Motyl, wstęga Möbiusa i dwunastościan

Celem zadania jest wprowadzenie do programowania w DirectX oraz pokazania różnych metod konstruowania macierzy przekształceń dla obiektów trójwymiarowej sceny. Do Operacji na wektorach i macierzach posłuży nam biblioteka DirectXMath dołączona do Windows SDK. Dodatkowo wykorzystane zostaną funkcjonalności Direct3D jak: bufor szablonu, mieszanie kolorów, shadery wierzchołków i pikseli.

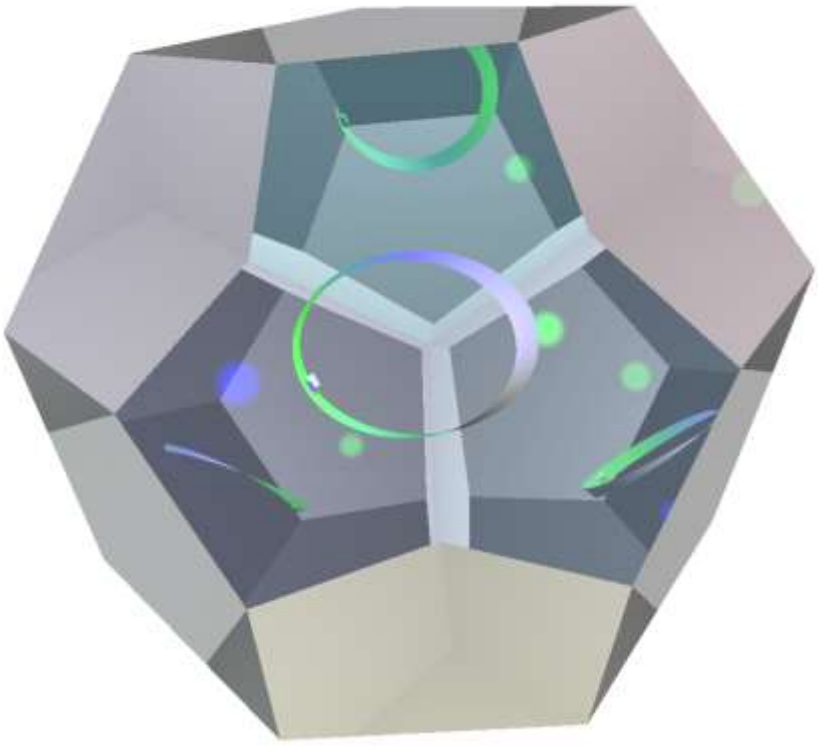
Treść zadania:

Wzdłuż wstęgi Möbiusa (utworzonej z możliwie małej liczby trójkątów), umieszczonej w centrum sceny, i stycznie do niej porusza się motyl, machając skrzydełkami. Motyl modelowany jest przy pomocy dwóch prostokątów odpowiadających skrzydłom. Machanie skrzydłami polega na zmianie kąta rozwarcia kąta rozwarcia między nimi.

Kamera sparametryzowana jest przez współrzędne sferyczne, dwa kąty i promień, z których każdą można niezależnie modyfikować. Kamera, wstęga i motyl znajdują się wewnątrz dużego dwunastościanu foremnego (pięciokątne ściany).

Scena odbija się w każdej z jego ścian. Do uzyskania efektu wykorzystany jest bufor szablonu w celu obcięcia obszaru rysowania odbitej geometrii, pomijając jednocześnie ściany znajdujące z tyłu kamery. Efekt uwzględnia tylko jednokrotne odbicia.

Scena oświetlona jest przy pomocy trzech świateł punktowych: zielonego, niebieskiego i białego. Rozmieszczone są one w różnych punktach i oświetlenie nie uwzględnia zaniku światła wraz z odległością. Odbita geometria jest odpowiednio oświetlona przez odbite wersje tych źródeł światła.



W miejscu każdego źródła światła rysowany jest bilbord, włączając źródła światła odbite. Bilbordy realizowane są przez ustawienie macierzy przekształcenia dla kwadratu jednostkowego zdefiniowanego w płaszczyźnie XY.

Po narysowaniu odbić, używając mieszania alfa przy rysowaniu powierzchni, lustra pokryte są półprzeźroczystymi kolorami mającymi pomoc w orientacji w przestrzeni.

Wstępne informacje na temat macierzy przekształceń 3D.

Korzystając z biblioteki DirectXMath warto pamiętać, że wektory przekształcane są przez macierze według schematu:

$$\boldsymbol{v'} = \boldsymbol{vM} \qquad \begin{vmatrix} v'_x & v'_y & v'_z & v'_z \end{vmatrix} = \begin{vmatrix} v_x & v_y & v_z & v_w \end{vmatrix} \begin{vmatrix} a_x^x & a_y^x & a_z^x & 0 \\ a_x^y & a_y^y & a_z^y & 0 \\ a_x^z & a_y^z & a_z^z & 0 \\ t_x & t_y & t_z & 1 \end{vmatrix}$$

Wynika z tego, że jeżeli punkt ma zostać przekształcony kolejno przez macierze $M_1, M_2, \dots, M_n$, to złożenie tych przekształceń będzie określone za pomocą macierzy $M = M_1 \cdot M_2 \cdot \dots \cdot M_n$. W pamięci wektory przechowywane są w taki sposób, że po sobie wierszami następują kolejne wektory. Warto zauważyć, że w Direct3D można używać zarówno lewo- jak i prawoskrętnego układu współrzędnych, częściej używany jest układ lewoskrętny (oś z skierowana jest w głąb ekranu).

Przy przekształcaniu punktów przyjmuje się, że czwarta współrzędna wektora jest równa 1, a przy przekształcaniu wektorów (kierunków), za czwartą współrzędną wektora przyjmuje się 0. Mimo to, jeśli macierz przekształcenia nie reprezentuje przekształcenia podobieństwa, to kierunki nie zostaną poprawnie obrócone. Do poprawnego przekształcania samego kierunku wektora, bez zmiany jego długości, powinno się używać transpozycji odwrotności macierzy $(M^{-1})^T$.

Na tą samą macierz można patrzeć na dwa sposoby:

a) Jako opis transformacji obiektu w scenie 3D.
b) Jako opis konwersji współrzędnych punktu z jednego układu współrzędnych (tzw. Frame) do innego.

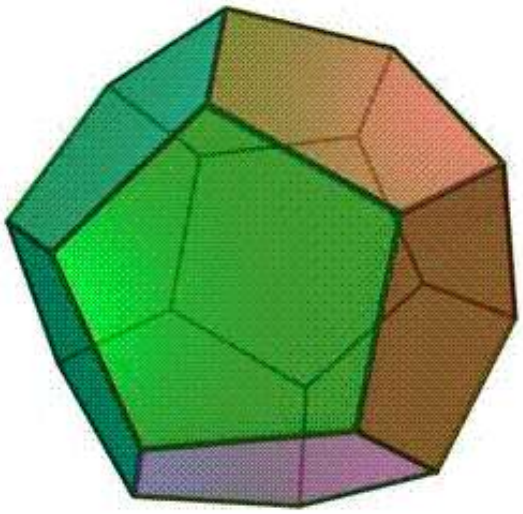
Przyjmijmy, że pewien obiekt trójwymiarowy zdefiniowany jest przy pomocy współrzędnych wyrażonych w jego lokalnym układzie współrzędnych i początkowo ten układ pokrywa się z układem współrzędnych sceny. Jeśli dana jest transformacja tego obiektu przy pomocy macierzy **A** (nowe współrzędne punktów w scenie powstają ze starych poprzez pomnożenie ich przez tą macierz), to macierz **A** jest jednocześnie przekształceniem współrzędnych z lokalnego układu współrzędnych obiektu w nowym położeniu do układu współrzędnych sceny.

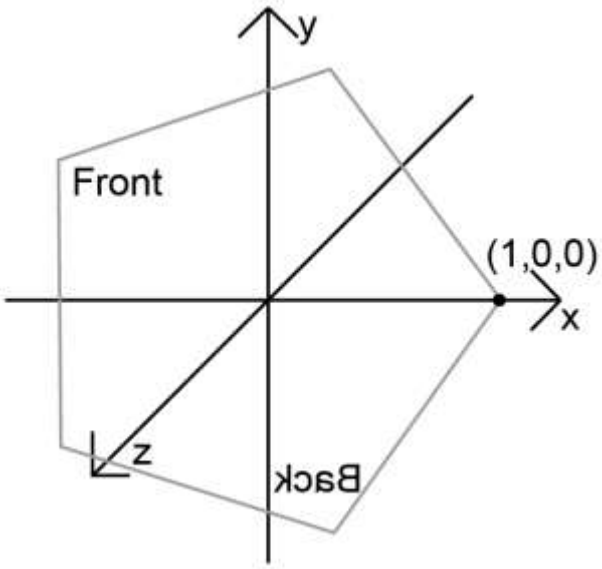
Macierze potrzebne do definicji sceny.

Do narysowania sceny przydatne będą metody konstruowania następujących macierzy, które pozwolą wyrazić współrzędne danych punktów w dowolnym interesującym układzie współrzędnych:

- Macierz przekształcenia z układu sceny do układu kamery (**m_camera.getViewMatrix()**).
- Macierz przekształcenia współrzędnych z lokalnego układu każdej ściany dwunastościanu do układu sceny (**m_dodecahedronMtx[i]**).
- Macierz przekształcenia współrzędnych z układu sceny odbitej w danym lustrze do oryginalnego układu sceny (**m_mirrorMtx[i]**).
- Macierz przekształcenia współrzędnych z układu statycznego do wstęgi Möbiusa w punkcie (s, t), gdzie $s = 0.5$, a $t \in [0; 1)$ (**moebiusMtx(float t)**).
- Macierz przekształcenia współrzędnych z lokalnego układu bilbordu zaczepionego w punkcie **P** do układu sceny (**billboardMtx(vector p)**).
- Macierz przekształcenia z lokalnego układu kwadratu reprezentującego lewe lub prawe skrzydło motyla, do układu współrzędnych motyla (**wingMtx[i]**).

Konstrukcja dwunastościanu foremnego.





Ściany dwunastościanu foremnego są pięciokątami. Do jego zbudowania posłużymy się siatką trójkątów pięciokąta foremnego, zdefiniowanego w lokalnym układzie współrzędnych w sposób pokazany powyżej. Wierzchołki znajdują się na okręgu jednostkowym w płaszczyźnie XY, zaczepionym w środku układu. Przyjmijmy zgodną z ruchem wskazówek zegara kolejność wierzchołków dla trójkątów zwróconych przodem i utwórzmy jeden wachlarz trójkątów dla pięciokąta.

Następnie skonstruujemy dwanaście macierzy przekształceń, z których każda umieści pięciokąt w miejscu jednej ze ścian dwunastościanu foremnego tak, żeby przednia strona pięciokąta zwrócona była do wewnątrz. Macierze te przydadzą się nie tylko do narysowania samej bryły na ekranie, ale również przy rysowaniu odbicia lustrzanego w danej ścianie do konstrukcji macierzy odbicia. Z tego powodu najlepiej policzyć je i zapamiętać w tablicy.

Do konstrukcji macierzy przyda się znajomość pewnych faktów dotyczących kątów i odległości w dwunastościanie foremnym. Poniżej zamieszczone są wartości dla przypadku, gdy okrąg opisany na każdej ze ścian jest jednostkowy. Więcej informacji o dwunastościanie foremnym można znaleźć po tymi adresami: http://en.wikipedia.org/wiki/Dodecahedron , http://mathworld.wolfram.com/Dodecahedron.html .

Promień okręgu wpisanego w ścianę boczną:

$$\boldsymbol{dodecahedron\_r} = \sqrt{\frac{3}{8} + \frac{1}{8}\sqrt{5}}$$

Odległość między dwoma równoległymi ścianami:

$$\boldsymbol{dodecahedron\_h} = 1 + 2\,\boldsymbol{dodecahedron\_r}$$

Kąt dwuścienny:

$$\boldsymbol{dodecahedron\_a} = \arccos\left(-\frac{\sqrt{5}}{5}\right)$$

Niech przekształcanie danego pięciokąta na dolną ścianę dwunastościanu będzie następstwem dwóch kroków:

1. _____ 2. _____

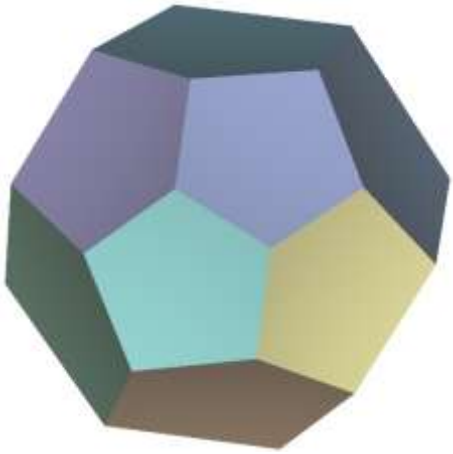
Przekształcenie danego pięciokąta na jedną z pięciu ścian bocznych z dolnej połowy jest nieco bardziej złożone i można je przedstawić jako złożenie następujących kroków:

1. _____ 2. _____
3. _____ 4. _____
5. _____ 6. [opcjonalnie] _____

Górną połówkę dwunastościanu łatwo uzyskać z dolnej obracając o 180° wokół osi Z.

Zgodnie z treścią zadania z każdą ścian ma być związany inny kolor. Kolory zdefiniowane powinny być w następujący sposób:

```
colors[12] = {
      (253, 198, 137), (255, 247, 153), (196, 223, 155), (162, 211, 156),
      (130, 202, 156), (122, 204, 200), (109, 207, 246), (125, 167, 216),
      (131, 147, 202), (135, 129, 189), (161, 134, 190), (244, 154, 193) }
```



Konstrukcja macierzy odbicia lustrzanego dla każdej ściany dwunastościanu.

Interesuje nas macierz odbicia lustrzanego przekształcająca współrzędne w układnie sceny na współrzędne po odbiciu, również w układzie sceny. Z kolei odbicie jest dane w trywialny sposób w układzie lokalnym, związanym z płaszczyzną lustra, za pomocą skali o współczynnikach $(1,1,-1)$. Dysponując przekształceniem współrzędnych z układu lokalnego jednego z dwunastu luster do układu sceny (**dodecahedronMtx[i]**) możemy łatwo wyrazić macierz odbicia:

$$\boldsymbol{mirrorMtx}[\boldsymbol{i}] = \_\_\_\_\_ * \_\_\_\_\_ * \_\_\_\_\_$$

Konstrukcja wstęgi Möbiusa.

Wprowadźmy dwie stałe, które będą określały wielkość i szerokość wstęgi Möbiusa:

```
const float MOEBIUS_R =1.0f;
const float MOEBIUS_W = 0.1f;
```

Skorzystajmy z następującej parametrycznej definicji wstęgi:

$$P(s,t) = \begin{vmatrix} x \\ y \\ z \end{vmatrix} = \begin{vmatrix} \cos(t) * \left(moebiusR + moebiusW * s * \cos\left(0.5t\right)\right) \\ \sin(t) * \left(moebiusR + moebiusW * s * \cos\left(0.5t\right)\right) \\ moebiusW * s * \sin\left(0.5t\right) \end{vmatrix}$$

gdzie $s \in [-1;1], t \in [0;4\pi)$.

Do policzenia normalnej oraz przestrzeni stycznej potrzebne będą również parametryzacje dwóch wektorów stycznych do powierzchni – pochodne cząstkowe powyższej parametryzacji:

$$P_t = \begin{vmatrix} x_t \\ y_t \\ z_t \end{vmatrix} = \begin{vmatrix} -moebiusR * \sin(t) - 0.5s * moebiusW * \sin(0.5t)\cos(t) - moebiusW * s * \cos(0.5t) * \sin\left(t\right) \\ moebiusR * \cos(t) - 0.5s * moebiusW * \sin(0.5t) * \sin(t) + moebiusW * s * \cos(0.5t)\cos\left(t\right) \\ 0.5s * moebiusW * \cos\left(t\right) \end{vmatrix}$$

$$P_s = \begin{vmatrix} x_s \\ y_s \\ z_s \end{vmatrix} = \begin{vmatrix} \cos(0.5t)\cos\left(t\right) \\ \cos(0.5t)\sin\left(t\right) \\ \sin\left(0.5t\right) \end{vmatrix}$$

W tym przypadku, te dwa wektory są prostopadłe dla ustalonych *s* i *t*. Po znormalizowani i dodaniu znormalizowanego iloczynu wektorowego $P_s \times P_t$ powstanie ortonormalna baza przestrzeni stycznej. Znormalizowane P_t będziemy nazywać styczną, znormalizowane $P_s \times P_t$ normalną, a znormalizowane P_s binormalną. Jeśli te trzy wektory (z czwartą współrzędną równą 0) oraz wektor P (z czwartą współrzędną równą jeden) w tej kolejności zapiszemy w wierszach macierzy, to otrzymamy macierz przekształcenia współrzędnych punktu z przestrzeni stycznej do przestrzeni świata. Przyda się ona do umieszczenia motyla w scenie.

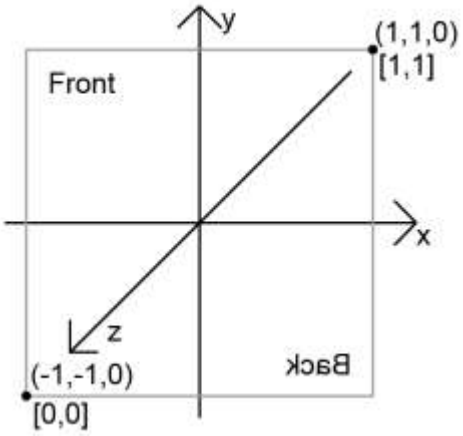
Geometrię wstęgi zdefiniujemy w postaci jednego paska trójkątów. Wierzchołki i normalne policzymy dla $s \in \{-1,1\}$ oraz dla zdyskretyzowanych wartości *t* po podziale przedziału $[0,2\pi)$ na np. 128 części. Trzeba pamiętać o podzieleniu całego przedziału $[0,4\pi)$, aby stworzyć trójkąty po obu stronach wstęgi z odwróconymi normalnymi.

Motyl.

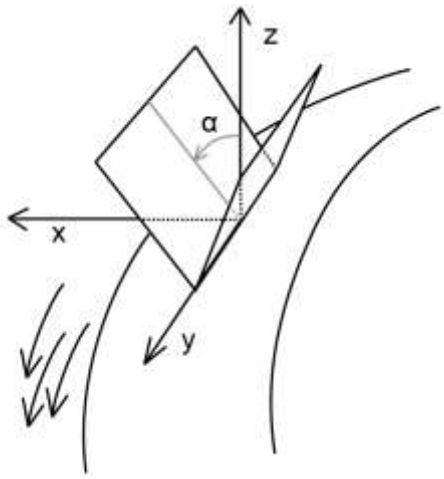
Motyl składa się z dwóch prostokątów. Wspólna krawędź prostokątów jest styczna do wstęgi Möbiusa. Punktem styczności jest jej środek. Oba prostokąty odchylone są od normalnej wstęgi o kąt ±α. Kąt dany jest jako funkcja od czasu. Oznaczmy: $k = \frac{t}{T} - \left\lfloor \frac{t}{T} \right\rfloor$, gdzie t do czasu od początku symulacji, T to okres ruchu skrzydeł.

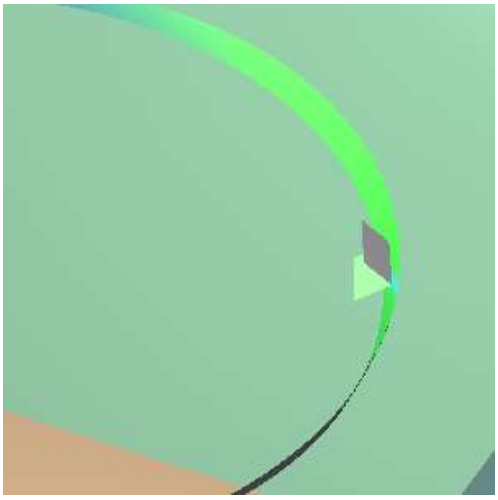
$$\propto = \begin{cases} 80° * 2k, & gdy\ k \leq 0.5 \\ 160° - 80° * 2k, & gdy\ k > 0.5 \end{cases}$$

Niech siatka dla kwadratu będzie zdefiniowana w następującym układzie współrzędnych:



Musimy wyznaczyć macierze przekształceń, które przetransformują wierzchołki tego kwadratu na wierzchołki prostokątów skrzydeł w układzie stycznym do wstęgi Möbiusa, a następnie użyć macierzy przejścia współrzędnych z układu stycznego wstęgi do układu sceny (budowa tej macierzy została już opisana w punkcie dotyczącym konstruowania wstęgi). W przestrzeni stycznej do wstęgi motyla wygląda tak:





Rysowanie odbić w lustrze.

Idea rysowania odbić w płaskich lustrach polega na tym, żeby narysować kopię całej sceny przekształconej przy pomocy macierzy odbicia, jednocześnie zapewniając, żeby obszar rysowania był ograniczony na ekranie do pikseli lustra. Rysując odbicia lustrzane trzeba pamiętać o kilku rzeczach:

- Po odbiciu zmienia się na przeciwną orientacja trójkątów: jeśli w oryginalnej scenie przednią stronę wyznaczało ułożenie kolejnych wierzchołków zgodnie z ruchem wskazówek zegara, to po odbiciu kolejne wierzchołki dla tej samej strony trójkąta ułożone będą przeciwnie do ruchu wskazówek zegara. Ma to znaczenie dla obcinania trójkątów zwróconych tyłem.
- Odbić należy nie tylko geometrię, ale również położenia i kierunki świateł.
- Odbita scena korzysta z tego samego bufora głębokości, co oryginalna. Trzeba zapewnić, żeby narysowanie jednej z nich nie wpływało na wyniki testów głębokości podczas rysowania drugiej. W przypadku tego zadania wystarczy, żeby wszystkie odbicia lustrzane były renderowane przed oryginalną sceną. Dowolny obiekt odbitej sceny znajdzie się dalej od obserwatora niż dowolny obiekt oryginalnej, więc niezależność każdej takiej pary scen jest zapewniona. Z kolei dwie sceny odbite w dwóch różnych lustrach modyfikują i testują bufor głębokości tylko dla pikseli należących do obrazów tych luster na ekranie, a te zbiory pikseli są oczywiście rozłączne.

Poniżej przedstawione zostanie sposób rysowania odbić lustrzanych z użyciem bufora szablonu. Bufor szablonu przyporządkowuje każdemu pikselowi dodatkowe bity informacji. Ilość dodatkowych bitów na piksel określa się podczas inicjalizowania kontekstu renderowania. Kontekst Direct3D ma domyślnie wspólny bufor głębokości i szablonu, gdzie z 32 bitów na piksel, 24 poświęcone są na bufor głębokości, a pozostałe 8 na bufor szablonu. Sposób wykorzystania bufora szablonu dany jest w postaci:

1. wartości referencyjnej – ustalonej liczby całkowitej nieujemnej
2. sposobu modyfikacji bufora szablonu podczas rysowania pikseli na ekranie (w zależności od zdania testu szablonu, głębokości oraz czy ściana zwrócona jest przodem czy tyłem do kamery).
3. rodzaju testu bufora szablonu.

Wykorzystamy bufor szablonu w sposób następujący. Na początek bufor zostanie wyczyszczony – wypełniony zerami. Algorytm renderowania pojedynczego lustra z użyciem bufora szablonu będzie wyglądać tak:

1. Pięciokąt każdej ze ścian bocznych dwunastościanu zostanie narysowany przy wyłączonym zapisie do bufora głębokości, tylko po to, żeby podczas rysowania należących do niego pikseli w buforze szablonu zapisać niezerowy, unikalny indeks ściany. Na tym etapie ustawiona jest operacja modyfikacji bufora szablonu, ale nie ma testu bufora szablonu. Po tym zabiegu piksele na ekranie zostaną na podstawie wartości w buforze szablonu przyporządkowane do odpowiednich luster.
2. Każda z dwunastu odbitych scen zostanie narysowana (przy włączonym zapisie do bufora głębokości) z testem bufora szablonu „równe indeksowi ściany”. Przez macierz odbicia mnożona będzie macierz widoku po to, aby przekształceniu do układu sceny odbitej uległy nie tylko wierzchołki na scenie, ale i pozycje świateł.

3. Pięciokąt każdej ze ścian bocznych dwunastościanu może teraz zostać narysowany ponownie, tym razem normalnie: z modyfikacją bufora głębokości, bez testu i modyfikowania bufora szablonu. Należy użyć mieszania alfa w celu uzyskania półprzezroczystości.
4. Rysowana jest oryginalna scena – wnętrze dwunastościanu foremnego.

Bilbordy.

Będziemy reprezentować światła w postaci bilbordów. Bilbord to prostokąt o środku zaczepionym w określonym punkcie sceny, którego orientacja zawsze dopasowuje się do kamery. Niezależnie położenia i orientacji kamery widzimy powierzchnię prostopadłą do kierunku patrzenia.

Światła będą reprezentowane przez kwadratowe bilbordy. Przez kolor światła mnożona będzie wartość odcienia szarości, którego jasność będzie równa $\max\left(0; 1-\frac{d}{0.5a}\right)$, gdzie *a* to bok bilbordu, *d* to odległość piksela od jego środka. Operacje te zostaną zaimplementowane w postaci shadera pikseli. Wynikowy kolor zostanie dodany do koloru tła – „rozjaśni je".

Mieszanie addytywne w przeciwieństwie do zwykłej przezroczystości nie wymaga określonej kolejności rysowania obiektów przezroczystych. Wymaga jednak dwóch rzeczy:

a) Nieprzezroczyste elementy sceny muszą zostać narysowane wcześniej.
b) Obiekty ze zwykłą przezroczystością muszą być rysowane po wszystkich obiektach z mieszaniem addytywnym, które częściowo są przez niezasłonięte, a przed wszystkimi, które mogą je zasłaniać.
c) Zapis do bufora głębokości powinien być wyłączony (ale test bufora głębokości nie). Wyobraźmy sobie scenariusz, w którym warunek ten nie jest spełniony i rysowane są dwa bilbordy, przy czym drugi rysowany jest częściowo przesłonięty przez pierwszy. Pierwszy bilbord ustawi wartość w buforze głębokości dla całego swojego prostokąta. W przesłoniętej części drugiego bilbordu żaden piksel nie przejdzie testu bufora głębokości.

Chcemy, żeby bilbordy były zrealizowane poprzez narysowanie kwadratu z odpowiednio ustawioną macierzą rzutowania. Wymaga to pewnej uwagi. Przypuśćmy, że mamy daną macierz przekształcenia środka bilbordu do odpowiedniego punktu sceny. Potrzebujemy teraz macierzy podobieństwa, która środek danego kwadratu przemieści w ten sam sposób, ale jednocześnie zapewni, że boki kwadratu będą miały określoną długość w układzie świata, oraz że będą one miały w układzie sceny te same kierunki, co osie układu ekranu.

Szukana macierz przejścia z układu lokalnego danego kwadratu do układu świata jest złożeniem trzech macierzy: T*S*R.

- T to macierz translacji o wektor będący pozycją środka bilbordu w układzie kamery.
- S to macierz skalowania jednorodnego o połowę długości boku bilbordu.
- R to macierz orientacji bilbordu w układzie świata. W układzie kamery byłaby to macierz jednostkowa. W układzie sceny będzie to iloczyn macierzy translacji o wektor przeciwny do położenia kamery w układzie świata oraz macierzy odwrotnej do macierzy kamery V. Położenie kamery w układzie świata można zapisać jako: $V^{-1}[0 \quad 0 \quad 0 \quad 1]^T$

Również przy renderowaniu bilbordów w scenach odbitych należy pamiętać o wyłączeniu zapisu do bufora głębokości (zostawiając włączony test bufora szablonu).