Maciej Krosman 257341

Sprawozdanie z projektów 1. i 2a.

Cel i temat projektu.

Celem projektu było napisanie programu komputerowego, w dowolnej technologii i języku programowania, realizującego wirtualną kamerę(projekt 1.) oraz eliminację elementów zasłoniętych dowolnym algorytmem (projekt 2a.).

Założenia.

1. Scena będzie zbudowana z wielościanów zwykłych w sensie Eulera.
2. Wielościany nie będą się stykać i przecinać.
3. Wielościany będą prostopadłościanami.
4. Każda ściana prostopadłościanu będzie zbudowana z 2 trójkątów.

Ogólny schemat działania programu:

1. Utworzenie sceny i ustawienie podstawowych parametrów(np. fi rotacji, Tx translacji, odległość rzutni).
2. Przetworzenie sceny:
   a. Transformacje trójkątów.
   b. Odrzucenie niewidocznych trójkątów.
   c. Rzutowanie trójkątów.
   d. Odrzucenie trójkątów wychodzących poza okno programu.
   e. Przekształcenie do współrzędnych 2D rysunku.
   f. Eliminacja elementów zasłoniętych i pomalowanie odpowiednich pikseli.
   g. Wyświetlenie bufora rysunku w oknie programu.
3. Oczekiwanie na naciśnięcie klawisza, aktualizacja macierzy

Szczegóły przetwarzania sceny.

1. Transformacje trójkątów.

Macierz transformacji geometrycznych jest mnożona przez macierz nowej transformacji. Wektor współrzędnych każdego punktu każdego trójkąta mnożony jest przez wynikową macierz transformacji.

$M = M_n * \ldots * M_2 * M_1$

$P' = M * P$

2. Odrzucanie niewidocznych trójkątów.

Wierzchołki trójkątów są tak zorientowane, aby wektor normalny powierzchni zawierającej trójkąt był skierowany na zewnątrz prostopadłościanu. Dzięki temu tylne trójkąty mogą być zidentyfikowane przez nieujemny iloczyn skalarny wektora normalnego trójkąta i wektora od położenia obserwatora do dowolnego punktu trójkąta. Wektor normalny trójkąta, tworzony jest z współrzędnych równania płaszczyzny: [A, B, C]. Jako dowolny punkt trójkąta załóżmy punkt P1 = (x1,y1,z1). Położenie obserwatora jest stałe i znajduje się w początku układu współrzędnych ([0, 0, 0]). Jeżeli więc

A*x1 + B*y1 + C*z1 < 0

to trójkąt jest skierowany we właściwym kierunku i jest zachowywany do dalszego wykorzystania.

Ponadto sprawdzane jest czy położenie dowolnego punktu trójkąta względem rzutni jest nieujemne(z1>=1) co zapewnia, że trójkąt o właściwym skierowaniu ale znajdujący się ze złej strony rzutni nie będzie zachowany.

3. Rzutowanie trójkątów.

Zastosowałem rzutowanie perspektywiczne. Rzutnia jest równoległa do płaszczyzny OXY i odległa od początku układu współrzędnych początkowo o d=1, które jest zmieniane przy operacji przybliżania i oddalania. Środek rzutowania znajduje się w początku układu współrzędnych. Punkt P = (x , y , z) jest rzutowany na punkt

$$P' = \left(\frac{x*d}{z}, \frac{y*d}{z}, d\right)$$

4. Odrzucenie trójkątów wychodzących poza okno programu.

Rzutnia jest ograniczona: -1 <= x <= 1 oraz -1 <= y <= 1.

Trójkąty których przynajmniej jeden punkt nie znajduje się na rzutni są odrzucane.

5. Przekształcenie do współrzędnych 2D rysunku.

Współrzędne punktów P(x, y) dla wszystkich krawędzi są przekształcane wg wzorów:

$$x' = \frac{(x - x_{min}) * szerokość}{x_{maks} - x_{min}} \qquad y' = wysokość - \frac{(y - y_{min}) * wysokość}{y_{maks} - y_{min}}$$

gdzie: x_{min}, x_{maks}, y_{min}, y_{maks} – ograniczenia okna na rzutni

wysokość, szerokość – rozmiary rysunku

Współrzędne są zaokrąglane do liczb całkowitych.

6. Eliminacja elementów zasłoniętych i pomalowanie odpowiednich pikseli.

   Do eliminacji elementów zasłoniętych zastosowałem algorytm skaningowy.

   a. Koloruje cały bufor na kolor biały.
   b. Dla każdej linii poziomej pikseli składającej się na okno, tworzę listę skanlinii - linia utworzona z przecięcia linii poziomej z trójkątem.
   c. Dla każdej piksela od najbardziej lewego należącego do listy skanlinii do najbardziej prawego:
      1. sprawdzam czy występuje konflikt między pojedynczymi skanliniami
      2. w przypadku konfliktu wybieram najbliższą skanlinie i maluje dany piksel kolorem wybranej skanlinii
      3. w przypadku braku konfliktu maluje kolorem aktualnej skanlinii

7. Wyświetlenie bufora rysunku w oknie programu.

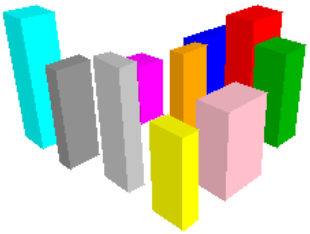
   Buforem jest lista/tablica kolorów buffer[wysokość*y+x] = pixel(x,y).

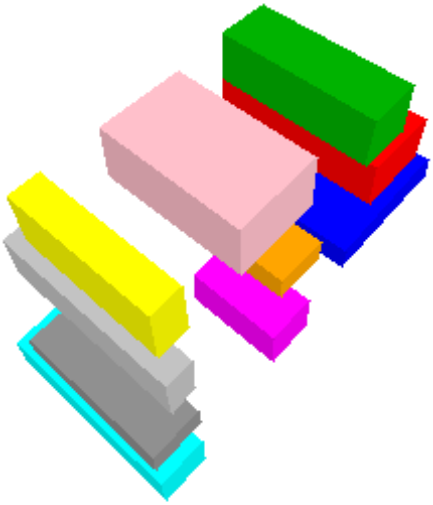
   Bufor jest wyświetlany przy pomocy prostych funkcji i struktur biblioteki SDL2.

Uzyskany efekt.

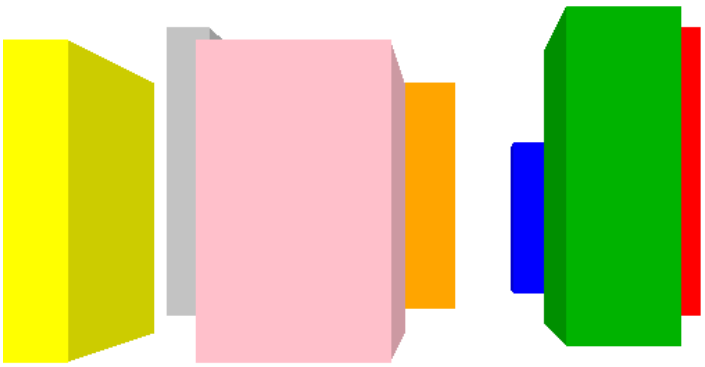
Udało się zrealizować zakładaną funkcjonalność. Wszystkie warianty położenia są realizowane. Efekt jest zgodny z wyobrażeniem. Scenę można oglądać z każdej strony, za każdym razem otrzymując zakładany wynik.

Testowana scena:





Translacja w kierunku sceny



Zoom na scenę