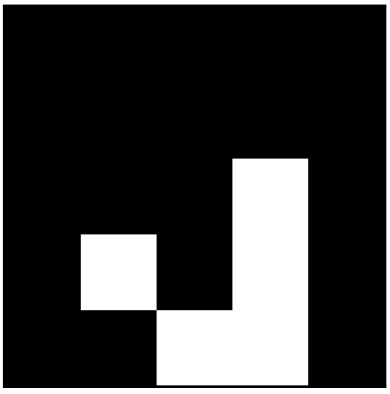
(c) Pokolenie 3



(d) Pokolenie 4



(e) Pokolenie 5



(f) Pokolenie 6

Rysunek 2. Wygenerowane pliki

4. Podsumowanie testów modułów

Do pisania oraz automatyzacji wykonania testów jednostkowych został wykorzystany framework *CUnit*. Dzięki temu na dowolnym etapie projektu można było łatwo sprawdzić poprawność już istniejącego kodu.

4.0.1. Wygenerowany raport

Wyniki testów jednostkowych przedstawia poniższy raport.

```
  CUnit - A unit testing framework for C - Version 2.1-3
  http://cunit.sourceforge.net/


Suite: Board tests
Test: To string test ...passed
Suite: Loader tests
Test: Parse small file test ...passed
Suite: Rules tests
Test: Dead cell stays dead ...passed
Test: Dead cell comes to live ...passed
Test: Alive cell dies from overpopulation ...passed
Test: Alive cell dies from loneliness ...passed
Test: Alive cell stays alive ...passed
Suite: Simulator tests
Test: Simulate one next generation on small board ...passed

Run Summary:    Type  Total    Ran Passed Failed Inactive
              suites      4      4    n/a      0        0
               tests      8      8      8      0        0
             asserts     27     27     27      0      n/a

Elapsed time =    0.001 seconds
```

4.1. Wnioski

Z raportu testów jednostkowych wynika, że wszystkie wszystkie sprawdzane funkcjonalności dają prawidłowe wyniki. Wszystkie testy są oznaczone jako `...passed` oraz liczba zapewnień, których wyniki są poprawne jest równa liczbie wszystkich zapewnień.

5. Analiza pamięci

Raport programu `valgrind` zwraca następujące wyniki:

```
==18728== LEAK SUMMARY:
==18728== definitely lost: 0 bytes in 0 blocks
==18728== indirectly lost: 0 bytes in 0 blocks
==18728== possibly lost: 72 bytes in 3 blocks
==18728== still reachable: 23,728 bytes in 28 blocks
==18728== suppressed: 18,077 bytes in 153 blocks

==18728== ERROR SUMMARY: 0 errors from 0 contexts
```

2 pierwsze linie raportu informują o braku definitywnych wycieków pamięci, co oznacza że program prawidłowo zarządza pamięcią.

Wycieki oznaczone jako `still reachable` oraz `possibly lost` spowodowane są przez funkcje bibliotek `ctime` oraz `libpng`, więc nie mamy na nie wpływu i nie jesteśmy w stanie ich wyeliminować. Załączamy fragmentu raportu opisujące te wycieki:

possible leaks:

```
==18752== 72 bytes in 3 blocks are possibly lost in loss record 34 of 60
==18752== at 0x1000B26EA: calloc (in /usr/local/Cellar/valgrind/3.14.0/lib/valgrind/vgpreload_memcheck-amd64-darwin.so)
```

```
==18752== by 0x1007AD7C2: map_images_nolock (in /usr/lib/libobjc.A.dylib)
==18752== by 0x1007C04E0: map_images (in /usr/lib/libobjc.A.dylib)
==18752== by 0x10000DC64: dyld::notifyBatchPartial(dyld_image_states, bool, char const* (*)(dyld_image_states,
unsigned int, dyld_image_info const*), bool, bool) (in /usr/lib/dyld)
==18752== by 0x10000DE39: dyld::registerObjCNotifiers(void (*)(unsigned int, char const* const*,
mach_header const* const*), void (*)(char const*, mach_header const*), void (*)(char const*, mach_header
const*)) (in /usr/lib/dyld)
==18752== by 0x10027871D: _dyld_objc_notify_register (in /usr/lib/system/libdyld.dylib)
==18752== by 0x1007AD073: _objc_init (in /usr/lib/libobjc.A.dylib)
==18752== by 0x100202B34: _os_object_init (in /usr/lib/system/libdispatch.dylib)
==18752== by 0x100202B1B: libdispatch_init (in /usr/lib/system/libdispatch.dylib)
==18752== by 0x1001119C2: libSystem_initializer (in /usr/lib/libSystem.B.dylib)
==18752== by 0x10001FAC5: ImageLoaderMachO::doModInitFunctions(ImageLoader::LinkContext const&)
(in /usr/lib/dyld)
==18752== by 0x10001FCF5: ImageLoaderMachO::doInitialization(ImageLoader::LinkContext const&)
(in /usr/lib/dyld)
```

still reachable:

```
==18805== 18,280 bytes in 1 blocks are still reachable in loss record 60 of 60
==18805== at 0x1000B26EA: calloc (in /usr/local/Cellar/valgrind/3.14.0/lib/valgrind/vgpreload_memcheck-amd64-darwin.so)
==18805== by 0x100351930: tzsetwall_basic (in /usr/lib/system/libsystem_c.dylib)
==18805== by 0x1003537C9: localtime (in /usr/lib/system/libsystem_c.dylib)
==18805== by 0x100353945: ctime (in /usr/lib/system/libsystem_c.dylib)
==18805== by 0x1000039AF: setup (Saver.c:5)
==18805== by 0x100003A2E: saveCommon (Saver.c:12)
==18805== by 0x100003E59: saveAsPng (Saver.c:53)
==18805== by 0x10000405E: runProgram (main.c:83)
==18805== by 0x100003F31: main (main.c:32)
```