|  |  |  |  |
| --- | --- | --- | --- |
| **PW OKNO WEiTI** | | Przedmiot: | **Grafika Komputerowa** |
| Kierunek: | Techniki Multimedialne | Zadanie: | Projekt |
| Autor: | **Mirosław Rychel** | Temat: | **Kamera wirtualna** |
| Data: | 30/04/2015 | Etap: 2 | Obraz rzutowany z obcinaniem ścian oraz ukrywaniem 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.

W etapie drugim program ukrywa niewidoczne ściany obiektów oraz elementy zasłonięte przez inne obiekty.

W poniższym opisie nowe elementy realizacji etapu II zaznaczono kolorem niebieskim.

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. Punktem wyjścia do realizacji drugiego etapu projektu był program z pierwszego etapu realizujący funkcją kamery wirtualnej bez ukrywania elementów zasłoniętych.
   2. Program został napisany w języku C++ przy użyciu WinApi pakietu Visual C++ 2010 Express.
   3. Program działa w systemach Windows, dołączając statycznie biblioteki potrzebne do jego uruchomienia na dowolnym komputerze.
   4. Przemieszczanie i obroty kamery są wywoływane naciśnięciami odpowiednich przycisków klawiatury. Opis przypisania klawiszy do poszczególnych ruchów kamery znajduje się w tekście wyświetlanym w górnej części okienka.
   5. Zdefiniowane zostały podstawowe ruchy kamery:
      1. Translacje: góra, dół, prawo, lewo, przód, tył
      2. Obroty: „zegarowy” zgodny i przeciwny do ruchu wskazówek, pochylanie, podnoszenie, skręt w lewo i w prawo
      3. Zoom: przybliżanie i oddalanie (realizowane poprzez zmianę odległości kamery od rzutni, przy zachowaniu wymiarów rzutni)
      4. Home: zawsze powrót do pozycji startowej
   6. Definicje rozmiaru okna, obszaru widzialnego, położenia i rozmiarów prostopadłościanów, wielkości skoku translacji i obrotów, odległości kamery od rzutni znajdują się w kodzie programu (na początku pliku kamera.cpp).
   7. 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).
   8. Program nie wyświetla obiektów, które przecinają płaszczyznę rzutni
3. Opis działania programu
   1. Utworzenie tablicy dynamicznej na bieżące współrzędne wierzchołków i przepisanie do niej wartości początkowych. Tablica ta określa zawsze bieżące położenie obiektów. Początkowe dane zostają zachowane w celu powrotu do pozycji startowej.
   2. Utworzenie obrazu startowego na podstawie zdefiniowanych parametrów
   3. Utworzenie obramowania okna, osi współrzędnych oraz opisu
   4. Oczekiwanie na wciśnięcie odpowiednich klawiszy sterujących ruchem kamery
   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. Posortowanie obiektów od najdalszych do najbliższych
      1. Sortowanie według odległości od środka ciężkości obiektu do kamery
      2. Procedura uruchamiana tylko po ruchach kamery, które mogą wpłynąć na zmiany wzajemnego zasłaniania obiektów: obroty
   7. Utworzenie i posortowanie tablicy ścian dla każdego obiektu
      1. Sortowanie według odległości od środka ciężkości ściany do kamery
   8. Wyczyszczenie okna
   9. Utworzenie nowego obrazu
   10. Utworzenie obramowania okna, osi współrzędnych oraz opisu
   11. Oczekiwanie na wciśnięcie odpowiednich klawiszy sterujących ruchem kamery
4. Procedura tworzenia obrazu
   1. Sprawdzenie, czy obiekt nie przecina rzutni, a jeśli przecina, to pominięcie go przy wyświetlaniu
   2. Rysowanie wszystkich prostopadłościanów od najdalej położonego do najbliższego
      1. Algorytm malarski zasłaniania elementów niewidocznych
      2. Następuje zamalowywanie elementów zasłoniętych przez obiekty położone bliżej kamery
   3. Rzutowanie wszystkich wierzchołków prostopadłościanu
   4. Sprawdzanie i oznaczenie położenia każdego wierzchołka na rzutni według algorytmu Sutherlanda-Hodgmana
   5. Rysowanie wszystkich ścian w kolejności od najdalej położonej do najbliższej
      1. Algorytm malarski zasłaniania ścian niewidocznych
      2. Następuje zamalowywanie ścian „tylnych” przez ściany położone bliżej kamery
   6. Ściany wychodzące poza wyznaczony obszar widoczności są obcinane:
      1. Analizowane jest położenie wierzchołków kolejno każdej krawędzi i ewentualne odpowiednie obcięcie krawędzi jedno lub dwustronne lub jej odrzucenie
      2. Tworzony jest wielokąt do narysowania, jako tablica kolejnych, nowych wierzchołków utworzonych na podstawie analizy i obcinania krawędzi ścian
      3. Do tablicy wpisywane są kolejno wierzchołki z kolejnych krawędzi, oryginalne lub po przycięciu krawędzi
      4. Wierzchołki krawędzi odrzuconych są pomijane
      5. W przypadku, kiedy dwa kolejne wierzchołki tworzonego wielokąta (nowe) znajdują się na dwóch różnych brzegach obszaru rysowania (po przycięciu jeden znajduje się np. na lewym brzegu a drugi na górnym) następuje analiza położenia ściany względem odpowiedniego rogu obszaru rysowania (np. lewy-górny róg)
         1. Przy potwierdzeniu przynależności rogu do wnętrza rysowanej ściany dodawany jest kolejny punkt wielokąta ze współrzędnymi analizowanego rogu
      6. W ten sposób obcięta ściana może przyjąć kształt wielokąta o ilości boków od 3 do 8.
      7. Program pomija analizę przypadków, kiedy ściana ma wymiary większe od szerokości lub wysokości wyświetlanego obrazu – wtedy przycięcie ściany mogłoby się wiązać z dodaniem dwóch dodatkowych kolejnych wierzchołków – taki przypadek nie został w programie uwzględniony
   7. W celu łatwej identyfikacji obiektów sceny, każdy obiekt został pomalowany innym kolorem
5. Dodatkowe uwagi
   1. Dodatkowo w stosunku do projektu z I etapu w celu uzyskania mniejszych błędów przeliczeń, zastosowano format danych float przy przeliczaniu wszystkich zmian położeń i odległości (w etapie I część obliczeń była wykonywana na zmiennych typu integer). Konwersja na typ całkowity została zastosowana dopiero podczas wpisywania współrzędnych wierzchołków po rzutowaniu. Przy wszystkich operacjach konwersji niejawnej typu float na int/long dodano wartość 0.5, celem uzyskania właściwego zaokrąglenia liczby.
6. 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.