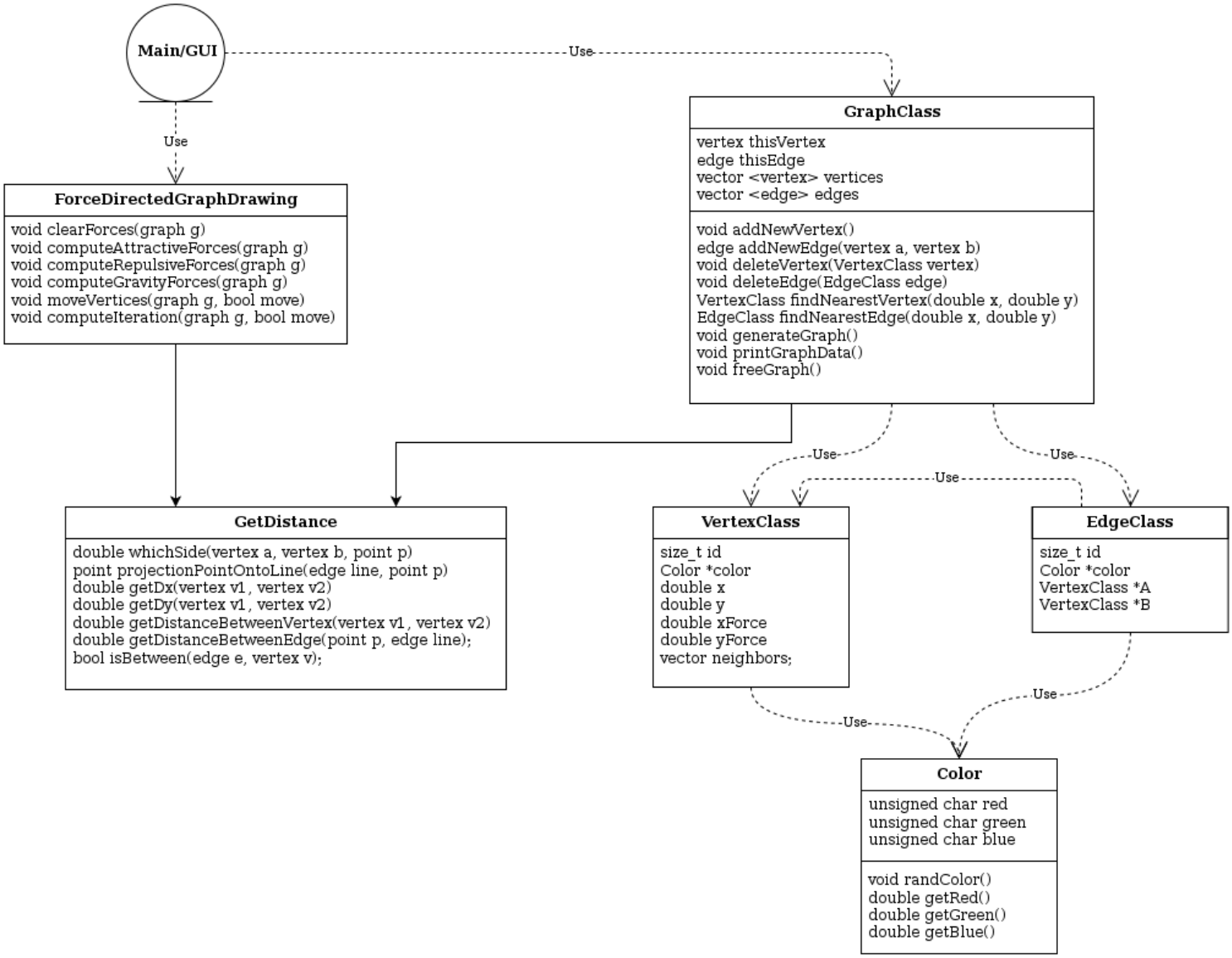
Rysowanie grafów instrukcja programisty

Diagram: UML:



1. Main

Tworzenie okna aplikacji oraz obsługa zdarzeń - główna pętla programu.

Funkcje:

- `static void activate ()`-ustawia parametry aplikacji, tworzy okno, ustawia przyciski i kontrolki oraz przekierowuje sygnały do odpowiednich funkcji
- `static void close_window (void)` -obsługa zamknięcia okna
- `static gboolean motionNotifyEvent (GtkWidget *widget, GdkEventMotion *event, gpointer user_data)` -obsługa ruchu myszy
- `static gboolean buttonReleaseEvent (GtkWidget *widget, GdkEventButton *event, gpointer user_data)` -obsługa zwolnienia przycisku myszy
- `static gboolean buttonPressEvent (GtkWidget *widget, GdkEventButton *event, gpointer user_data)` -obsługa naciśnięcia przycisku myszy
- `static gboolean drawLayers (GtkWidget *widget, cairo_t *cr, gpointer data)` -rysuje warstwy na ekran
- `static gboolean configureLayers (GtkWidget *widget, GdkEventConfigure *event, gpointer data)` -ustawia parametry warstw
- `void buttonClicked(GtkWidget *button, gpointer widget)` -obsługa naciśnięcia przycisku w aplikacji
- `void dataWidgetChanged(GtkWidget *widget, gpointer object)` -obsługa zmiany wartości w kontrolce
- `GtkWidget* createTextBox(const char *name, GtkWidget *boxMenu, const char *str, const char *oName)` -tworzy pole tekstowe oraz obramowanie które jest umieszczane w podanej grupie po czym zwraca wskaźnik do pola tekstowego
- `static void drawVertex (GtkWidget *widget, VertexClass* v )` -rysuje wierzchołki
- `static void drawEdge (GtkWidget *widget, EdgeClass *e)` -rysuje krawędzie
- `void drawGraph(GraphClass *g)` -rysuje graf
- `void computeGraphPos(GraphClass *g)` -oblicza nową pozycję wierzchołków
- `void preDrawGraph(GtkButton *button, GtkWidget* widget)` -przygotowuje plansze do narysowania grafu, opcjinalnie wywołuje funkcję obliczajacą kolejne współrzędne wierzchołków po czym rysuje graf
- `static void clearSurface()` -czyści obszar rysowania

2. Klasa GraphClass

Przechowuje strukturę grafu. Klasa ta z powodzeniem może być użyta w innych projektach do reprezentowania dowolnego grafu nieskierowanego. Zawiera w sobie reprezentację zarówno opartą o listy sąsiedztwa jak i o listę krawędzi.

Każdy graf składa się z:

- vector'a wskaźników na wszystkie wierzchołki
- vector'a wskaźników na wszystkie krawędzie
- wskaźnika na aktualnie zaznaczony wierzchołek
- wskaźnika na aktualnie zaznaczoną krawędź
- informacji o rozmierze okna aplikacji

Nowy obiekt typu GraphClass tworzymy przy pomocy konstruktora:

```
GraphClass(int *wysOkna, int *szerOkna, int *liczbaWierz, int* licznaKraw )
```

Funkcje:

- `void addNewVertex(double x = 0, double y = 0)`–dodaje nowy wierzchołek do grafu i przypisuje mu współrzędne
- `void addNewEdge(VertexClass *a, VertexClass *b)`–dodaje nową krawędź pomiędzy wierzchołkami a i b
- `void deleteVertex(VertexClass *vertex)`–usuwa dany wierzchołek z listy
- `void deleteEdge(EdgeClass *edge)`–usuwa daną krawędź z listy
- `void generateGraph()`-generuje graf z randomowymi pozycjami wierzchołków (liczba krawedzi i wierzchołków jest ustalana przy tworzniu obiektu)
- `void printGraphData()`-wypisuje graf do terminala
- `VertexClass* findNearestVertex(double x, double y)`–zwraca wierzcholek, który znajduje się najbliżej punktu x,y
- `EdgeClass* findNearestEdge(double x, double y)`–zwraca krawędź, która znajduje się najbliżej punktu x,y
- `void freeGraph()`-usuwa wszystkie wierzchołki i krawędzie oraz zwalnia pamięć (w obecnej wersji nie jest używana ponieważ korzystam z shared_ptr)

3. Klasa VertexClass

Przechowuje strukturę wierzchołka grafu.

Każdy wierzchołek ma przypisane:

- id
- pozycję x,y
- wypadkowy wektor sił
- vector wskaźników na wszystkie wierzchołki, z którymi ma bezpośrednie połączenie(w obecnej wersji aplikacji ta informacja nie jest wykorzystywana)
- kolor

Najważniejsze informacje można przypisać do wierzchołka podczas tworzenia obiektu za pomocą konstruktora.

```
VertexClass(double x = 0, double y = 0, int id = -1)
```

4. Klasa EdgeClass

Przechowuje strukturę krawędzi grafu.

Każda krawędź ma przypisane:

- id
- wskaźniki na wierzchołki, króre łączy
- wagę(w obecnej wersji aplikacji ta informacja nie jest wykorzystywana)
- kolor

Najważniejsze informacje można przypisać do krawędzi podczas tworzenia obiektu za pomocą konstruktora

```
EdgeClass(VertexClass *a = NULL, VertexClass *b = NULL, int w = 0, int i = -1)
```

5. Klasa ForceDirectedGraphDrawing

Klasa ta jest odpowiedzialna za obliczenie pozycji każdego wierzchołka na podstawie „sił", które działają między wierzchołkami. Wypadkowy wektor przesunięcia powstaje poprzez zestawienie trzech sił: przyciagania(krawędź przyciąga połączone wierzchołki), odpychania(każdy wierzchołek odpycha wszystkie inne), grawitacji(wszystkie wierzchołki są przyciagane do środka planszy).

Aby obliczenia wykonywały się poprawnie przy tworzeniu obiektu należy podać dane:

- wskaźnik na wysokość i szerokość okna
- wskaźnik na współczynnik siły grawitacji
- wskaźnik na współczynnik siły odpychania
- wskaźnik na współczynnik siły przyciągania
- wskaźnik na wspólczynnik prędkości przesunięcia
- wskaźnik na maksymalną długość, o która można przesunąć wierzchołek

Funkcje:

- *`void clearForces(GraphClass *g)`-ustawia wartości sił na zero
- *`void computeAttractiveForces(GraphClass *g)`-oblicza siłę przyciągania
- *`void computeRepulsiveForces(GraphClass *g)`-oblicza siłę odpychania
- *`void computeGravityForces(GraphClass *g)`-oblicza siłę grawitacji
- *`void moveVertices(GraphClass *g, bool moveThisVertex)`–zmienia pozycje wierzchołków
- `void computeIteration(GraphClass *g, bool moveThisVertex)`–oblicza wektor siły i przesuwa wierzchołki, funkcja przyjmuje parametr dzięki któremu możliwa jest zmiana tylko niezaznaczonych wierzcholków

* - funkcja prywatna

6. Klasa GetDistance

Jej głównym zadaniem jest oblicznie odległości pomiędzy danymi obiektami.

Funkcje:

- `MDtype getDx(VertexClass *v1, VertexClass *v2)`–zwraca róznicę pomiędzy

współrzędnymi x
- `MDtype getDy(VertexClass *v1, VertexClass *v2)`–zwraca róznicę pomiędzy współrzędnymi y
- `MDtype getDistanceBetweenVertices(VertexClass *v1, VertexClass *v2)` –zwraca odległość pomiędzy wierzchołkami
- `MDtype getDistanceBetweenEdges(VertexClass *p, EdgeClass *line)`–zwraca odległość pomiędzy punktem a krawędzią
- `bool isBetween(EdgeClass *edge, VertexClass *v)`–sprawdza czy wierzchołek jest pomiędzy punktami w krawędzi
- `MDtype whichSide(VertexClass *A, VertexClass *B, VertexClass *p)`–zwraca liczbę na podstawie której możliwe jest odczytanie po której stronie krawędzi znajduje się punkt
- `VertexClass* projectionPointOntoLine(EdgeClass *line, VertexClass *p)`– zwraca punkt, który jest rzutem punktu *p* na prosta *line*

*`MDtype` = double

7. Klasa Color

Służy do reprezentacji kolorów RGB, każda składowa koloru jest przechowywana w zmiennej typu char.
Konstruktor pozwala na przypisanie koloru przy deklaracji obiektu:

```
Color(char r, char g, char b)
```

lub nadania losowego koloru:

```
Color()
```

Funkcje:

- `void randColor()` - generuje losowy kolor
- `double getRed() / double getGreen() / double getBlue()`-zwarca składową koloru zrzutowaną na typ double (w GTK składowe koloru reprezentowane są przez liczby rzeczywiste z przedziału [0, 1]).