Interpreter opisu działań obiektów – Etap 3.

1 Zakres prac

W ramach pracy nad programem należy zrealizować następujące podzadania:

- Wczytywanie pliku opisu działań, w którym zawierających sekwencję poleceń będzie podział na zbiory równolegle wykonywanych poleceń.
- Sekwencyjne wykonywanie zbioru równoległych poleceń i wizualizacja ich efektu z wykorzystaniem dostarczonego serwera graficznego.

2 Składnia pliku poleceń

Składnia pliku poleceń w wersji podstawowej jest zgodna z tą przedstawioną w ogólnym opisie zadania w rodziale 5. (na końcu opisu). W celu ułatwienia implementacji metody `ExecCmd` na rys. 1 pokazano diagram czynności z wyszczególnieniem blokowania elementu sceny i kanału komunikacyjnego. Blokowanie dostępu jest konieczne ze względu na równolegle wykonywane inne polecenia.

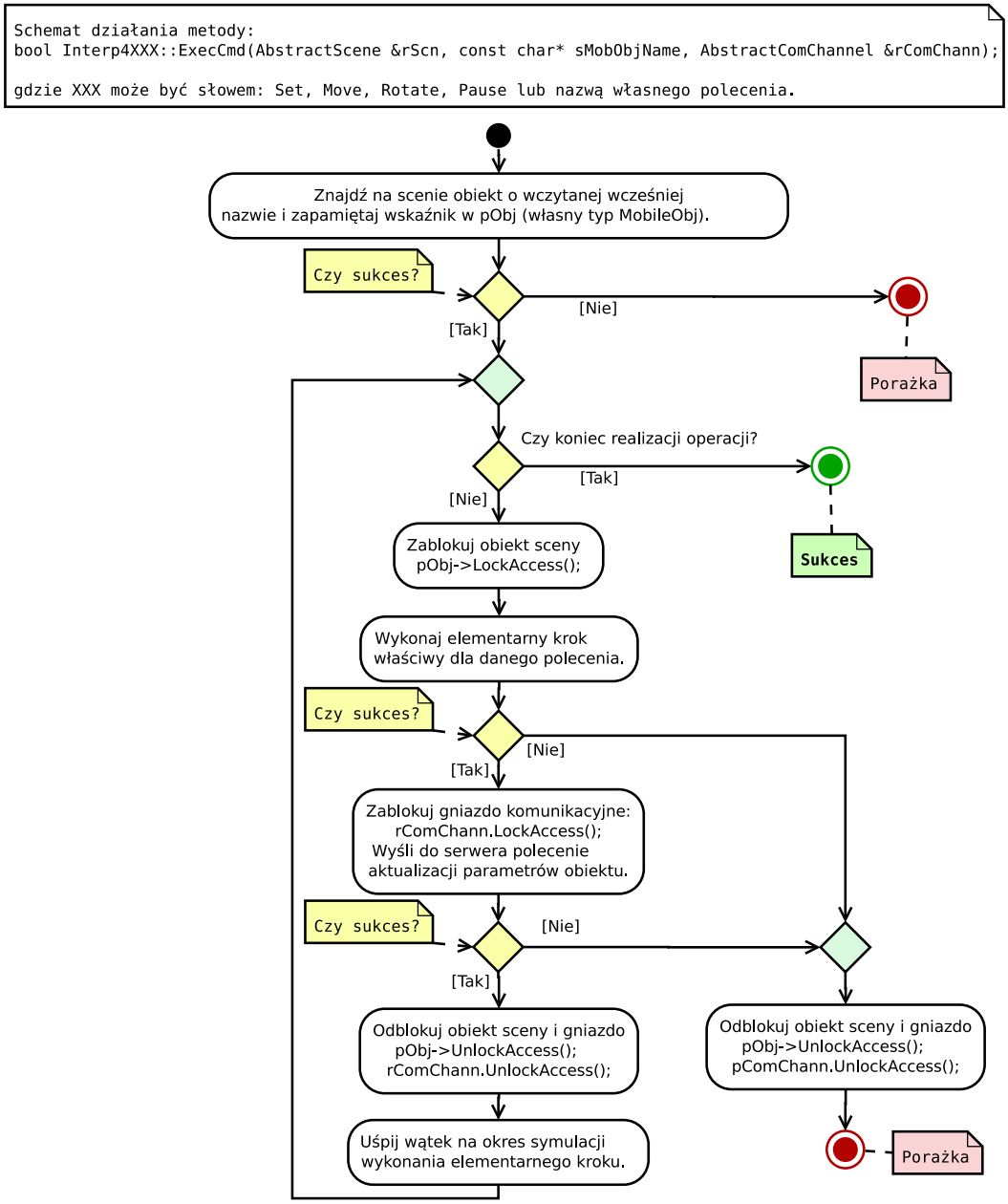
3 Uproszczenia wersji końcowej

Przewidywanych jest kilka wersji uproszczeń. Poziom uproszczeń każdej z wersji wyznacza górną granicę oceny, jaką można uzyskać za realizację interpretera poleceń. W każdej jednak wersji uproszczeń należy wykorzystać wątki w wersji języka C++ oraz obiekty zarządzające dostępem (mutex) do wspólnego zasobu. Dla każdego poniżej przedstawionego wariantu należy także stworzyć pakiet instalacyjny opisany w etapie 4. Ten etap będzie realizowany bezpośrednio na zajęciach i na tych samych zajęciach powinien zostać zakończony.

3.1 Uproszczenie dla górnej granicy oceny: 3,5

Sposób interpretacji poleceń z zewnętrznego pliku może ograniczać się do wariantu zadania z etapu 1. Należy go rozszerzyć do wersji, która jednocześnie interpretuje i wykonuje polecenia z dwóch plików. Wystarczy, że polecenia będą wyświetlane na wyjściu standardowym, tak jak w zadaniu z etapu 1. Wspólnym zasobem w tym przypadku będzie wyjście standardowe. Każdy z wątków powinien wykorzystywać wspólny obiekt typu *mutex*, aby zapewnić sobie wyłączność dostępu do wyjścia standardowego na czas wyświetlania polecenia.

Chcąc mieć możliwość identyfikacji wątków z poziomu, z którego wyświetlane jest dane polecenie, należy również wyświetlić identyfikator wątku. Wspomniany identyfikator odnoszący się do bieżącego wątku można otrzymać poprzez odwołanie się do metody `std::this_thread::get_id()`.



Rysunek 1: Schemat blokowania elementów sceny i kanału komunikacyjnego w metodzie `Interp4Commnad::ExecCmd`

3.2 Uproszczenie dla górnej granicy oceny: 4,0

Możliwe są dwa warianty tego uproszenia:

1. Zasadnicza idea tak jak w wariancie na ocenę 3,5 dodatkowo wykorzystanie serwera graficznego do obrazowania wykonania poleceń. Nie jest wymagana animacja ruchu.

2. Interpretacja poleceń z wykorzystaniem kolekcji obiektów reprezentujących poszczególne biblioteki współdzielone. Kolekcja powinna być zrealizowana z wykorzystaniem szablonu `std::map<>`. Nie jest wymagane użycie serwera graficznego do obrazowania ruchu obiektów.

3.3 Uproszczenie dla górnej granicy oceny: 4,5

W tym uproszczeniu możliwe są również dwa warianty

1. Zasadnicza idea tak jak w wariancie 1. na ocenę 4,0 dodatkowo tworzenie kolekcji obiektów reprezentujących biblioteki współdzielone. Zestaw bibliotek należy wczytać z wykorzystaniem parsera pliku konfiguracyjnego zapisanego w XML.

2. Zasadnicza idea tak jak w wariancie 2. na ocenę 4,0 dodatkowo interpretacja poleceń współbieżnych. Program będzie interpretował tylko jeden plik. Jednak w nim mogą występować polecenia współbieżne, co oznacza konieczność tworzenia dowolnej ilości wątków. W tym wariancie nie jest konieczne korzystanie z serwera graficznego. Polecenia mogą być wyświetlane na wyjściu standardowym jako wspólnym zasobie.