UNIWERSYTET GDAŃSKI

Zaawansowane języki programowania

Symulator mrowiska

Ruby

Karolina Jędrzejewska
Karol Kawski

UNIWERSYTET GDAŃSKI

Harmonogram prezentacji

1. Założenia projektu
2. Wykorzystane technologie
3. Implementacja
4. Kluczowe mechanizmy
5. Refaktoryzacja
6. Prezentacja aplikacji

UNIWERSYTET GDAŃSKI

Założenia projektu

Stworzenie graficznej aplikacji symulującej mrowisko, w którym mrówki zabijają się nawzajem.

UNIWERSYTET GDAŃSKI

Wykorzystane technologie

Simple2d - to niewielki silnik graficzny o otwartym kodzie źródłowym, zapewniający podstawowe możliwości rysowania, multimediów i wprowadzania danych 2D. Jest napisany w języku C i działa na wielu platformach, tworząc rodzime okna i wchodząc w interakcję ze sprzętem za pomocą SDL podczas renderowania zawartości za pomocą OpenGL.

Ruby2d - biblioteka wieloplatformowa pozwala tworzyć aplikacje, gry oraz wizualizacje

ascii_charts - biblioteka generująca wykresy w terminalu

UNIWERSYTET GDAŃSKI

Implementacja cz.1

Aplikacja składa się z 3 klas (Board, Ant, Counter) oraz funkcji pomocniczych.

- Board – instancja tworzy planszę mrowiska, na której znajdą się mrówki

- Counter – tworzy licznik, który pokazuje ilość żywych mrówek w mrowisku z metodą, która aktualizuje jego stan po każdym ticku

- Ant – odpowiada za mrówki, ich położenie, ruch w każdym ticku jak również posiada metodę, która sprawdza czy mrówki są na tyle blisko siebie aby mogły się zabić.

UNIWERSYTET GDAŃSKI

Implementacja cz.2

Class Ant < Sprite

- setXCord(x, w, value) - metoda pozwalająca ustawić koordynat x
- setYCord(y, w, value) - metoda pozwalająca ustawić koordyna y
- getXCord() - metoda zwracająca aktualny koordynat x
- getYCord() - metoda zwracająca aktualny koordynat y
- crash(j, x, y, ary, toDelete) - metoda obsługująca zderzenia obiektów klasy Ant

Klasa nadrzędna Sprite należy do gemu ruby2d i odpowiada za tworzenie animacji z obrazów.

UNIWERSYTET GDAŃSKI

Implementacja cz.3

Class Counter < Text

- setText(text) - metoda ustawiająca tekst w naszym liczniku

Klasa nadrzędna Text należy do gemu ruby2d i odpowiada za obsługę obiektów tekstowych .

funkcja update - główna funkcja programu tworząca wszystkie obiekty oraz wywołująca metody. Wywołuje się co 10ms i odświeża pozycje obiektów w oknie aplikacji.

UNIWERSYTET GDAŃSKI

Refaktoryzacja

Podczas refaktoryzacji wykorzystywane zostały zasady ze strony refactoring.com, a do metryk **Flog** i **Flay**

Nazewnictwo
Nazwy zmiennych i metod zmienione zostały na dokładnie opisujące dany stan i czynność, np:

$howManyAnts, $antWidth, $antHeight
drawChart(), playSong(), removeAnts(), insertAnts()

UNIWERSYTET GDAŃSKI

Refaktoryzacja

If… else?

Liczba wystąpień zredukowana liczba do minimum wyjątki, np:

```
if ((x-antWidth...x+antWidth).include?(ary[i].x) &&
(y-antHeight...y+antHeight).include?(ary[i].y))
        #if we find another ant, we push them into toDelete array
        $toDelete.push(i)
end
```

Puts

Wykorzystane tylko do wyświetlania wykresu w konsoli:

```
def drawChart()
    ## data must be a pre-sorted array of x,y pairs
    chart = AsciiCharts::Cartesian.new(($time), :title => 'Time/Population').draw
    puts "\e[H\e[2J"
    puts chart
end
```

UNIWERSYTET GDAŃSKI

Refaktoryzacja

Zawartość funkcji

Podział funkcji na mniejsze. Metody są możliwie jak najkrótsze i odpowiadają za dokładnie jedną funkcjonalność

```
def updateTimeArry(ary)
        endTime = Time.now
        $time.push([ary.length, (endTime-$startTime)])
end
```

UNIWERSYTET GDAŃSKI

Refaktoryzacja

Parametry

przekazywanie parametrów zmniejsza potrzebę dużej ilości zmiennych globalnych.

W tym projekcie: max 3 parametry

Usunięcie powtórzeń kodu

Dzięki bibliotece **flay** wyeliminowane zostały powtórzenia:

before:	***after***:
Total score (lower is better) = 96 1) Similar code found in :defn (mass = 62) script.rb:38 script.rb:45 2) Similar code found in :defn (mass = 34) script.rb:58 script.rb:61	Total score (lower is better) = 0

UNIWERSYTET GDAŃSKI

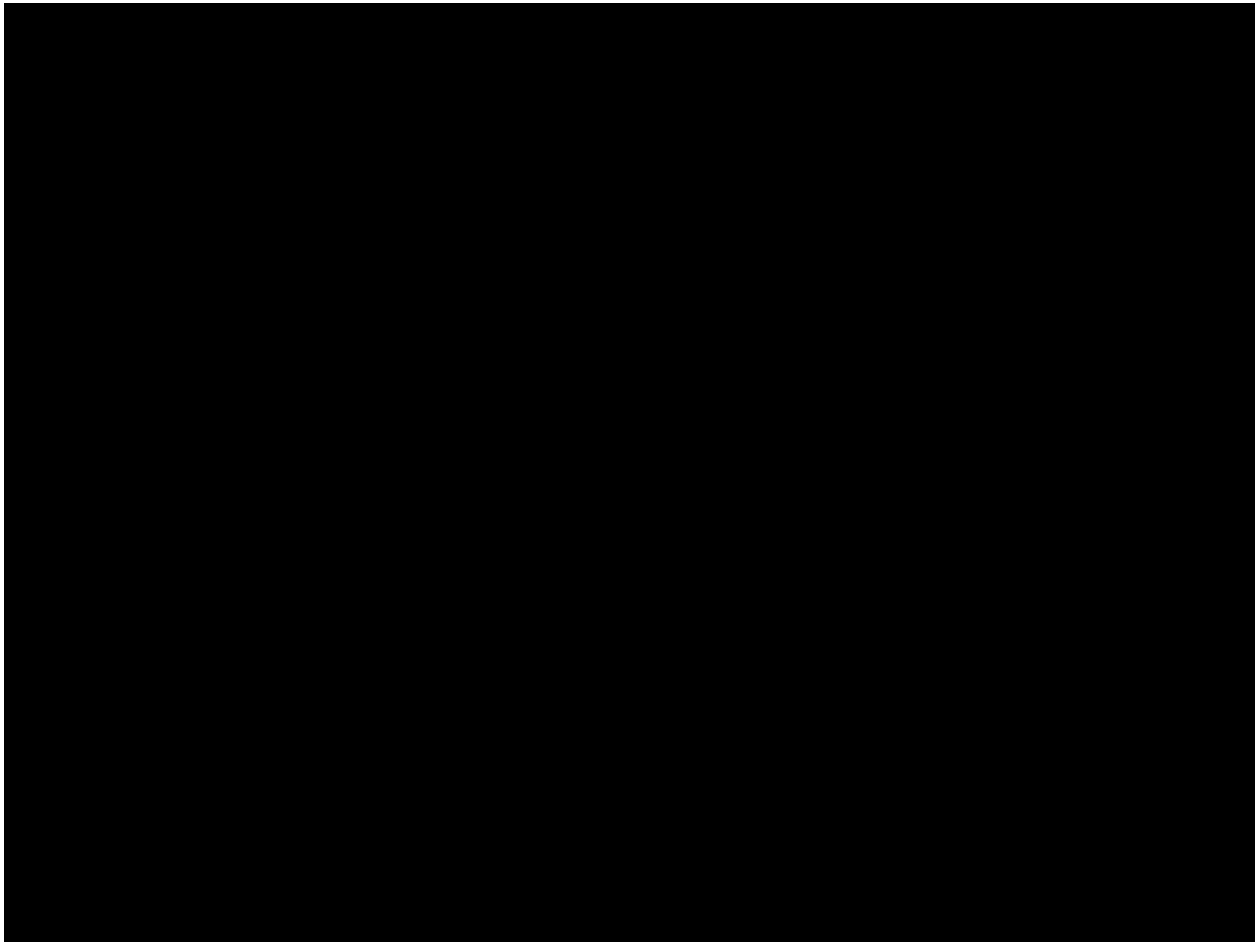
Refaktoryzacja

Złożoność kodu:

215.7: flog total **43.1: flog/method average** 151.5: main#none 29.1: Ant#crash 17.0: Ant#setYCord 17.0: Ant#setXCord	**214.0: flog total** 30.6: flog/method average … **207.2: flog total** 23.0: flog/method average … **189.9: flog total** 12.7: flog/method average	**158.0: flog total** **11.3: flog/method average** 49.9: main#none 28.5: Ant#crash 15.4: Ant#changeCords 13.8: main#insertAnts 13.1: main#updateBoardView 7.0: Ant#updatePosition 6.2: Counter#upDateC 4.9: main#updateTimeArry 4.7: main#drawChart 3.7: Ant#setXCord 3.7: Ant#setYCord 3.5: main#removeAnts 2.5: main#playSong 1.1: Board::info

UNIWERSYTET GDAŃSKI

Prezentacja aplikacji



UNIWERSYTET GDAŃSKI

DZIĘKUJEMY ZA UWAGĘ

Repozytorium:
github.com/mrfrappe/UG_Zaawansowane_jezyki_programowania