|  |  |  |  |
| --- | --- | --- | --- |
| **PW OKNO Wydział Elektryczny** | | Przedmiot: | **Grafika Komputerowa i Wizualizacja** |
| Kierunek: | Informatyka stosowana | Zadanie: | Projekt |
| Autor: | **Mirosław Rychel** | Temat: | **Kamera wirtualna** |
| Data: | 10 września 2016 | Etap: 1 | Obraz rzutowany z obcinaniem krawędzi, bez ukrywania elementów zasłoniętych |

**Sprawozdanie**

1. Cel i temat projektu:

Celem projektu było napisanie programu komputerowego, w dowolnej technologii i języku programowania, realizującego tworzenie obrazu poprzez wirtualną kamerę. W pierwszym etapie program nie analizuje i nie ukrywa elementów zasłoniętych.

1. Założenia:
   1. Prosta scena zawierająca kilka prostopadłościanów (analogicznie do widoku budynków w terenie). Elementy sceny nie stykają się i nie przenikają.
   2. Wszystkie warianty położenia kamery powinny generować poprawny obraz.
   3. Program musi uruchamiać się w standardowym systemie, bez instalacji dodatkowych bibliotek.
2. Zastosowane rozwiązania:
   1. Program został napisany w języku Java bez zastosowania dodatkowych bibliotek.
   2. Program działa w systemie Windows, ale język programowania Java umożliwia skompilowanie go dla innych systemów.
   3. Parametry kamery są ustawiane przy pomocy elementów menu.
   4. Zdefiniowane zostały podstawowe ruchy kamery:
      1. Przesunięcia w trzech osiach.
      2. Obroty wokół trzech osi.
      3. Powiększanie obrazu w trzech kierunkach.
      4. Pochylanie obrazu.
      5. Zoom: przybliżanie i oddalanie (realizowane poprzez zmianę odległości kamery od rzutni, przy zachowaniu wymiarów rzutni)
      6. Home: zawsze powrót do pozycji startowej
   5. Definicje rozmiaru okna, obszaru widzialnego, położenia i rozmiarów prostopadłościanów, wielkości skoku translacji i obrotów, odległości kamery od rzutni znajdują się w kodzie programu (klasa Algorytumy.java).
   6. Przyjęto zasadę, że rzutnia ma wielkość zdefiniowanego obszaru widzialnego i znajduje się na płaszczyźnie z = 0 (lewoskrętny układ obserwatora położonego na osi z w punkcie z = -d).
   7. Program nie wyświetla obiektów, które przecinają płaszczyznę rzutni.
3. Opis działania programu:
   1. Utworzenie tablicy dynamicznej na bieżące współrzędne wierzchołków i przypisanie do niej wartości początkowych. Tablica ta określa zawsze wyjściowe, początkowe położenie obiektów. Po każdej zmianie parametrów dane te są przeliczane w celu wyświetlenia aktualnego obrazu.
   2. Utworzenie obrazu startowego na podstawie początkowych parametrów.
   3. Utworzenie obramowania okna oraz menu.
   4. Oczekiwanie na ustawienie wartości parametrów w menu.
   5. Przeliczenie współrzędnych wszystkich wierzchołków przy użyciu:
      1. Dla translacji: dodania lub odjęcia skoku do odpowiedniej współrzędnej
      2. Dla obrotów: odpowiednie mnożenie i normalizacja uprzednio zdefiniowanych macierzy obrotów (transponowanych) z wektorem położenia wierzchołka
      3. Dla zoom: zmniejszenie lub zwiększenie odległości obserwatora od rzutni (dostępny zakres d>0)
      4. Dla „Home” przepisanie danych startowych
   6. Wyczyszczenie okna
   7. Utworzenie nowego obrazu
4. Opis struktury programu:

Klasa Kamera w pakiecie interfejs\_uzytkownika zawiera elementy pakietu Java Swing niezbędne do wyświetlenia okna. Następnie tworzone jest menu oraz instancja podklasy Obiektyw, która zajmuje się narysowaniem obiektów z uzyskanej kolekcji.

Klasa Algorytmy z pakietu o tej samej nazwie dostarcza klasie Kamera trzy kolekcje: parametrów trójwymiarowych, parametrów jednowymiarowych oraz elementów (linii lub płaszczyzn) do narysowania na ekranie. Algorytmy dziedziczą wszystkie klasy z pakietu narzędzia. Są to kolejne stopnie abstrakcji elementów obrazu: Punkt3D, Punkty3D, Odcinek, Odcinki, Plaszczyzna, Plaszczyzny. Są to niewielkie struktury i z punktu widzenia samego programu można ich było nie wydzielać, ale dzięki temu program jest trochę bardziej przejrzysty i jest przygotowany do ewentualnej rozbudowy. Klasa zawiera także realizację algorytmów 4 przekształceń 3D, rzutowania obrazu, przycinania krawędzi wystających poza ekran oraz zasłaniania powierzchni. W zasadzie można było wydzielić same algorytmy od parametrów i struktur przechowujących dane, ale nie poprawiłoby to czytelności programu, bo algorytmy są tylko cztery, za to skomplikowało by komunikację, gdyż parametry, zastosowane algorytmy oraz przetwarzane dane są ze sobą ściśle powiązane.

1. Procedura tworzenia obrazu
   1. Sprawdzenie, czy obiekt nie przecina rzutni, a jeśli przecina, to pominięcie go przy wyświetlaniu
   2. Rysowanie po kolei wszystkich zadeklarowanych wcześniej prostopadłościanów
   3. Rzutowanie wszystkich wierzchołków prostopadłościanu
   4. Sprawdzanie i oznaczenie położenia każdego wierzchołka na rzutni według algorytmu Sutherlanda-Hodgmana
   5. Rysowanie 3 krawędzi wychodzących z jednego wierzchołka, powtarzane 4 razy w celu utworzenia obrazu całego prostopadłościanu
      1. Podczas rysowania każdej krawędzi analizowane jest położenie wierzchołków i ewentualne odpowiednie obcięcie krawędzi jedno lub dwustronne lub jej odrzucenie
      2. Dla odróżnienia w trakcie testów krawędzie ścian przedniej, tylnej oraz boczne łączące (przypisanie nomenklatury na podstawie pozycji startowej) są rysowane przy użyciu różnych kolorów
   6. Program posiada również procedurę rysowania prostopadłościanów bez obcinania (niewykorzystywaną w obecnej wersji programu).
2. Ocena uzyskanego efektu

Zakładana funkcjonalność została zrealizowana poprawnie. Podczas różnorodnych testów wszystkie obrazy były wyświetlane zgodnie z wyobrażeniem. Scenę można oglądać z każdej strony. Obejście jej o kąt pełny daje obraz zgodny z obrazem startowym.