Laboratorium
Projektowanie Obiektowe – Wzorce Projektowe

Przykładowe projekty

Opiekuni projektu: dr inż. Tomasz Kowalski

Cele i założenia

- Opracowanie i zaimplementowanie dodatkowej funkcjonalności do aplikacji PlotterSoft
- Zadania kładą nacisk na wykorzystanie wzorców projektowych i zasad projektowania w ujęciu obiektowym.

A. Kontrola poziomu tuszu

Prace nad rzeczywistym sterownikiem plotera wykazały, że nie będzie możliwe uzyskanie dostępu do informacji o tuszu bezpośrednio z plotera. Konieczne jest autorskie rozwiązanie umożliwiające kontrolę i zarządzanie poziomem tuszu w ploterze.

Zalecenia opiekuna projektu:

- kontrola tuszu powinna być możliwa zarówno w trybie rzeczywistego użycia plotera jak i podczas symulacji, a abstrakcje korzystające ze sterowników (klienci sterownika) nie muszą być świadome o zaimplementowanym mechanizmie,
- ilość tuszu powinna być mierzona w jednostkach długości rysowanych linii,
- na potrzeby symulacji przy braku tuszu ploter powinien przesuwać swoją pozycję nie rysując do momentu doładowania (alternatywnie mogłaby następować zmiana typu linii),
- mile widziane byłyby powiadomienie ostrzegające o kończącym się poziomie tuszu oraz opcja doładowywania tuszu w krytycznych momentach,
- rysowanie w symulatorze przebiega zdecydowanie zbyt szybko (nie widać nawet kolejności rysowanych linii), warto to poprawić (jako opcje),
- przydatny (np. do testowania) mógłby okazać się mechanizm manualnej obsługi plotera.

B. Zaawansowana fabryka poleceń

Hierarchia poleceń daje duży potencjał do wielokrotnego użycia. Ciężko jednak z góry przewidzieć ograniczony zestaw poleceń złożonych, który należałoby udostępnić użytkownikowi. Konieczne jest autorskie rozwiązanie umożliwiające dynamiczne tworzenie dowolnego zbioru poleceń.

Zalecenia opiekuna projektu:

- ingerencja w hierarchię poleceń jest możliwa, ale powinna być pragmatyczna,
- nowe polecenia do zbioru mogą być dodawane na podstawie istniejących poleceń,
- w przypadku dużego zbioru konieczny jest skuteczny system klasyfikacji i filtracji poleceń,
- aby zbiór poleceń mógł być wielokrotnie wykorzystany przydałby się mechanizm ich serializacji (import i eksport też mile widziany),
- programiści powinni dostać łatwy do zrozumienia szablon (np. w hierarchii poleceń mogą się pojawić różne implementacje *ICompoundCommand*),
- przydatny by był jakiś prosty generator poleceń na bazie rysowanych przez ploter figur. (patrz zadanie 3.1.5 z laboratorium o wzorcu command).

C. Edycja poleceń złożonych

Hierarchia poleceń daje duży potencjał do wielokrotnego użycia. Ciężko jednak zaprojektować polecenie złożone bez odpowiednich narzędzi. Konieczne jest autorskie rozwiązanie umożliwiające efektywne tworzenie poleceń złożonych i ich podstawową obróbkę (zmiana parametrów, kolejności, itp.). Innym problemem, który przy okazji trzeba rozwiązać jest obawa, że polecenia złożone tworzone w oparciu o bieżącą hierarchię poleceń mogą zajmować dużo pamięci.

Zalecenia opiekuna projektu:

- ingerencja w hierarchię poleceń lub jej rozbudowa jest pożądana, ale postarajmy się o wsteczną kompatybilność,
- należy uwzględnić, że polecenie złożone może potencjalnie składać się z innych poleceń złożonych,
- aby polecenie mogło być wielokrotnie wykorzystane przydałby się mechanizm ich importu i eksportu,
- w hierarchii poleceń mogą się pojawić różne implementacje *ICompoundCommand*,
- pożądana jest jak największa integracja z naszą aplikacją, więc do edycji i tworzenia dobrze by było wykorzystać „wbudowaną" w aplikacje przestrzeń do rysowania,
- w symulatorze nie widać nawet kolejności rysowanych linii, warto to jakoś w ramach opcji edytora umożliwić,
- dla użytkownika liczy się przede wszystkim wygoda i intuicyjność w edytowaniu.

D. Przekształcanie poleceń

Niewątpliwą wadą poleceń, które mogą przedstawiać dowolne figury, jest ich statyczność. Nie ma sensu wykorzystywać polecenia złożonego więcej niż jeden raz w ramach tego samego rysunku. Konieczne jest autorskie rozwiązanie umożliwiające tworzenie poleceń poprzez przekształcanie istniejących. Zestaw operacji przekształcających może uwzględniać m.in.: przesuwanie, obracanie, skalowanie, rozciąganie, odbijanie.

Zalecenia opiekuna projektu:

- ingerencja w hierarchię poleceń jest możliwa, ale powinna być minimalna,
- warto uwzględnić, że pewne przekształcenia (lub ich sekwencje) są często wykorzystywane,
- z praktycznego punktu widzenia przydatne mogłyby się okazać informacje na temat przekształconego polecenia, np.:
  - sumaryczna długość narysowanych linii,
  - sumaryczna droga przebyta przez głowicę rysującą plotera,
  - najbardziej wysunięte pozycje w osi X i osi Y,
- programiści powinni dostać łatwy do zrozumienia szablon (np. na poleceniach mogą się pojawić w przyszłości nowe przekształcenia czy obliczenia),
- w hierarchii poleceń mogą się pojawić różne implementacje *ICompoundCommand*,
- przydatny by był jakiś prosty generator poleceń na bazie rysowanych przez ploter figur. (patrz zadanie 3.1.5 z laboratorium o wzorcu command).

E. Zoom IN and OUT

Inny problem aplikacji jest brak możliwości zmiany parametrów wyświetlanej symulacji. Rysunki są czasem zbyt małe, czasem zbyt duże. Oprócz możliwości powiększania lub zmniejszania można uwzględnić np.: przesuwanie, obracanie, rozciąganie, odbijanie.

Zalecenia opiekuna projektu:

- ingerencja w hierarchię poleceń jest możliwa, ale nie zalecana (przekształcenia mogłyby to wpłynąć nie tylko na symulacje),
- przydatny (np. do testowania) mógłby okazać się mechanizm manualnej obsługi plotera.