Symulacja komputerowa zaniku sygnału luminescencyjnego w skaleniach

Generated by Doxygen 1.8.12

Contents

Chapter 1

Symulacja komputerowa zaniku sygnału luminescencyjnego w skaleniach

Chapter 2

Class Index

2.1 Class List

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

Chapter 3

Class Documentation

3.1 Crystal Class Reference

`#include <Crystal.h>`

Public Member Functions

- Crystal (long long int n_el, long long n_holes, double min, double max)
- double changeTime (unsigned long time) const
- void startSimulation (int time)
- void tunnelEffect (Trap &trap, int time)
- unsigned long countElectrons () const

  funkcja zliczająca ilość elektronów w pułapkach
- void saveToFile (std::string)

  zapis wyniku do plików

Static Public Member Functions

- static double tunnelEffectProbability (double time, double tau)

Private Member Functions

- double calculateTau (double distance, ElectronHole ∗el_hole, const Trap &trap) const
- double calculateDistance (const Trap &trap, const ElectronHole ∗hole) const

Private Attributes

- const double S = 3e15

  wartość stałej 'attempt-to-escape frequency'
- std::map< unsigned long, unsigned long > amount_electrons

  mapa klucz - czas, wartosc - ilosc elektronow w stanie wzbudzonym
- std::vector< Trap > el_traps

  wektor przechowujący wszystkie pułapki
- std::vector< Electron ∗ > electrons

  wektor przechowujący wskaźniki do elektronów
- std::vector< ElectronHole ∗ > electron_holes

  wektor przechowujący wskaźniki do dziur elektronowych

3.1.1 Detailed Description

Klasa reprezentująca kryształ

3.1.2 Constructor & Destructor Documentation

3.1.2.1 Crystal()

```
Crystal::Crystal (
            long long int n_el,
            long long n_holes,
            double min,
            double max )
```

Konstruktor

Parameters

n_el	ilosc obiektów elektronu do stworzenia
n_el	ilosc obiektów dziury elektronowej do stworzenia
min	dolna granica którą mogą przyjmować współrzędne cząstek
max	górna granica którą mogą przyjmować współrzędne cząstek

3.1.3 Member Function Documentation

3.1.3.1 calculateDistance()

```
double Crystal::calculateDistance (
            const Trap & trap,
            const ElectronHole * hole ) const  [private]
```

Funkcja do obliczania odległości między pułapką a dziurą

Parameters

trap	referencja do pułapki
hole	wskaznik na dziurę elektronową

Returns

odległość między parametrami

3.1.3.2 calculateTau()

```
double Crystal::calculateTau (
            double distance,
```

```
            ElectronHole * el_hole,
            const Trap & trap ) const  [private]
```

Funkcja do obliczania wartosci tau

See also

calculateDistance()

Parameters

distance	dystans do przetunelowania
el_hole	wskaznik na dziurę elektronową
trap	referencja do pułapki

Returns

wartość tau

3.1.3.3 changeTime()

```
double Crystal::changeTime (
            unsigned long time ) const
```

Funkcja zmieniajaca jednostę czasu

Parameters

time	czas do zamiany

Returns

czas podany w jednostce log10(t/2dni)

3.1.3.4 countElectrons()

```
unsigned long Crystal::countElectrons ( ) const
```

funkcja zliczająca ilość elektronów w pułapkach

Returns

ilość elektronów w pułapkach

3.1.3.5 saveToFile()

```
void Crystal::saveToFile (
            std::string name )
```

zapis wyniku do plików

Parameters

name	nazwa pliku wyjściowego

3.1.3.6 startSimulation()

```
void Crystal::startSimulation (
            int time )
```

Funkcja rozpoczynająca symulacje

Parameters

time	symulowany czas działania

3.1.3.7 tunnelEffect()

```
void Crystal::tunnelEffect (
            Trap & trap,
            int time )
```

Funkcja wykonująca efekt tunelowania

Parameters

trap	referencja do pułapki
time	czas

3.1.3.8 tunnelEffectProbability()

```
double Crystal::tunnelEffectProbability (
            double time,
            double tau )  [static]
```

Funkcja do obliczania prawdopodobienstwa NIEZAJŚCIA tunelowania

Parameters

time	czas
tau	wartość tau

See also

calculateTau()

Returns

wartość prawdopodobieństwa

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

- C:/Users/olav/ClionProjects/Dissertation/Crystal.h
- C:/Users/olav/ClionProjects/Dissertation/Crystal.cpp

3.2 Electron Class Reference

`#include <Electron.h>`

Public Member Functions

- Electron (std::vector< double > pos)
- void setX (double x)
- void setY (double y)
- void setZ (double z)

Private Attributes

- std::vector< double > position

  wektor współrzędnych elektronu

Friends

- std::ostream & **operator**<< (std::ostream &s, const Electron &v)

3.2.1 Detailed Description

Klasa reprezentująca elektron

3.2.2 Constructor & Destructor Documentation

3.2.2.1 Electron()

```
Electron::Electron (
            std::vector< double > pos )
```

Konstruktor tworzy obiekt o podanych wspolrzednych

Parameters

pos	wektor współrzędnych

3.2.3 Member Function Documentation

3.2.3.1 setX()

```
void Electron::setX (
            double x )
```

Zmianna x-owej wartości współrzędnej

Parameters

x	nowa wartość współrzędnej

3.2.3.2 setY()

```
void Electron::setY (
            double y )
```

Zmianna y-owej wartości współrzędnej

Parameters

y	nowa wartość współrzędnej

3.2.3.3 setZ()

```
void Electron::setZ (
            double z )
```

Zmianna z-owej wartości współrzędnej

Parameters

z	nowa wartość współrzędnej

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

- C:/Users/olav/ClionProjects/Dissertation/Electron.h
- C:/Users/olav/ClionProjects/Dissertation/Electron.cpp

3.3 ElectronHole Class Reference

```
#include <ElectronHole.h>
```

Public Member Functions

- ElectronHole (std::vector< double > pos, Trap &trap)
- double getEnergy () const
- double getX () const
- double getY () const
- double getZ () const
- Trap * getTrap ()
- void nullTrap ()

  usuwa dziurę z pułapki, ustawia wskaznik trap na NULL
- ~ElectronHole ()

  destruktor

Private Attributes

- std::vector< double > position

  wektor współrzędnych dziury
- Trap * trap = NULL

  wskaźnika na obiekt typu Trap (informacja czy obiekt znajduje się w pułapce)
- double energy = 1.

  energia dziury [w eV]

Friends

- std::ostream & **operator**<< (std::ostream &s, const ElectronHole &v)

3.3.1 Detailed Description

Klasa reprezentująca dziurę elektronową

3.3.2 Constructor & Destructor Documentation

3.3.2.1 ElectronHole()

```
ElectronHole::ElectronHole (
            std::vector< double > pos,
            Trap & trap )
```

Konstruktor tworzy obiekt o podanych wspolrzednych i łączy go z pułapką

Parameters

pos	wektor współrzędnych
trap	referencja do pułapku

3.3.3 Member Function Documentation

3.3.3.1 getEnergy()

```
double ElectronHole::getEnergy ( ) const
```

Returns

zwraca energię dziury

3.3.3.2 getTrap()

```
Trap* ElectronHole::getTrap ( )
```

Returns

zwraca adres do pułapki w której się obecne znajduje

3.3.3.3 getX()

```
double ElectronHole::getX ( ) const
```

Returns

zwraca x-ową współrzędną

3.3.3.4 getY()

```
double ElectronHole::getY ( ) const
```

Returns

zwraca y-ową współrzędną

3.3.3.5 getZ()

```
double ElectronHole::getZ ( ) const
```

Returns

zwraca z-ową współrzędną

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

- C:/Users/olav/ClionProjects/Dissertation/ElectronHole.h
- C:/Users/olav/ClionProjects/Dissertation/ElectronHole.cpp

3.4 Trap Class Reference

`#include <Trap.h>`

Public Member Functions

- Trap (std::vector< double > position)
- double getEnergy () const
- double getX () const
- double getY () const
- double getZ () const
- void setElectron (Electron *electron1)
- Electron * getElectron () const
- void removeElectron (std::vector< double > position)
- bool isOccupied () const

Private Attributes

- std::vector< double > position

  wektor współrzędnych pułapki
- Electron * electron = NULL

  wskaznik na uwięziony elektron
- double energy = 2.

  energia pułapku [w eV]

Friends

- std::ostream & **operator**<< (std::ostream &s, const Trap &v)

3.4.1 Detailed Description

Klasa reprezentująca pułapkę

3.4.2 Constructor & Destructor Documentation

3.4.2.1 Trap()

```
Trap::Trap (
            std::vector< double > position )
```

Konstruktor tworzy obiekt o podanych wspolrzednych

Parameters

position	wektor współrzędnych

3.4.3 Member Function Documentation

3.4.3.1 getElectron()

```
Electron * Trap::getElectron ( ) const
```

Returns

wskaźnik na elektron znajdujący się w pułapce

3.4.3.2 getEnergy()

```
double Trap::getEnergy ( ) const
```

Returns

zwraca energię pułapku

3.4.3.3 getX()

```
double Trap::getX ( ) const
```

Returns

zwraca x-ową współrzędną

3.4.3.4 getY()

```
double Trap::getY ( ) const
```

Returns

zwraca y-ową współrzędną

3.4.3.5 getZ()

```
double Trap::getZ ( ) const
```

Returns

zwraca z-ową współrzędną

3.4.3.6 isOccupied()

```
bool Trap::isOccupied ( ) const
```

sprawdza czy w pułapce znajduje się elektron

Returns

TRUE jesli elektron jest spułapkowany

3.4.3.7 removeElectron()

```
void Trap::removeElectron (
            std::vector< double > position )
```

Usuwa elektorn z pułapki

Parameters

position	nowa pozycja elektronu

3.4.3.8 setElectron()

```
void Trap::setElectron (
            Electron * electron1 )
```

pułapkuje elektron, ustawia wskaźnik na niego

Parameters

electron1	elektron do spułapkowania

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

- C:/Users/olav/ClionProjects/Dissertation/Trap.h
- C:/Users/olav/ClionProjects/Dissertation/Trap.cpp

Index