

Life -- Fight!

0.1

Generated by Doxygen 1.8.1.2

Mon Mar 11 2013 22:50:34

Contents

1	README	1
2	SCENARIO	2
3	TODO	5
4	Class Index	5
4.1	Class Hierarchy	5
5	Class Index	6
5.1	Class List	6
6	Class Documentation	6
6.1	As Class Reference	6
6.1.1	Member Function Documentation	6
6.2	Test Class Reference	7
6.2.1	Detailed Description	7
6.2.2	Member Enumeration Documentation	7
6.2.3	Constructor & Destructor Documentation	8
6.2.4	Member Function Documentation	8
6.2.5	Member Data Documentation	9

1 README

eLife

- Let's play game for life!

Strategia:

x Co może być tu abstrakcyjnego?

- x abstrakcja strategii, która będzie służyła obiektom do przetrwania
- x abstrakcja obiektu-zwierzaka, z jego podstawowymi cechami i funkcjami
- x abstrakcja planszy, którą łatwo można przeprogramować na inne systemy i biblioteki

x Propozycje funkcji życiowych:

- x podstawowe potrzeby życiowe wg. Maslowa, piramida potrzeb
- x sex, potrzeba utrzymania gatunku
- x jedzenie
- x picie
- x spanie

x Jak działają funkcje życiowe

- x gdy niski poziom wody w obiekcie
 - x uruchamiamy szukajWodę()?
 - x potem rozpaczliwieSzukajWodę()?

x Jak świat wpływa na obiekty?

- x różne środowisko = różne wychowanie
- x odległość od wody = mniejsza szansa na przetrwanie?

x Rodzaje środowisk

- x mapa różnych środowisk
- x pustynia
- x dżungla
- x knieje
- x góry
- x step
- x mazury
- x coś tam innego

x Jak pogrupować obiekty?

- x roślinożercy i mięsożercy
- x ułożyć specjalną hierarchię członków gatunku
- x lista klanów?
 - x na jej czele głowy klanów
 - x najstarsi, członkowie klanu
 - x poniżej członkowie rodu niżsi i w jakichś sposób pogrupowani

x Mutacje?

- x blisko spokrewnione klany mogą generować zmodyfikowane dziecko
- x kopulacja dwóch różnych gatunków powoduje stworzenie muła?

2 SCENARIO

Propozycja dokumentacji funkcjonalnej

Fragmenty oznaczone kursywą to funkcjonalności których nie jestem pewien (czy damy radę, czy warto etc.)

Budowa programu

Aplikacja będzie się składać z dwóch modułów, nazwanych roboczo Klient i Serwer.

• Klient

- * Program, którego zadaniem jest komunikacja z użytkownikiem.
- * Na początku wyświetla okno z opcjami (zwane dalej ekranem startowym), w którym użytkownik może dobrać
- * Następnie uruchamia Serwer i przesyła mu zebrane dane, wtedy rozpoczyna się właściwa symulacja.
- * Podczas symulacji Klient odbiera od Serwera informacje o obecnych wynikach symulacji i na bieżąco wysyła
- * Do wyświetlania planszy klient używa biblioteki OpenGL.

• Serwer

- * Program, którego zadaniem jest obliczać kolejne kroki symulacji z zadaną częstotliwością i wysyłać wyniki
- * Serwer może także przyspieszać, zwalniać lub wstrzymywać symulację na żądanie użytkownika (przekazywane

Klient i serwer komunikują się za pomocą współdzielonej pamięci, która jest zrealizowana przy pomocy biblioteki Boost::Interprocess.

Świat gry

Symulacja rozgrywa się na prostokątnej planszy, której wymiary ustala użytkownik na ekranie startowym. Początkowo obiekty zostają umieszczone w sposób losowy, ale z ograniczeniami opisanymi niżej. Rodzaje obiektów:

- osobnik (roślinożerca lub drapieżnik)

- * posiada zestaw indywidualnych cech; stanowi punkt (nie ma wymiarów), może się poruszać

- drzewo

- * źródło pożywienia dla roślinożerców; stanowi punkt (nie ma wymiarów)

- wodopój

- * źródło wody dla osobników; stanowi punkt (nie ma wymiarów)

- jaskinia

- * miejsce, w którym osobniki mogą spać; stanowi punkt (nie ma wymiarów)

- skały

- * obszar na planszy, po którym osobniki nie mogą chodzić, stanowi koło o promieniu 5 jednostek; takie k

Cechy (parametry) osobników

Każdy osobnik posiada zestaw indywidualnych cech, ustalanych w momencie narodzin (liczby całkowite):

- Zasięg widzenia (ozn. R), R [5, 100]
- Szybkość biegu (V), V [5, 100]
- Odporność na głód (F), F [5, 100]
- Odporność na pragnienie (W), W [5, 100]
- Wytrzymałość [odporność na zmęczenie] (S), S [5, 100]
- Wydajność reprodukcyjna (P), P [5, 100]
- Maksymalny czas życia (L), L [5, 100]
- Płeć (X), X {„F”, „M”}

Zależności

- Indywidualne cechy osobnika (z wyjątkiem płci) mają tę własność, że „więcej = lepiej”. Spełniają one warunek: $R+V+F+W+S+P = 200$
- A także parametry chwilowe (liczby rzeczywiste, zmieniające się w sposób pseudo-ciągły): Poziom najeżenia (ozn. f), f [0, F], maleje w stałym tempie
- Jeśli poziom najeżenia spadnie poniżej połowy (f [0, $\frac{1}{2}F$]), to osobnik znajduje się w stanie „głodny”
- Jeśli głodny osobnik dotrze do drzewa (roślinożerca) lub złapie roślinożercę (drapieżnik) jego poziom najeżenia natychmiast rośnie do poziomu F .
- Roślinożerca zjedzony przez drapieżnika znika z gry.
- Jeśli poziom najeżenia spadnie do zera, osobnik umiera z głodu

Poziom napojenia

- Poziom napojenia (w) w $[0, W]$, maleje w stałym tempie
- Jeśli poziom napojenia spadnie poniżej połowy ($w [0, \frac{1}{2}W]$), to osobnik znajduje się w stanie „spragniony”.
- Jeśli spragniony osobnik dotrze do wodopoju, jego poziom napojenia natychmiast rośnie do poziomu W
- Jeśli poziom napojenia spadnie do zera, osobnik umiera z pragnienia

Poziom energii

Poziom energii (wypoczęcia) (s) s $[0, S]$, maleje w zmiennym tempie

- Jeśli poziom energii spadnie poniżej połowy ($s [0, \frac{1}{2}S]$), to osobnik znajduje się w stanie „zmęczony”.
- Jeśli głodny osobnik dotrze do kryjówki, jego poziom napojenia energii rośnie do poziomu S , ale jednocześnie zapada on w sen trwający stały czas TS .
- W czasie snu osobnik nie porusza się, jest niewidoczny dla innych osobników, ale może zostać zjedzony przez drapieżnika, który akurat wejdzie do tej samej kryjówki.
- Jeśli poziom energii spadnie do zera, osobnik umiera z wycieńczenia Poziom energii maleje dwa razy szybciej podczas biegu.

Poziom zaspokojenia reprodukcyjnego

Czas do reprodukcji (p), $p [0, 1]$, maleje w stałym tempie (zależnym od wartości P)

- Jeśli $p = 0$, to osobnik może się rozmnażać; po akcie „kopulacji” wartość p jest ustawiana na 1.
- Wartość ta maleje w tempie $P/1000$ na sekundę (czyli osobnik o najwyższej możliwej zdolności reprodukcyjnej $P=100$ będzie „pauzować” 10 sekund, a o najniższej możliwej $P=5$, będzie „pauzować” 200 sekund).

Wiek

Wiek (l), $l [0, L]$, rośnie w stałym tempie

- Przy narodzinach jest ustawiane $l = 0$, po osiągnięciu $l = L$ osobnik umiera ze starości

Zachowanie osobników

Każdy osobnik porusza się po planszy w sposób losowy, dopóki w jego polu widzenia (okrąg o promieniu R) nie znajdzie się jakiś interesujący go obiekt:

- Jeśli w polu widzenia osobnika znajduje się jaskinia i osobnik jest zmęczony, to idzie w stronę wodopoju – priorytet 1.
- Jeśli w polu widzenia osobnika znajduje się drzewo, osobnik jest głodny i jest roślinożercą, to osobnik idzie w stronę drzewa – priorytet 2.
- Jeśli w polu widzenia osobnika znajduje się wodopój i osobnik jest spragniony, to osobnik idzie w stronę wodopoju – priorytet 3.

- Jeśli w polu widzenia osobnika znajduje się drugi osobnik tego samego gatunku i przeciwnej płci oraz oba te osobniki mają $p=0$ (są gotowe do reprodukcji) oraz żaden z nich nie jest głodny, spragniony, ani zmęczony, to osobniki idą w swoim kierunku – priorytet 4.
- Jeśli w polu widzenia osobnika znajduje się roślinożerca, osobnik jest głodny i jest drapieżnikiem, to osobnik biegnie w stronę roślinożercy (poluje) – priorytet 5.
- Jeśli w polu widzenia osobnika znajduje się drapieżnik i osobnik jest roślinożercą, to osobnik biegnie w stronę przeciwną (ucieka), niezależnie od tego, czy drapieżnik jest głodny – priorytet 5.

Jakieś zachowania społeczne – osobniki mogą trzymać się innych osobników swojego gatunku, wspólnie polować etc. – tylko nie mam pojęcia, jak to realizować.

Jeśli w zasięgu wzroku osobnika znajduje się kilka interesujących obiektów, to wybiera akcję o najwyższym priorytecie (najpierw polowanie / ucieczka itd.).

Rozmnażanie

Jeśli dwa osobniki tego samego gatunku i różnych płci spotkają się, następuje akt prokreacji – pojawia się nowy osobnik, którego cechy (R, V, F, W, S, P, L) wynikają z odpowiednich cech rodziców:

```
RDZIECKA = random(ROJCA, RMATKI) + random(-10, 10)
```

(analogicznie dla pozostałych cech) gdzie `random(a,b)` jest funkcją zwracającą losową wartość z zakresu $[\min(a,b); \max(a,b)]$. Cechy są skalowane w taki sposób, aby spełniały warunki: $R+V+F+W+S+P+L=200$; $R, V, F, W, S, P, L \in [5, 100]$

Możliwości modyfikacji scenariusza

Na ekranie startowym użytkownik może ustalić pewne parametry symulacji, takie jak:

- wielkość planszy
- gęstość rozmieszczenia drzew, wodopojów, skał, ilość drapieżników i roślinożerców etc.,

Może także wpłynąć na określone cechy całej populacji (np. dodać wszystkim roślinożercom +30 do szybkości).

Do programu będzie też dołączonych kilka (wybranych przez twórców gry) ciekawych scenariuszy.

Wizualizacja

Osobniki i inne obiekty będą reprezentowane przez proste figury geometryczne, po najechnięciu myszką na obiekt wyświetli się więcej informacji o nim.

3 TODO

4 Class Index

4.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

Test	7
As	6

5 Class Index

5.1 Class List

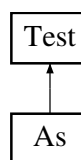
Here are the classes, structs, unions and interfaces with brief descriptions:

As	6
Test	
A test class	7

6 Class Documentation

6.1 As Class Reference

Inheritance diagram for As:



Public Member Functions

- void [testMeToo](#) (char c1, char c2)
 A pure virtual member.

Additional Inherited Members

6.1.1 Member Function Documentation

6.1.1.1 void As::testMeToo (char c1, char c2) [virtual]

A pure virtual member.

See Also

[testMe\(\)](#)

Parameters

<i>c1</i>	the first argument.
<i>c2</i>	the second argument.

Implements [Test](#).

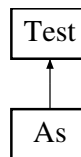
The documentation for this class was generated from the following file:

- src/main.cpp

6.2 Test Class Reference

A test class.

Inheritance diagram for Test:



Public Types

- enum `TEnum` { `TVal1`, `TVal2`, `TVal3` }
An enum.

Public Member Functions

- `Test` ()
A constructor.
- `~Test` ()
A destructor.
- int `testMe` (int a, const char *s)
a normal member taking two arguments and returning an integer value.
- virtual void `testMeToo` (char c1, char c2)=0
A pure virtual member.

Public Attributes

- enum `Test::TEnum` * `enumPtr`
enum pointer.
- enum `Test::TEnum` `enumVar`
enum variable.
- int `publicVar`
a public variable.
- int(* `handler`)(int a, int b)
a function variable.

6.2.1 Detailed Description

A test class.

A more elaborate class description.

6.2.2 Member Enumeration Documentation

6.2.2.1 enum `Test::TEnum`

An enum.

More detailed enum description.

Enumerator:

TVal1 enum value TVal1.

TVal2 enum value TVal2.

TVal3 enum value TVal3.

6.2.3 Constructor & Destructor Documentation

6.2.3.1 Test::Test ()

A constructor.

A more elaborate description of the constructor.

6.2.3.2 Test::~~Test ()

A destructor.

A more elaborate description of the destructor.

6.2.4 Member Function Documentation

6.2.4.1 int Test::testMe (int a, const char * s)

a normal member taking two arguments and returning an integer value.

Parameters

a	an integer argument.
s	a constant character pointer.

See Also

[Test\(\)](#)
[~Test\(\)](#)
[testMeToo\(\)](#)
[publicVar\(\)](#)

Returns

The test results

6.2.4.2 virtual void Test::testMeToo (char c1, char c2) [pure virtual]

A pure virtual member.

See Also

[testMe\(\)](#)

Parameters

c1	the first argument.
c2	the second argument.

Implemented in [As](#).

6.2.5 Member Data Documentation

6.2.5.1 `enum Test::TEnum * Test::enumPtr`

enum pointer.

Details.

6.2.5.2 `enum Test::TEnum Test::enumVar`

enum variable.

Details.

6.2.5.3 `int(* Test::handler)(int a, int b)`

a function variable.

Details.

6.2.5.4 `int Test::publicVar`

a public variable.

Details.

The documentation for this class was generated from the following file:

- `src/main.cpp`

Index

- ~Test
 - Test, [7](#)
- As, [5](#)
 - testMeToo, [6](#)
- enumPtr
 - Test, [8](#)
- enumVar
 - Test, [8](#)
- handler
 - Test, [8](#)
- publicVar
 - Test, [8](#)
- TVal1
 - Test, [7](#)
- TVal2
 - Test, [7](#)
- TVal3
 - Test, [7](#)
- TEnum
 - Test, [7](#)
- Test, [6](#)
 - ~Test, [7](#)
 - enumPtr, [8](#)
 - enumVar, [8](#)
 - handler, [8](#)
 - publicVar, [8](#)
 - TVal1, [7](#)
 - TVal2, [7](#)
 - TVal3, [7](#)
 - TEnum, [7](#)
 - Test, [7](#)
 - testMe, [7](#)
 - testMeToo, [8](#)
- testMe
 - Test, [7](#)
- testMeToo
 - As, [6](#)
 - Test, [8](#)