Układanki z lat dziecinnych

Projekt zaliczeniowy nr **13** z przedmiotu **Zaawansowane C++** (wiosna 2018)

Diagram klas

Realizacja projektu

Zgodnie z założeniami przedstawionymi we wstępnej analizie funkcjonalności aplikacji zostanie ona zrealizowana wg założeń wzorca architektonicznego MVC – zatem struktura aplikacji jest podzielona na 3 osobne części: Model, GUI (view) i Kontroler (controler). Klasy należące do każdej z tych części zostaną przedstawione na osobnym diagramie – na końcu tego dokumentu.

Główna funkcja aplikacji (main)

Funkcja main będzie bardzo prosta – zostanąw niej utworzone obiekty Kontrolera, GUI i Modelu z domyślnymi parametrami, następnie ustawione zostanąw każdym z tych obiektów wskaźniki na pozostałe (by zapewnić komunikację).

GUI

Klasa CPlansza

Klasa dziedzicząca z klasy QFrame będąca kompozytem zwykłej ramki (QFrame) z dodanym layoutem siatkowym (QGridLayout) do rozmieszczania przycisków realizujących pola układanki. Głównym atrybutem klasy jest listaPol – wektor wskaźników na przyciski QPushButton umieszczane w kolejnych polach siatki layoutu. Dodatkowo klasa przechowuje rozmiar (liczba pól w pionie i w poziomie), wymiary przycisku (długość jego krawędzi – przyciski są kwadratowe) oraz wskaźnik na kontroler
Metody:

- zamienPola – zamienia miejscami przyciski (elementy układanki) o podanych lokalizacjach – lokalizacje podane są w postaci łańcucha par liczb (para liczb = pozycje pól do zamiany, elementy znajdujące się w tych miejscach są zamieniane miejscami) Zwraca true jeśli operacja się powiodła
- ustawKolejnosc – pola(przyciski) przenumerowywane są zgodnie z podanym argumentem (wektor liczb, musi być odpowiedniej długości – odpowiadającej rozmiarowi planszy (N*M)). Zwraca true jeśli operacja się powiodła
- on_actionPrzycisk_triggered() – metoda obsługująca wciśnięcie przycisku – powoduje wysłanie wiadomości do kontrolera

Klasa COknoGlowne

Klasa definiująca główne okno aplikacji
Większość kontrolek zostanie zdefiniowana w pliku ui za pomocą graficznego edytora Qt Creator.
Wyjątkiem jest klasa CPlansza – obiekt (QWidget) tej klasy zostanie utworzony i sparametryzowany w kodzie konstruktora – jego wygląd zależy od parametrów ustawionych przez użytkownika
Atrybuty klasy to głównie wskaźniki na kontrolki, których stan jest kontrolowany przez aplikację i może się zmieniać. Są to: plansza i statusBar. Atrybut stan określa stan GUI za pomocą odpowiednio zdefiniowanego typu wyliczeniowego (EGUIstan). Dodatkowo klasa przechowuje wskaźnik na obiekt kontrolera (a dokładnie na interfejs kontrolera przeznaczony dla GUI) – jest on wykorzystywany do wywoływania metody przekazującej kontrolerowi akcje użytkownika.
Przez konstruktor klasy przekazywane są: wskaźnik na kontroler, wektor opisujący stan układanki (nr elementów czytane kolejno wierszami od lewego górnego rogu) oraz jej rozmiar (przyjęto że układanka jest prostokątem o N na M elementach).
Metody zdefiniowane bezpośrednio w tej klasie to głównie metody obsługujące zdarzenia powstałe w wyniku interakcji użytkownika, takie jak: wybranie z menu opcji Konfiguracja, Wyjscie, Tasuj, Rozwiąż, Resetuj, ustawienie opcji Ręczne Ustawienie, wybranie opcji Instrukcja czy O programie (wszystkie metody z prefiksem on_action na diagramie)
Metody te polegają na wywołaniu metody wysyłającej odpowiedni obiekt wiadomości (z hierarchii klas dziedziczących z CGUIMsg) w interfejsie IKontroler

Interfejs IGUI

Metody zadeklarowane w tym interfejsie są metodami udostępnionymi kontrolerowi:

- rysujPlansze – powoduje zresetowanie obiektu klasy CPlansza i narysowanie nowej planszy o podanych w argumentach wymiarach i z podanym w postaci wektora liczb układem elementów. Zwraca true jeśli operacja się powiodła
- zamienPola – zamienia parami pola o numerach podanych w postaci wektora par liczb całkowitych. Zwraca true jeśli operacja się powiodła
- wyswietlKonfiguracje – powoduje wyświetlenie okna klasy COknoKonf z parametrami podanymi w postaci mapy (nazwy i odpowiadające im wartości – oba typu string). Zwraca true jeśli operacja się powiodła

- wyswietlStatus – umieszcza w StatusBar napis podany jako argument. Zwraca true jeśli operacja się powiodła
- zamknijOkno – zamyka okno
- wyswietlInfoOprogramie – powoduje wyświetlenie okna klasy COknoOprogramie z informacjami o programie podanymi w postaci mapy (nazwy i odpowiadające im wartości – oba typu string). Zwraca true jeśli operacja się powiodła
- wyswietlInstrukcje– powoduje wyświetlenie okna klasy COknoInstrukcje z instrukcjami gry podanymi w postaci mapy (nazwy i odpowiadające im wartości – oba typu string). Zwraca true jeśli operacja się powiodła

klasa COknoInstrukcje

klasa dziedzicząca z QDialog. Zawiera kontrolki do wyświetlania tekstu instrukcji oraz przycisk OK. Metoda obsługująca wciśnięcie tego przycisku powoduje zamknięcie tego okna

klasa COknoKonf

klasa dziedzicząca z QDialog. Zawiera kontrolki do wyświetlania/ustawiania parametrów aplikacji oraz przyciski OK. i Anuluj. Metoda obsługująca wciśnięcie przycisku Anuluj powoduje zamknięcie tego okna.
Metoda obsługująca wciśnięcie przycisku OK powoduje zamknięcie tego okna oraz wysłanie do Kontrolera parametrów ustawionych przez użytkownika – za pomocą wywołania metody wyslijWiadomosc z odpowiednio utworzonym obiektem klasy CGUIMsgKonfiguracja (parametry ustawione przez użytkownika są przekazywane do konstruktora).

klasa COknoOprogramie

klasa dziedzicząca z QDialog. Zawiera kontrolki do wyświetlania tekstu O programie oraz przycisk OK. Metoda obsługująca wciśnięcie tego przycisku powoduje zamknięcie tego okna

KONTROLER

Klasa CKontroler

Tutaj zdefiniowana jest hierarchia klas pochodnych abstrakcyjnej klasy CGUIMsg. Obiekty należące do tych klas są używane do przekazywania z GUI do kontrolera akcji wykonanych przez użytkownika. Są przekazywane jako argument metody interfejsu IKontroler::wyslijWiadomosc. Metoda ta zawiera główną instrukcję sterowania – na podstawie typu argumentu uruchamiany jest odpowiedni scenariusz. Scenariusze te są zdefiniowane w prywatnych metodach z prefiksem sc.
Scenariusze to:

- scZamknij – wywoływany gdy z gui przyjdzie wiadomość o wciśnięciu Koniec lub zamknięciu okna. Kontroler usuwa obiekty CUkladanka i CAlgorytm i wywołuje metode Zamknij w gui po czym kończy działanie
- scKonfiguruj – kontroler usuwa obiekt Model i tworzy nowy zgodnie z otrzymanymi parametrami, po czym wywołuje metodę resetuj()w gui – również z otrzymanymi parametrami
- scTasuj – wywołuje metodę modelu generującą losowy układ planszy (zapomniałem dodać tę metodę na diagramie klas), pobiera aktualny układ z modelu metodą getPlansza, wywołuje metodę resetującą w gui z podaniem aktualnego układu planszy
- scRozwiazuj – pobiera aktualny układ z Modelu (getPlansza), tworzy obiekt CAlgorytmAstar podając aktualny stan planszy, w pętli wywołuje metodę IAlgorytm::wykonajKrokAlgorytmu, pętla przerywana jest gdy metoda zwróci true, po wyjściu z pętli wywołuje metodę IAlgorytm::zwrocSciezke, w wyniku otrzymuje listę kolejnych węzłów od początkowego (identyczny z wyświetlanym przez gui stanem planszy) do końcowego (rozwiązanie). Przegląda w pętli te węzły, wywołuje metodę getRuch i wynik jej przekazuje do GUI za pomocą metody wykonajRuch, po każdym kroku odczekuje odpowiedni czas, by animacja prowadząca do rozwiązania była czytelna. *Metoda ta powinna być uruchomiona w osobnym wątku, by nie powodowała blokowania gui. Dodatkowo GUI powinien móc wysłać wiadomość o akcji użytkownika przerywającej działanie algorytmu przeszukiwania – wtedy kontroler powinien przerwać działanie pętli i usunąć obiekt algorytmu (dopisane po przygotowaniu diagramów – nie uwzględniają tych uwag)*
- scResetuj – wywołuje metodę resetującą model i resetującą gui oraz usuwającą obiekt algorytmu
- scReczneUstawienie – ustawia flagę oznaczającą ręczne ustawianie planszy przez użytkownika – kliknięcia w pola nie są zliczane, czas nie jest mierzony
- scWykonajRuch – w wiadomości z gui przekazany jest nr pola, w które kliknął użytkownik. Kontroler sprawdza, czy ruch jest prawidłowy (IUkladanka::sprawdzRuch), jeśli tak, wykonuje ruch IUkladanka::wykonajRuch, pobiera stan planszy (IUkladanka::getPlansza) i przekazuje to do GUI. Jeśli ruch nie był poprawny – ignoruje.
  Jeśli flaga oznaczająca ręczne ustawianie nie jest włączona, a zegar nie został uruchomiony – uruchamia zegar i resetuje licznik ruchów (przypisuje 1). Jeśli flaga nie jest włączona, a zegar został zainicjowany – dodaje do liczby ruchów 1.

Jeśli ruch został wykonany (był prawidłowy) uruchamia metodę IUkladanka::czyUlozona – jeśli zwróciła true – wyświetla komunikat o ułożeniu, podaje łączny czas i liczbę ruchów

Pozostałe metody do get I set wskaźników na obiekty gui, model I algorytm

Interfejs IKontroler

Zawiera jedyną metodę – wyslijWiadomosc

Jest to główna metoda sterująca, uruchamiająca na podstawie typu argumentu – odpowiedni scenariusz

MODEL

Klasa CModelPlansza

Jest to pomysł na szablon klasy dziedziczący z szablonu klasy vector<T> z biblioteki STL. Od klasy nadrzędnej różnić się ma tylko pomysłem na „podwójny" iterator umożliwiający przemieszczanie się po planszy zgodnie z regułami układanki (jedne do ruchu w pione, drugi w poziomie). Ruch w poziomie to dodawanie/odejmowanie 1 pozycji do pozycji domyślnego iteratora z klasy vector – przy czym modulo N (aby dojście do końca wiersza i próba przejścia dalej powodowała pojawienie się na początku tego wiersza). Ruch w pionie to przesunięcie domyślnego iteratora o +/- N pozycji (modulo N*M, by nie wyjść poza planszę). Podobnie działałby podwójny operator [][].

Podstawowym parametrem szablonu klasy byłby typ T – typ przechowywany przez ten kontener (wektor). Dodatkowym parametrem byłaby liczba całkowita N – opisująca liczbę „kolumn" – wymiar poziomy. Wymiar pionowy wynikałby z rozmiaru całego wektora. Klasa posiadałaby metodę zwracającą wartość N.

Jako że nie mam doświadczenia z definiowaniem takich konstrukcji – nie wiem, czy powyższy opis jest realizowalny/prawidłowy. Jeśli nie, to zostanie stworzona „zwykła" klasa z wykorzystaniem specjalizacji vector<int> i odpowiednimi metodami realizującymi powyższe pomysły.

Klasa CModelUkładanka

Klasa reprezentuje układankę – atrybuty to aktualny stan układanki (zapamiętany w obiekcie typu CModelPlansza) oraz wymiary planszy.

Wszystkie metody umożliwiające interakcję z układanką umieszczone są w interfejscie IUkladanka

Interfejs IUkladanka

Interfejs zawiera dwie grupy metod – statyczne, które do uruchomienia wymagają za każdym razem podania stanu układanki, oraz metody pobierające aktualny stan układanki z instancji klasy.

Metody statyczne umożliwiają analizę problemu przez algorytm bez zmiany stanu aktualnego modelu układanki:

- sprawdzRuch – sprawdza czy ruch podany jako pozycja pola z którego ma być przesunięty element w podanym stanie (układzie) układanki jest wykonywalny – jeśli tak metoda zwraca true
- wykonajRuch – wykonuje ruch podany jako pozycja pola z którego ma być przesunięty element w podanym stanie (układzie) układanki. Jeśli ruch nie jest wykonywalny, zwraca pusty wynik, jeśli jest – wykonuje ruch i zwraca stan (układ) po jego wykonaniu
- czyUlozona – zwraca true, jeśli stan podany jako argument to stan ułożenia (wszystkie elementy na swoich pozycjach)
- podajOcene – zwraca wartość funkcji heurystyki (na potrzeby algorytmu przeszukiwania) *Ta metoda jest „kandydatem" na przeniesienie do osobnej klasy – heurystyk.*
- zwrocMozliweRuchy – zwraca wektor liczb oznaczających pozycje pól, z których można wykonać ruch w podanym w argumencie układzie/stanie (na potrzeby algorytmu przeszukującego)
- czyPrawidlowyUklad – zwraca true, jeśli podany przez argument układ/stan jest prawidłowy, tzn. jest rozwiązywalny – są to tzw. Permutacje parzyste, czyli permutacje powstałe przez parzystą liczbę zamian sąsiednich liczb/pozycji (więcej tu: http://www.matematyka.wroc.pl/lamiglowki/pietnastka). Metoda ta może być użyta przez generator losowy pozycji startowej – wtedy wystarczy wygenerować losową permutację pól i sprawdzić, czy jest poprawna (zamiast generować losową sekwencję ruchów i ją wykonywać)

Metody niestatyczne to odpowiedniki powyższych – polegają na wywołaniu metody statycznej, gdzie jako stan podawany jest stan przechowywany w obiekcie (sprawdzRuch,wykonajRuch,czyUlozona)

Metody getPlansza i setPlansza to get i set dla atrybutu opisującego stan planszy

W Diagramie brak metody generującej losowy układ elementów: tasuj(int n), gdzie n – to liczba ruchów (długość) losowej sekwencji

Klasa CAlgorytm

Klasa realizuje algorytm przeszukiwania A*. Metody klasy odseparowane są od problemu, co umożliwia wykorzystanie algorytmu do innych problemów – należy wtedy tylko przeprojektować metody i atrybuty klasy CWezel
Atrybuty tej klasy to:

- aktualnyWezel – obiekt klasy CWezel opisujący aktualnie rozpatrywany przez algorytm węzeł drzewa przeszukiwań
- zbiorOPEN – zbiór elementów typu CWezel. Wykorzystana jest klasa set<T, compareT> z biblioteki STL. Na potrzeby prawidłowego porównywania elementów zbioru konieczne jest opracowanie klasy(a właściwie struktury) CWezelCompare z operatorem umożliwiającym porównywanie dwóch obiektów klasy CWezel. Obekty będą porównywane leksykalnie po kolejnych elementach pól „stan", za obiekt mniejszy będzie uznany ten, który na pozycji i wektora stan będzie miał mniejszą liczbę, przy czym na pozycjach 0,...,i-1 liczby były identyczne (gdzie i może być od 0 do N*M-1, gdzie N,M – wymiar planszy)
- zbiorCLOSED – zbiór reprezentowany podobnie jak powyższy
- liczbaOdwiedzonych – liczba odwiedzonych węzłów w trakcie przeszukiwania przestrzeni stanów (na potrzeby statystyki)

Zbiory OPEN I CLOSED oraz działąnie algorytmu opisałem dokładnie we wstępnym opisie funkcjonalności

Interfejs IAlgorytm

Metody klasy Algorytmu udostępnione kontrolerowi:

- wykonajKrokAlgorytmu – wykonuje jeden krok algorytmu (przejrzenie następników aktualnego węzła i wybór najlepszego pod względem oceny). Na koniec sprawdzany jest stan układanki – jeśli jest ułożona, zwracany jest true, w p.p. - false
- zwrocSciezke – zwraca wektor węzłów (obiektów CWezel) od aktualnego do startowego (a więc ścieżkę w drzewie poszukiwań prowadzącą od aktualnej pozycji do korzenia) – metoda uruchamiana po znalezieniu rozwiązania w celu ustalenia sekwencji ruchów do wykonania na planszy
- getLiczbaOdwiedzonych – zwraca liczbę odwiedzonych dotychczas węzłów (statystyki na koniec działania algorytmu)

Klasa CWezel

Klasa opisująca konkretny stan problem przeszukiwania (czyli konkretny układ elementów układanki) – czyli węzeł w drzewie przeszukiwań algorytmu A*
Klasa ma następujące atrybuty:

- stan – przechowuje układ elementów układanki
- ocena – wartość funkcji heurystyki zapamiętana dla tego stanu
- koszt – zapamiętany koszt dotarcia do tego węzła (liczba ruchów od węzła początkowego, czyli głębokość w drzewie przeszukiwania)
- poprzednik – wskaźnik na węzeł z którego tu przyszliśmy
- ruch – pozycja pola, z którego należy przesunąć element (wykonać ruch) w układzie opisanym przez węzeł zapamiętany w Poprzedniku

Metody:

- getOcena – zwraca zapamiętaną ocenę
- szacujOdleglosc – zwraca wartość funkcji heurystyki, czyli szacowaną odległość do rozwiązania – wyliczana za pomocą statycznej metody z interfejsu IUkladanka
- getKoszt – zwraca zapamiętany koszt dotarcia do tego węzła od węzła początkowego
- getStan – zwraca stan, czyli układ pól układanki
- aktualizujOceneKoszt – aktualizuje koszt i ocenę (koszt+heurystyka) dla tego węzła – metoda wykonywana, gdy do węzła dotarliśmy pierwszy raz, lub gdy zapamiętana ocena była gorsza niż obecnie znaleziona. Metoda ta ustawia również poprzednika i wykonany ruch
- getPoprzednik – zwraca węzeł będący poprzednikiem
- getNastepniki – zwraca wektor węzłów do których można dotrzeć z tego węzła
- getRuch – zwraca ruch, czyli nr pola z którego należy przesunąć element w stanie układanki opisanym przez poprzednik (potrzebne do zmiany ścieżki z drzewa przeszukiwań prowadzącej od węzła początkowego do węzła końcowego na sekwencję ruchów)

Diagram klas GUI

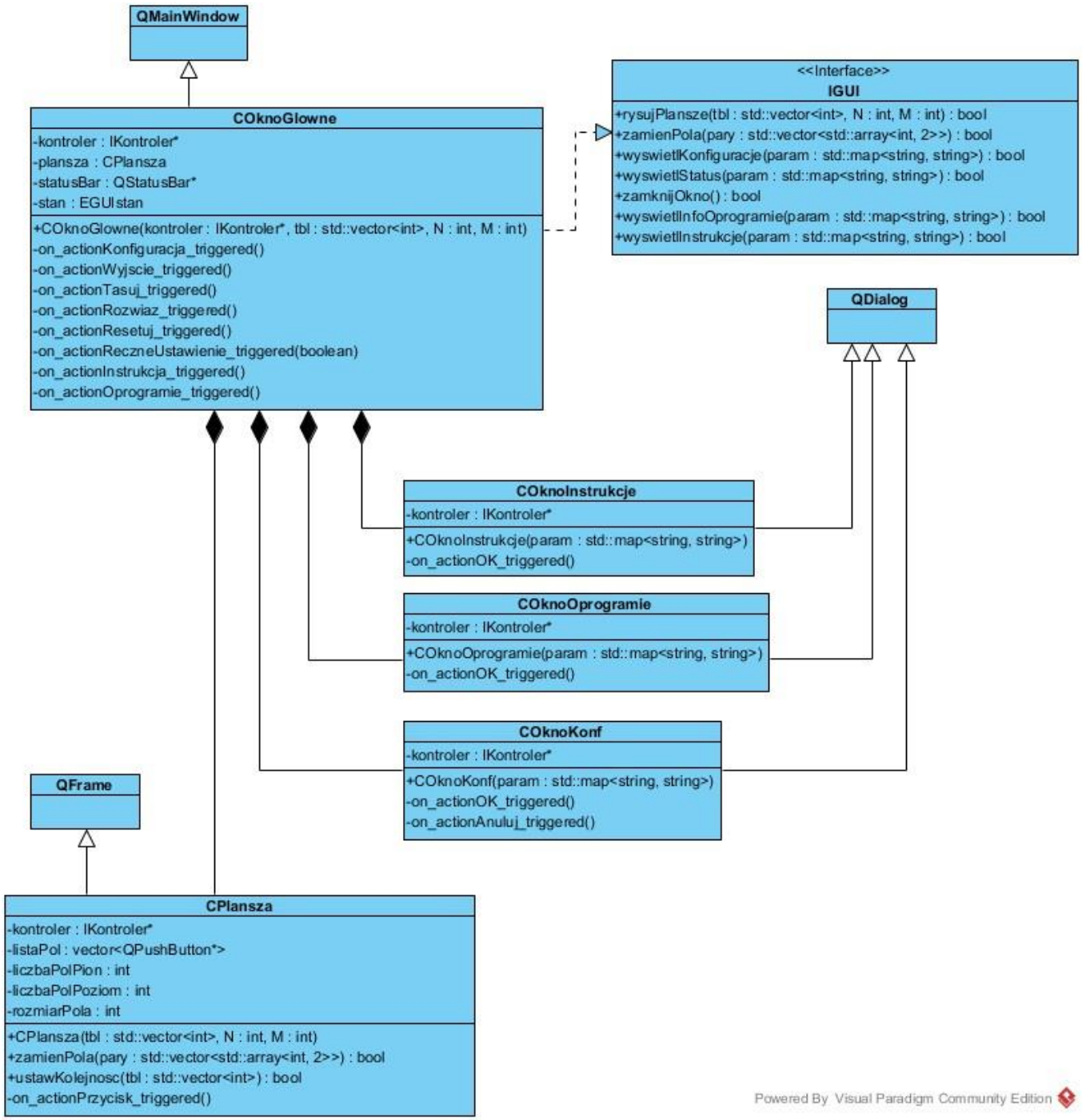

Diagram klas Kontrolera

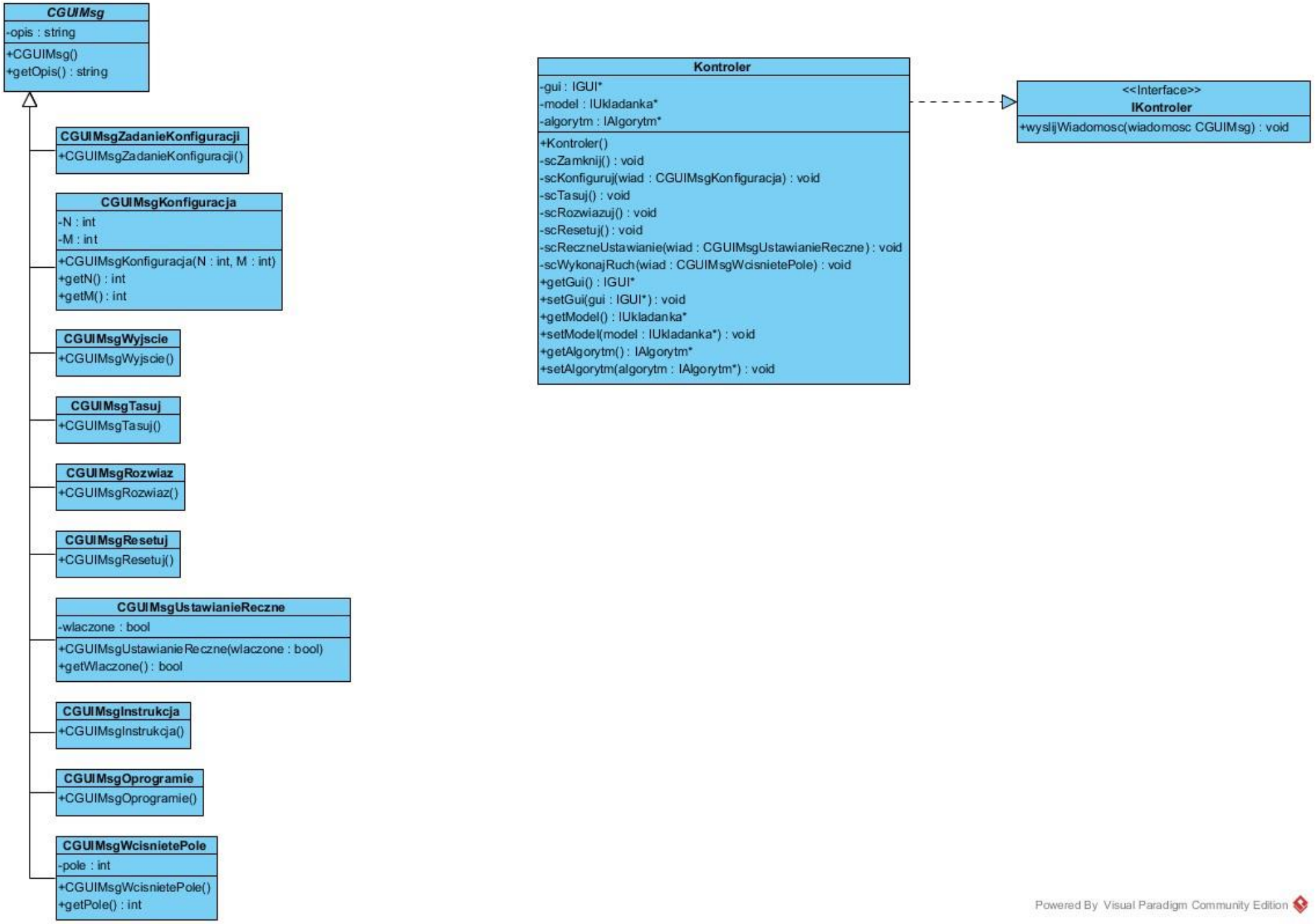

Diagram klas Modelu