POLITECHNIKA WROCŁAWSKA

AUTOMATYKA I ROBOTYKA, ARR, W4

Wizualizacja Danych Sensorycznych

SPRAWOZDANIE OKRESOWE NR 1/3
- WSTĘPNE REZULTATY -

Wizualizacja rozkładu ciśnienia cieczy na podstawie symulacji komputerowej

Autorzy: Balawender Adam Kwieciński Krzysztof Prowadzący: Dr inż. Bogdan Kreczmer

20 kwietnia 2015

Spis treści

1	Opis projektu	1		
2	Plan pracy2.1 Podział obowiązków2.2 Harmonogram2.3 Kamienie milowe2.4 Diagram Gantta	2		
3	Funkcjonalności aplikacji	3		
4	1 Interfejs graficzny			
5	Diagram klas			
6	Przepływ sterowania			

1 Opis projektu

Zgodnie z tematem projektu zajmiemy się komputerową symulacją zachowania cieczy oraz wizualizacją jej stanu i rozkładu ciśnienia w zbiorniku z płynem.

Symulacja będzie obejmowała ruch cieczy w przekroju 2D wybranego naczynia. Ciecz zostanie przedstawiona na płaszczyźnie jako zbiór oddziaływujących ze sobą cząsteczek. Postaramy się, żeby jej zachowanie było możliwie zbliżone do rzeczywistego. Ruch płynu zostanie zamodelowany metodą numeryczną SPH (smoothed particle hydrodynamics - wygładzona hydrodynamika cząstek). Pozwoli to na realistyczne odwzorowanie zachowania cieczy. Możliwe będzie badanie cieczy o różnych parametrach, dlatego też modelowane będą jej właściwości fizyczne: gęstość i lepkość. Dodatkowo mierzone będzie ciśnienie cieczy i zostanie ono zwizualizowane jako odcień koloru płynu. Im będzie on ciemniejszy, tym wyższe ciśnienie będzie odzwierciedlał.

2 Plan pracy

2.1 Podział obowiązków

Projekt zakłada powiązanie symulacji numerycznej (back-end) z aplikacją prezentującą wyniki w formie graficznej (front-end). Za pierwszą z ww. części odpowiedzialny będzie Adam Balawender, za drugą Krzysztof Kwieciński. Obie części powinny mieć możliwość niezależnego uruchomienia, co ułatwi ich testowanie we wstępnych etapach oraz ocenę w końcowym etapie projektu.

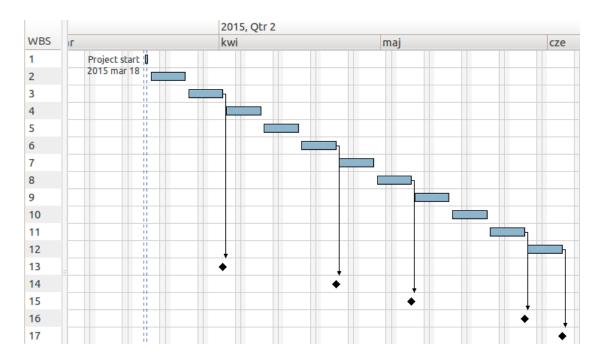
2.2 Harmonogram

Tydzień	Adam	Krzysztof	
Ι	Opis projektu		
II	Przegląd bibliotek Qt, szkic GUI		
III	Zapoznanie się z metodą SPH (Smoothed Particle Hydrodynamics)		
IV	Ustalenie struktur danych oraz API modułów		
V	Implementacja klas zbiornika oraz cząsteczek cieczy	Stworzenie statycznej wizualizacji zbiornika	
VI	Implementacja metod uaktualniania położenia cząsteczek	Dodanie wizualizacji położenia czą- stek cieczy	
VII	Analiza błędów działania programu		
VIII	Skorygowanie działania programu		
IX	Wyznaczanie ciśnienia w punkach zbiornika	Wizualizacja ciśnienia w punktach	
X	Weryfikacja projektu z założeniami		
XI	Odpowiednie modyfikacje programu		
XII	Napisanie raportu końcowego		

2.3 Kamienie milowe

- $\mathrm{K}1.$ Prze
analizowanie artykułów na temat SPH i zapoznanie się z t
ą metodą
- K2. Zaimplementowanie struktur danych, modelu cieczy i relacji między cząsteczkami
- K3. Wizualizacja symulowanego stanu cieczy
- K4. Wizualizacja ciśnienia w poszczególnych punktach zbiornika
- K5. Skończona dokumentacja

2.4 Diagram Gantta



Rysunek 1: Diagram Gantta

3 Funkcjonalności aplikacji

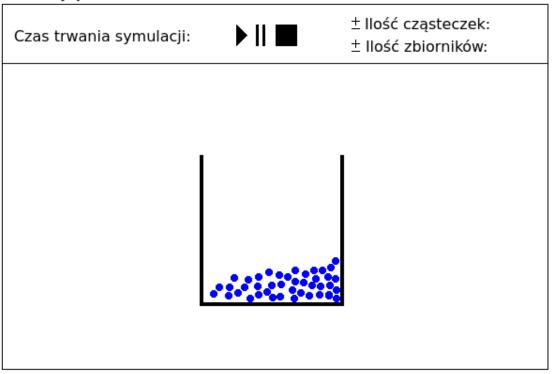
Najistotniejszymi funkcjonalnościami aplikacji będą:

- symulacja zachowania cieczy w zależności od zadanych warunków początkowych,
- możliwość przedefiniowania parametrów cieczy (gęstości, lepkości),
- możliwość obserwacji wyniku symulacji (położenia cząsteczek i rozkładu ciśnień).

4 Interfejs graficzny

Szkic planowanego interfejsu graficznego aplikacji przedstawiony jest na rysunku 2.

Plik Edycja Widok Pomoc



Rysunek 2: Szkic interfejsu graficznego

5 Diagram klas

Diagram przepływu sterowania przedstawiony jest na rysunku 3.

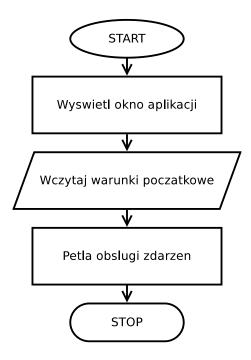
```
Wektor
                                                                                                       Zbiornik
x: double
                                                                                szerokosc: double
  y: double
                                                                                _wysokosc: double
                                                                                __ilosc_czasteczek: int = 0
+dodaj(in a:const Wektor&,in b:const Wektor&): Wektor const
                                                                                _czasteczki: list<Czasteczka>
+odejmij(in a:const Wektor&,in b:const Wektor&): Wektor const
+pomnoz(in a:const Wektor&,in b:const Wektor&): Wektor const
                                                                               +Zbiornik(in szerokosc:double,in wysokosc:double,
                                                                               in ilosc_czasteczek:int=0)
+maksymalnaPojemnosc(): double const
+Wektor(x:double,y:double)
+Wektor()
+x(): double&
                                                                                pojemnoscAktualna(): double const
+x(): double const
+y(): double&
                                                                               +dodajCzasteczke(): void
+usunCzasteczke(): Czasteczka
+y(): double const
                                                                                ⊦wypiszZawartosc(): void const
                                                                               +szerokosc(): double&
+szerokosc(): double const
                       Czasteczka
                                                                               +wysokosc(): double&
+wysokosc(): double const
+ilosc_czasteczek(): int const
-_promien: double
 _.
_lepkosc: double
- gestosc: double
```

```
+czasteczki(): list<Czasteczka>&
-_gestose. doubte
-_polozenie: Wektor
-_kolor: string
                                                                                                   PanelSterowania
+stworzNCzasteczek(in ilosc:int=0): list
+Czasteczka(in promien:double,in lepkosc:double,
                                                                                            _plik_testowy: string
                in gestosc:double,in polozenie:Wektor,
                                                                                            -_ilosc_zbiornikow: int = 1
-_zbiorniki: list<Zbiornik>
                in kolor:string)
+Czasteczka()
                                                                                           +play(): void
+pause(): void
+promien(): double&
+promien(): double const
                                                                                            ⊦stop(): void
+lepkosc(): double&
                                                                                            +dodajZbiornik(): void
+usunZbiornik(): Zbiornik
+lepkosc(): double const
+gestosc(): double&
+gestosc(): double const
                                                                                            +plik_testowy(): string&
                                                                                            +plik_testowy(): string const
+ilosc_zbiornikow(): int const
+polozenie(): Wektor&
+polozenie(): Wektor const
                                                                                            +zbiorniki(): list<Zbiornik>&
+kolor(): string&
+kolor(): string const
```

Rysunek 3: Diagram klas dla interfejsu graficznego

6 Przepływ sterowania

Diagram przepływu sterowania przedstawiony jest na rysunku 4.



Rysunek 4: Diagram przepływu sterowania dla interfejsu graficznego