Interpreter opisu działań obiektów

1 Ogólny opis całego zadania

Należy napisać program, który będzie w stanie przeczytać z pliku tekstowego sekwencję poleceń opisujących jednoczesne działanie wielu obiektów. W pliku powinno być możliwe stosowanie wszystkich konstrukcji, które dopuszcza preprocesor języka C. Program powinien móc wykonać wspomniane wcześniej polecenia obrazując przemieszczanie się zbioru obiektów z wykorzystaniem dostarczonej aplikacji graficznej. Wspominana aplikacja będzie pracowała jako serwer. Komunikacja z aplikacją ma być realizowana poprzez mechanizm gniazd. Zakłada się, że z każdym rodzajem polecenia skojarzona jest odpowiednia *wtyczka*, która obsługuje dane polecenie.

Wstępna konfiguracja sceny oraz programu powinna być wczytana z pliku tekstowego w formacie XML. Zakłada się, że we pliku tym będą zapisane następujące informacje:

- informacja o obiektach geometrycznych z przyporządkowaną im nazwą, kolorem, i rozmiarem,
- nazwy wtyczek, które należy załadować,
- adres sieciowy serwera graficznego (aplikacji obrazującej ruch prostopadłościanów) i numer portu.

2 Organizacja pracy

Realizacja pracy podzielona jest na 4 etapy:

Etap 1.
- Stworzenie czterech wtyczek, które będą ładowane przez program w momencie startu.
- Zaprojektowanie i zdefiniowanie struktur danych, które będą pozwalały łączyć nazwę polecenia, z funkcjami interfejsu wtyczki obsługującej dane polecenie.
- Rozwijanie makr preprocesora w czytanym pliku *programu działań*.
- Wywołanie funkcji właściwej wtyczki dla wczytanego polecenia z pliku działań i wczytanie parametrów działania.
- Wyświetlanie parametrów działania.

Etap 2.
- Wczytywanie konfiguracji w XML.
- Łączenie z serwerem graficznym.
- Przesyłanie do serwera poleceń wizualizacji i rysowania.
- Sekwencyjne wykonywanie poleceń.

Etap 3.
- Wczytywanie pliku opisu działań, w którym będzie podział na zbiory równolegle wykonywanych poleceń.
- Sekwencyjne wykonywanie zbioru równoległych poleceń.

Etap 4. Tworzenie pakietu instalacyjnego GNU (zadanie ma być wykonane w całości na zajęciach).

3 Składnia poleceń w pliku opisu działań

Przyjmuje się, że każde z poleceń zapisywane jest w jednej linii. Na początku występuje nazwa plecenia. Może być ona poprzedzona dowolną ilością znaków *białych*. Po niej następuje lista parametrów oddzielonych także dowolną ilością znaków *białych*. Zakłada się, że powinny istnieć co najmniej 4 *wtyczki*, które będą obsługiwały polecenia:

`Set` – przestrzenne ustawienie wybranego obiektu w danej pozycji i orientacji,

`Move` – ruch wybranego obiektu na przód z zadaną prędkością,

`Rotate` – obrót wybranego obiektu wokół jednej z jego trzech osi,

`Pause` – zatrzymanie działania interpretera na określony czas.

W dalszej części przedstawiona jest składnia poszczególnych poleceń oraz przykłady ich użycia.

3.1 Polecenie `Set`

Polecenie to powoduje ustawienie wybranego obiektu w zadanym miejscu na scenie. Obiekt identyfikowany jest poprzez nadaną mu wcześniej nazwę (na etapie wczytywania pliku konfiguracyjnego). Składnia polecenia:

`Set` *nazwa_obiektu* *wsp_x* *wsp_y* *wsp_z* *kat_OX* *kat_OY* *kat_OZ*

Wartość kąta podana jest w stopniach.

3.2 Polecenie `Move`

Polecenie to powoduje ruch wybranego obiektu do przodu z zadaną szybkością horyzontalną. Składnia polecenia:

`Move` *nazwa_obiektu* *szybkość* *długość_drogi*

Szybkość obiektu może być dodatnia (ruch do przodu) lub ujemna (ruch do tyłu). Długość drogi musi być liczbą nieujemną.

3.3 Polecenie `Rotate`

Zmiana orientacji wybranego obiektu poprzez obrót wokół zadanej własnej osi obiektu. Składnia polecenia:

`Rotate` *nazwa_obiektu* *nazwa_osi* *szybkość_kątowa* *kąt_obrotu*

Zakłada się, że jednostkami parametrów są odpowiednio: $[\frac{^\circ}{s}]$ (stopnie na sekundę), $[^\circ]$ (stopnie).

Kąt obrotu (podobnie jak we wcześniejszych przypadkach droga) może być tylko wartością nieujemną. Szybkość kątowa może być wartością ujemną (obrót w prawo) lub wartością dodatnią.

3.4 Polecenie `Pause`

Powoduje zawieszenie działania danego wątku interpretera poleceń na wybrany czas. Składnia polecenia:

`Pause` *Czas_pauzy_ms*

Czas wyrażony jest w milisekundach.

3.5 Przykład użycia poleceń

Poniżej przedstawiona jest przykładowa sekwencja wcześniej opisywanych poleceń. Sekwencja ta może znaleźć się w pliku tekstowym, który będzie musiał zostać wczytany przez program. W pliku tym mogą wystąpić polecenia dla preprocesora języka C, które muszą być poprawnie zinterpretowane.

```
#define ROTATE_SPEED     30
/*
 *  Przykładowy zestaw poleceń
 */
  Set   Ob_A 2 0 3 30 10 0   // Polozenie obiektu A
  Set   Ob_B 10 10 3 0 0 20  // Polozenie obiektu B
  Rotate Ob_B OY ROTATE_SPEED 40
  Pause 1000           /* Zawieszenie działania na 1 sek. */
  Move  Ob_A  10 10
  Rotate Ob_B OZ ROTATE_SPEED 60
  Move  Ob_B 10 20
```

4 Wymagania co do konstrukcji programu

Program realizujący sformułowane wcześniej zadanie ma być napisany w języku ANSI C++20. Nie ma on dostarczać żadnego menu. Program ma być wywoływany z dwoma parametrami. Są nimi nazwy plików. Pierwszy zawiera opis działań obiektów. Drugi zaś ma być plikiem konfiguracyjnym w formacie XML. Składnia wywołania programu ma więc postać:

```
./program  dowolna/sciezka/opis_dzialan.cmd  inna/sciezka/konfiguracja.xml
```

Oba pliki mogą być zlokalizowane w różnych ścieżkach. Oprócz pliku konfiguracyjnego potrzebne jest jeszcze wczytanie pliku zawierającego opis gramatyki pliku `konfiguracja.xml`. Dla ułatwienia zakłada się, że plik gramatyki znajduje się w tej samej lokalizacji jak plik `konfiguracja.xml`, ma ten sam rdzeń nazwy. Różni się jedynie rozszerzeniem. W przypadku pliku gramatyki jest to rozszerzenie `xsd`. Tak więc pełną nazwą tego pliku dla zademonstrowanego wcześniej przypadku będzie `inna/sciezka/konfiguracja.xsd`.

Program po uruchomieniu powinien wykonać następujące operacje:

1. sprawdzić czy liczba parametrów jest poprawna.
2. wczytać plik konfiguracji i zainicjalizować odpowiednie struktury danych,

3. nawiązać połączenie z serwerem graficznym,

4. zacząć czytać opis działania obiektów i wykonywać te działania poprzez wysyłanie do serwera odpowiednich poleceń,

5. po zakończeniu czytania pliku z opisem działań obiektów oraz ich wykonywania, powinien rozłączyć się z serwerem i zakończyć swoje działanie.

Program w trakcie swojego działania powinien wyświetlać bieżące wykonywane polecenia wraz z wartościami ich parametrów.

5 Pełna składnia opisu działań obiektów – wersja podstawowa

Ze względu na to, że w pliku opisu działań obiektów, działania te mogą być wykonywane równolegle, ich opis musi być dzielony na sekcje jednoczesnych akcji. Sekcje te natomiast będą wykonywane sekwencyjne. Przyjmuje się, że działanie danej sekcji kończy się, gdy zakończy się działanie wszystkich poleceń. Sekcja działań jednoczesnych rozpoczyna się słowem `Begin_Parallel_Actions` i kończy słowem `End_Parallel_Actions`. Zakłada się, że słowa i polecenia nie muszą być zapisywane w osobnych liniach. Muszą być jednak rozgraniczone znakami białymi. Przykład zapisu działań:

```
#define ROTATE_SPEED     30
/*
 *  Przykładowy zestaw poleceń
 */
 Begin_Parallel_Actions
  Set   Ob_A 2 0 0 30 0 0   // Polozenie obiektu A
  Set   Ob_B 10 10 5 0 0 0  // Polozenie obiektu B
 End_Parallel_Actions

 Begin_Parallel_Actions
  Rotate Ob_A OX ROTATE_SPEED 40
 End_Parallel_Actions

 Begin_Parallel_Actions  Pause 1000 /* Zawieszenie na 1 sek. */ End_Parallel_Actions

 Begin_Parallel_Actions
  Move  Ob_A  10 10
  Rotate Ob_B ROTATE_SPEED OZ 60 /* Rotate i Move wykonywane razem */
  Move  Ob_B 10 20               /* powoduja jazde po luku         */
 End_Parallel_Actions
```