

Grafika czasu rzeczywistego

Krzysztof Gdawiec



UNIWERSYTET ŚLĄSKI
INSTYTUT INFORMATYKI

Obiekty bufora obrazu

Biblioteka WebGL obok możliwości rysowania grafiki w oknie lub na całym ekranie oferuje rysowanie w tzw. obiektach bufora obrazu (ang. framebuffer object – FBO).

Domyślny obiekt FBO jest wiązany z tworzonym oknem. Możemy utworzyć wiele obiektów FBO i renderować obrazy w nich zamiast w oknie. Jest to tzw. metoda renderowania pozaekranowego (ang. off-screen rendering).

Technika ta stosowana jest do otrzymania wielu różnych efektów, np. mapowanie cieni, odbicie, przetwarzanie końcowe (ang. postprocessing).

Generowanie identyfikatora obiektu FBO:

```
gl.createFramebuffers();
```

Dowiązanie obiektu FBO:

```
gl.bindFramebuffer(target, framebuffer);
```

`target` określa rodzaj obiektu FBO i przyjmuje wartość `gl.FRAMEBUFFER`, zaś `framebuffer` to identyfikator obiektu FBO.

W danej chwili możemy mieć dowiązany tylko jeden obiekt FBO do rysowania i jeden do odczytu.

Usuwanie obiektów FBO:

```
gl.deleteFramebuffer(framebuffer);
```

`framebuffer` – obiekt FBO do usunięcia.

Po utworzeniu obiektu FBO musimy do niego dołączyć obiekty buforów renderowania (ang. `renderbuffer object` – RBO).

Generowanie identyfikatora obiektu RBO:

```
gl.createRenderbuffer();
```

Usuwanie obiektu RBO:

```
gl.deleteRenderbuffer(renderbuffer);
```

`renderbuffer` – obiekt RBO do usunięcia.

Dowiązanie obiektu RBO:

```
gl.bindRenderbuffer(target, renderbuffer);
```

`target` przyjmuje wartość `gl.RENDERBUFFER`, a `renderbuffer` to identyfikator obiektu RBO.

Po dowiezaniu obiektu RBO musimy przydzielić mu pamięć:

```
gl.renderbufferStorage(target, internalformat, width, height);
```

`target` przyjmuje wartość `gl.RENDERBUFFER`, `width`, `height` to wymiary (w pikselach) obiektu RBO, a `internalformat` to format wewnętrzny: `gl.DEPTH_COMPONENT16`, `gl.RGBA4`, `gl.RGB5_A1`, `gl.RGB565`, `gl.STENCIL_INDEX8`.

Dowiązanie obiektu RBO do obiektu FBO:

```
gl.framebufferRenderbuffer(target, attachment,  
renderbuffertarget, renderbuffer);
```

`target` przyjmuje wartość `gl.FRAMEBUFFER`, `renderbuffertarget` przyjmuje wartość `gl.RENDERBUFFER`, `renderbuffer` to identyfikator obiektu RBO, zaś `attachment` to tzw. punkt wiązania:

```
gl.DEPTH_ATTACHMENT, gl.STENCIL_ATTACHMENT, gl.COLOR_ATTACHMENT0.
```

Przed użyciem obiektu FBO z dowiązanymi obiektami RBO musimy sprawdzić kompletność bufora obrazu:

```
gl.checkFramebufferStatus(target);
```

`target` przyjmuje wartość `gl.FRAMEBUFFER`. Jeśli funkcja zwróci `gl.FRAMEBUFFER_COMPLETE`, to wszystko jest w porządku. W przypadku gdy coś jest nie tak funkcja zwróci jedną z wartości:

- ▶ `gl.FRAMEBUFFER_UNSUPPORTED` – kombinacja formatów wewnętrznych dowiązanych obrazów nie jest obsługiwana,
- ▶ `gl.FRAMEBUFFER_INCOMPLETE_ATTACHMENT` – jeden z buforów przeznaczonych do renderowania jest niekompletny,
- ▶ `gl.FRAMEBUFFER_INCOMPLETE_MISSING_ATTACHMENT` – brak buforów powiązanych z FBO,
- ▶ `gl.FRAMEBUFFER_INCOMPLETE_DIMENSIONS` – dowiązane obrazy mają różne wymiary.

Renderowanie do tekstury

Do obiektu FBO oprócz obiektów RBO możemy dowieźć teksturę:

```
gl.framebufferTexture2D(target, attachment, textarget, texture,  
level);
```

`target` przyjmuje wartość `gl.FRAMEBUFFER`, `attachment` to punkt wiązania, `textarget` określa typ tekstury (`gl.TEXTURE_2D`), `texture` to identyfikator obiektu tekstury, `level` to poziom mipmapy.

Odczytanie wartości z bufora

```
gl.readPixels(x, y, width, height, format, type, pixels);
```

`x`, `y` – współrzędne pierwszego piksela do odczytania (lewy dolny róg obszaru), `width`, `height` – wymiary obszaru do pobrania, `format` – format danych piksela, `type` – typ danych piksela, `pixels` – obiekt `ArrayBufferView` na pobrane dane.

Selekcja obiektu

Bardzo często potrzebujemy wybrać jakiś obiekt na scenie za pomocą myszki. W przypadku 3D nie jest to zadanie trywialne.

Do selekcji obiektu w 3D istnieją różne algorytmy, np.

- ▶ selekcja z użyciem bufora głębi,
- ▶ selekcja na podstawie koloru,
- ▶ selekcja na podstawie przecięcia z promieniem oka.

Selekcja z użyciem bufora głębi

Potrzebować będziemy sposobu na przeliczenie punktu w oknie na punkt w przestrzeni 3D.

Do tego celu wykorzystujemy wzór:

$$(PVM)^{-1} \begin{pmatrix} \frac{2(w_x - v_0)}{v_2} - 1 \\ \frac{2(w_y - v_1)}{v_3} - 1 \\ 2w_z - 1 \\ 1 \end{pmatrix}$$

gdzie P – macierz rzutowania, V – macierz widoku, M – macierz modelu, (w_x, w_y, w_z) – współrzędne w przestrzeni okna, (v_0, v_1, v_2, v_3) – informacje o obszarze renderingu.

W wielu bibliotekach wzór jest implementowany jako funkcja

`unProject`.

- ▶ Włączamy testowanie głębi.
- ▶ Po kliknięciu myszką odczytujemy wartość z bufora głębi znajdującą się pod kursorem.
- ▶ Przeliczamy punkt z przestrzeni okna (ze współrzędną z równą odczytanej wartości głębi) do przestrzeni modelu – funkcja `unProject`.
- ▶ Znajdujemy obiekt znajdujący się najbliżej punktu wyznaczonego w poprzednim punkcie.

Selekcja na podstawie koloru

- ▶ Każdy obiekt na scenie ma przypisany unikatowy kolor.
- ▶ Renderujemy scenę do dwóch buforów. Pierwszy zawiera normalnie wyrenderowaną scenę – bufor koloru. W drugim renderujemy obiekty używając unikatowych kolorów (nie obliczamy oświetlenia, nie teksturujemy itp.) – bufor selekcji.
- ▶ Po kliknięciu myszką z bufora selekcji odczytujemy kolor pod kursorem – funkcja `glReadPixels`.
- ▶ Znajdujemy obiekt, którego unikatowy kolor równy jest odczytanemu kolorowi.

Selekcja na podstawie przecięcia z promieniem oka

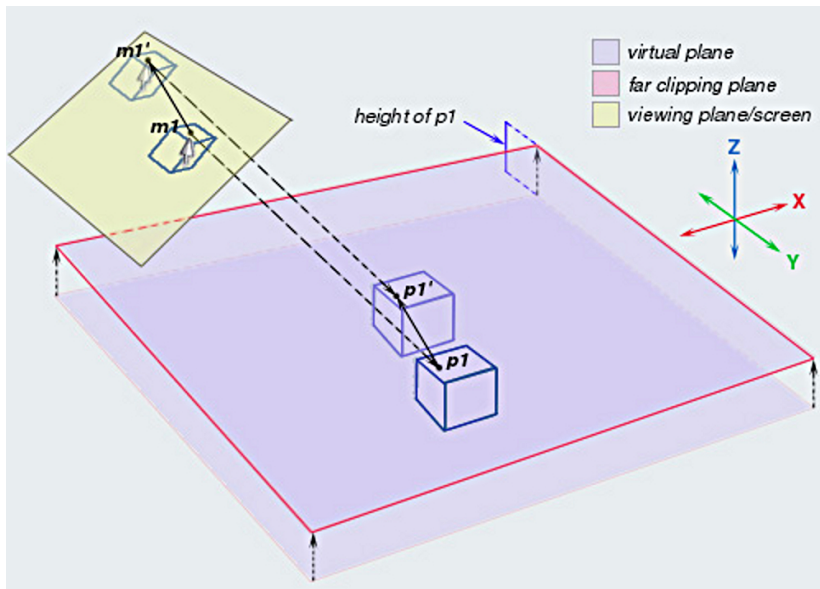
- ▶ Po kliknięciu myszką przeliczamy punkt (w przestrzeni okna) pod kursorem na przestrzeń modelu dla dwóch wartości współrzędnej z : 0 i 1. Otrzymujemy p_0 i p_1 .
- ▶ Tworzymy promień r , którego początkiem jest lokalizacja kamery, a wektor kierunku jest równy $\frac{p_1 - p_0}{\|p_1 - p_0\|}$.
- ▶ Sprawdzamy przecięcie każdego z obiektów na scenie z promieniem r i wybieramy ten, którego punkt przecięcia leży najbliżej kamery.

Sprawdzanie przecięcia całego obiektu z promieniem może być kosztowne. Dlatego możemy zastosować uproszczenie w postaci sprawdzania przecięcia promienia z prostopadłością ograniczającym model. Metoda będzie szybsza, ale mniej precyzyjna.

Poruszanie obiektem po scenie za pomocą myszki

Idea translacji bazującej na wirtualnej płaszczyźnie (w skrócie):

- ▶ Wybieramy obiekt i przesuwamy wirtualną płaszczyznę (płaszczyzna, której nie rysujemy na scenie, najczęściej wyrównana do jednej z płaszczyzn XY , XZ , YZ) do punktu, który został wybrany na obiekcie.
- ▶ Przy ruchu kursora wystrzelujemy promień z punktu, w którym znajduje się kursor w głębi sceny 3D.
- ▶ Testujemy przecięcie tego promienia z wirtualną płaszczyzną.
- ▶ Przesuwamy obiekt o wektor wyznaczony przez obecny punkt przecięcia oraz poprzedni punkt przecięcia.



Selekcja obiektu:

- ▶ wystrzel promień z punktu m_1 (punkt okna), który przecina obiekt; otrzymujemy pozycję piksela w punkcie przecięcia p_1 (punkt w 3D),
- ▶ wyrównaj wirtualną płaszczyznę do jednej z płaszczyzn (XY , XZ , YZ),
- ▶ przesuń wirtualną płaszczyznę na wysokość wyznaczoną przez punkt p_1 .

Ruch obiektem:

- ▶ wykonaj test przecięcia promień-płaszczyzna – wystrzel promień z punktu m'_1 (punkt okna) tak żeby przeciął wirtualną płaszczyznę; otrzymujemy w ten sposób punkt p'_1 ,
- ▶ przesuń obiekt o wektor $p'_1 - p_1$,
- ▶ niech $p_1 = p'_1$ (przygotowanie do następnego ruchu myszką)

Przy takim podejściu występuje jeden szczególny przypadek, gdy płaszczyzna widzenia jest prostopadła do wirtualnej płaszczyzny. Wówczas test przecięcia promień-płaszczyzna zakończy się niepowodzeniem.

Przy takim podejściu występuje jeden szczególny przypadek, gdy płaszczyzna widzenia jest prostopadła do wirtualnej płaszczyzny. Wówczas test przecięcia promień-płaszczyzna zakończy się niepowodzeniem.

Rozwiązaniem tego problemu może być wyrównanie wirtualnej płaszczyzny do płaszczyzny, która nie będzie do niej prostopadła. Do tego celu możemy przeprowadzić test przecięcia promień-płaszczyzna dla trzech możliwych płaszczyzn (XY , XZ , YZ) i wybrać tą, która przeszła ten test.

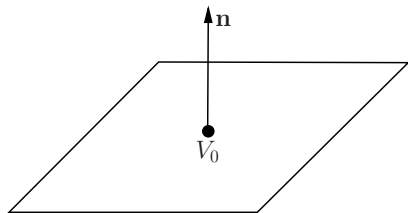
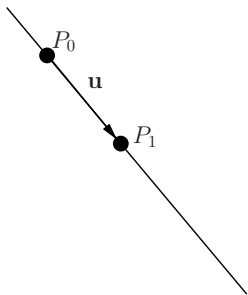
Test przecięcia promień-płaszczyzna

Promień dany jest równaniem parametrycznym:

$$P(s) = P_0 + s(P_1 - P_0) = P_0 + su,$$

gdzie P_0 – punkt początkowy, P_1 – punkt na promieniu, u – wektor kierunku, $s \in [0, +\infty)$.

Płaszczyzna jest dana za pomocą punktu V_0 należącego do tej płaszczyzny oraz wektora normalnego do niej n .

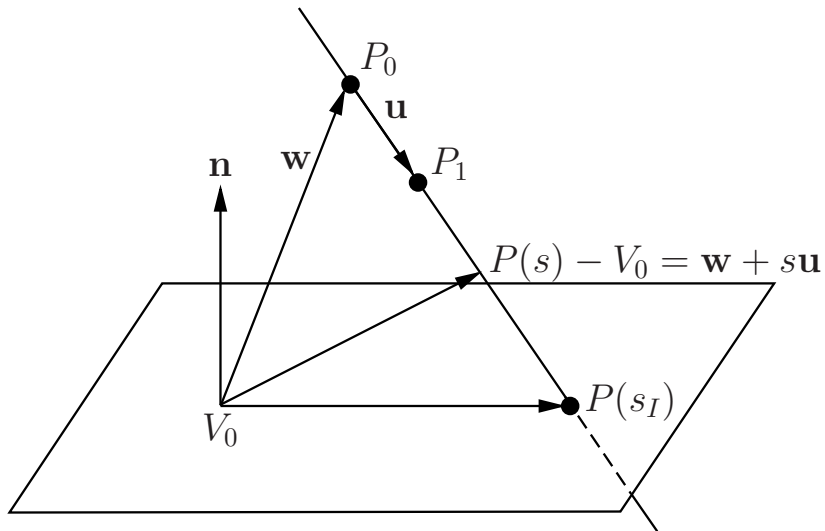


- ▶ Obliczamy:

$$a = n \cdot u,$$

gdzie \cdot to iloczyn skalarny.

- ▶ Jeśli $a = 0$, to promień i płaszczyzna są równoległe lub promień leży na płaszczyźnie. Brak przecięcia.
- ▶ Jeśli $a \neq 0$, to promień i płaszczyzna nie są równoległe, a zatem posiadają jeden punkt przecięcia $P(s_l)$.



- ▶ W punkcie przecięcia wektor

$$P(s) - V_0 = (P_0 - V_0) + su = w + su$$

jest prostopadły do wektora n . Zatem

$$n \cdot (w + s_I u) = 0.$$

Rozwiązując to równanie otrzymujemy

$$s_I = \frac{-n \cdot w}{n \cdot u} = \frac{-n \cdot w}{a}$$

- ▶ Jeśli $s_I < 0$, to przecięcie nastąpiło w kierunku odwrotnym do danego wektorem u i uznajemy, że nie nastąpiło przecięcie.

Kwaterniony – przypomnienie

Kwaternion jest to liczba postaci:

$$q = s + ia + jb + kc,$$

gdzie $s, a, b, c \in \mathbb{R}$ oraz $i^2 = j^2 = k^2 = ijk = -1$

$$ij = k, \quad jk = i, \quad ki = j,$$

$$ji = -k, \quad kj = -i, \quad ik = -j$$

Zbiór kwaternionów oznaczamy przez \mathbb{H} .

Kwaternion możemy zapisać w równoważnej postaci:

$$q = [s, v],$$

gdzie $s \in \mathbb{R}$, $v \in \mathbb{R}^3$.

Podstawowym działaniem na kwaternionach jakie będziemy potrzebować to mnożenie:

$$[s_a, a][s_b, b] = [s_a s_b - a \cdot b, s_a b + s_b a + a \times b]$$

Mnożenie kwaternionów nie jest przemienne!!

Kwaternion sprzężony do kwaternionu $q = [s, \mathbf{v}]$:

$$q^* = [s, -\mathbf{v}]$$

Norma kwaternionu $q = [s, \mathbf{v}]$:

$$|q| = \sqrt{s^2 + \|\mathbf{v}\|^2}.$$

Kwaternion q' nazywamy znormalizowanym, gdy

$$q' = \frac{q}{|q|}$$

Kwaternion odwrotny do q :

$$q^{-1} = \frac{q^*}{|q|^2}$$

Rotację o kąt θ wokół unormowanego wektora \hat{v} możemy za pomocą kwaternionów wyrazić następująco:

$$qpq^{-1},$$

gdzie

$$q = [\cos \frac{1}{2}\theta, \sin \frac{1}{2}\theta\hat{v}],$$

orz p to czysty kwaternion przechowujący wektor p , tzn.

$$p = [0, p].$$

Składanie obrotów sprowadza się do mnożenia kwaternionów, tzn. jeśli kwaterniony q_1 i q_2 reprezentują obroty to kwaternion q_2q_1 reprezentuje złożenie tych obrotów.

Niech $q = [s, v] = [s, xi + yj + zk]$ i $|q| = 1$ oraz

$p = [0, p] = [0, x_p i + y_p j + z_p k]$.

Kwaternion qpq^{-1} można zapisać w postaci macierzowej:

$$qpq^{-1} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 - 2(y^2 + z^2) & 2(xy - sz) & 2(xz + sy) \\ 0 & 2(xy + sz) & 1 - 2(x^2 + z^2) & 2(yz - sx) \\ 0 & 2(xz - sy) & 2(yz + sx) & 1 - 2(x^2 + y^2) \end{bmatrix} \begin{bmatrix} 0 \\ x_p \\ y_p \\ z_p \end{bmatrix}$$

Niech $q = [s, v] = [s, xi + yj + zk]$ i $|q| = 1$ oraz

$p = [0, p] = [0, x_p i + y_p j + z_p k]$.

Kwaternion qpq^{-1} można zapisać w postaci macierzowej:

$$qpq^{-1} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 - 2(y^2 + z^2) & 2(xy - sz) & 2(xz + sy) \\ 0 & 2(xy + sz) & 1 - 2(x^2 + z^2) & 2(yz - sx) \\ 0 & 2(xz - sy) & 2(yz + sx) & 1 - 2(x^2 + y^2) \end{bmatrix} \begin{bmatrix} 0 \\ x_p \\ y_p \\ z_p \end{bmatrix}$$

Macierz

$$\begin{bmatrix} 1 - 2(y^2 + z^2) & 2(xy - sz) & 2(xz + sy) \\ 2(xy + sz) & 1 - 2(x^2 + z^2) & 2(yz - sx) \\ 2(xz - sy) & 2(yz + sx) & 1 - 2(x^2 + y^2) \end{bmatrix}$$

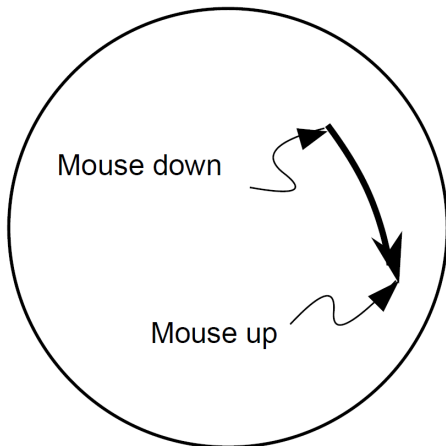
reprezentuje macierz obrotu.

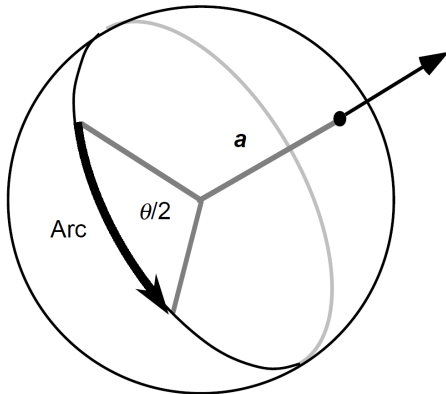
Rotacja sferyczna

Do reprezentacji rotacji sferycznej użyjemy ArcBall przedstawionego przez K.Shoemake'a w 1992 roku.

Rotacja sferyczna

Do reprezentacji rotacji sferycznej użyjemy ArcBall przedstawionego przez K.Shoemake'a w 1992 roku.





Nasza sfera będzie miała promień 1 i środek w środku okna programu.

Na początku ustawiamy macierz obrotu na identyczność.

Każdy punkt na ekranie (x, y) , który wybierzemy klawiszem myszki lub ciągnąc myszkę z wciśniętym klawiszem będzie najpierw przekształcany do kwadratu $[-1, 1]^2$, oznaczmy taki punkt przez (x_k, y_k) . Następnie współrzędna z obliczana jest następująco:

- ▶ $r = x_k^2 + y_k^2$,
- ▶ jeśli $r > 1$, to
 - ▶ $s = 1/\sqrt{r}$
 - ▶ $x_k = sx_k$,
 - ▶ $y_k = sy_k$,
 - ▶ $z_k = 0$,
- ▶ w przeciwnym przypadku $z = \sqrt{1 - r}$.

Mając wyznaczony punkt początkowy i końcowy łuku, a tym samym ich wektory p_0 , p_1 tworzymy kwaternion reprezentujący obrót:

$$q = [p_0 \cdot p_1, p_0 \times p_1]$$

Mając wyznaczony punkt początkowy i końcowy łuku, a tym samym ich wektory p_0 , p_1 tworzymy kwaternion reprezentujący obrót:

$$q = [p_0 \cdot p_1, p_0 \times p_1]$$

Następnie kwaternion q przekształcamy w macierz obrotu, którą mnożymy przez bieżącą macierz obrotu.