Politechnika Wrocławska

Grafika komputerowa i komunikacja człowiek-komputer

Laboratorium nr 4

Interakcja z użytkownikiem, transformacje wierzchołków

Szymon Datko

szymon.datko@pwr.edu.pl

Wydział Elektroniki,

Politechnika Wrocławska

semestr zimowy 2020/2021

HR EXCELLENCE IN RESEARCH

Politechnika Wrocławska

Cel ćwiczenia

1. Zapoznanie się z mechanizmem obsługi urządzeń peryferyjnych.
2. Zrozumienie zasady działania przekształceń wierzchołków.
3. Implementacja transformacji w reakcji na działania użytkownika.

Politechnika Wrocławska

Obsługa zdarzeń z klawiatury/myszy

- Zagadnienie to nie jest częścią specyfikacji OpenGL.
- W naszym programie całą obsługą zdarzeń zajmuje się biblioteka GLFW.
- Zasadniczo chodzi o to, aby:
  - zdefiniować funkcję do wykonania w reakcji na zdarzenie,
    - musi ona przyjmować odpowiednią liczbę i rodzaj argumentów,
  - uaktywnić ją – podpiąć do określonego rodzaju zdarzenia,
  - stosowany jest tu mechanizm wywołań zwrotnych (ang. *callback*).
- Funkcja bezpośredniej reakcji na zdarzenie powinna być możliwie szybka,
  - często rejestruje się same akcje, a obsługę realizuje osobny proces.
- Więcej informacji: `https://www.glfw.org/docs/latest/input_guide.html`.

Współrzędne jednorodne

- Podstawowym pojęciem w grafice komputerowej **wierzchołek**. Jest to punkt z określonym położeniem w przestrzeni 3D.
- W **reprezentacji jednorodnej** do zapisu położenia stosuje się wektory o liczbie elementów większej niż wymiar przestrzeni,

$$\begin{bmatrix} x & y & z & w \end{bmatrix} \quad \leftrightarrow \quad \left(\frac{x}{w}, \frac{y}{w}, \frac{z}{w} \right).$$

- Typowo ostatnia składowa w dla położenia ma wartość 1.
- Zastosowanie takiej reprezentacji pozwala na bardzo łatwą realizację podstawowych przekształceń geometrycznych.

---

ang. *homogeneous coordinates* – August Möbius, „Der barycentrische Calcül", 1827.

Podstawowe rodzaje przekształceń

- Afiniczne[1]:
  - identyczność,
  - skalowanie,
  - translacja,
  - obrót.
- Nieafiniczne:
  - rzutowanie ortogonalne,
  - rzutowanie perspektywiczne.

[1] czyli takie, które zachowują (w ogólności) proprorcje odległości między punktami na tej samej linii przed i po wykonaniu transformacji, ale nie muszą zachować położenia punktu początkowego; może być zareprezentowane jako iloczyn macierzy $n \times n$ i wektora $n \times 1$ z dodatkowym przesunięciem o wektor; n - wymiar przestrzeni.

Macierz przekształceń – identyczność

Macierz $\mathbf{I}$ jest także nazywana macierzą jednostkową.

$$\mathbf{I} = \begin{bmatrix} 1.0 & 0.0 & 0.0 & 0.0 \\ 0.0 & 1.0 & 0.0 & 0.0 \\ 0.0 & 0.0 & 1.0 & 0.0 \\ 0.0 & 0.0 & 0.0 & 1.0 \end{bmatrix}$$

Z reguły jest to domyślna macierz, od której rozpoczynamy wszelkie dalsze przekształcenia lub inne obliczenia.

- Przykład obliczeń:

$$\begin{bmatrix} 1.0 & 0.0 & 0.0 & 0.0 \\ 0.0 & 1.0 & 0.0 & 0.0 \\ 0.0 & 0.0 & 1.0 & 0.0 \\ 0.0 & 0.0 & 0.0 & 1.0 \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ z \\ 1.0 \end{bmatrix} = \begin{bmatrix} x & y & z & 1.0 \end{bmatrix}$$

Macierz przekształceń – skalowanie

$$\mathbf{S} = \begin{bmatrix} S_x & 0.0 & 0.0 & 0.0 \\ 0.0 & S_y & 0.0 & 0.0 \\ 0.0 & 0.0 & S_z & 0.0 \\ 0.0 & 0.0 & 0.0 & 1.0 \end{bmatrix}$$

- Gdy $S_x = S_y = S_z$ to mówimy o skalowaniu jednorodnym.
- Przykład obliczeń:

$$\begin{bmatrix} S_x & 0.0 & 0.0 & 0.0 \\ 0.0 & S_y & 0.0 & 0.0 \\ 0.0 & 0.0 & S_z & 0.0 \\ 0.0 & 0.0 & 0.0 & 1.0 \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ z \\ 1.0 \end{bmatrix} = \begin{bmatrix} S_x \cdot x & S_y \cdot y & S_z \cdot z & 1.0 \end{bmatrix}$$

Macierz przekształceń – translacja

$$\mathbf{T} = \begin{bmatrix} 1.0 & 0.0 & 0.0 & T_x \\ 0.0 & 1.0 & 0.0 & T_y \\ 0.0 & 0.0 & 1.0 & T_z \\ 0.0 & 0.0 & 0.0 & 1.0 \end{bmatrix}$$

- Reprezentacja jednorodna punktu pozwala wyrazić operację przez macierz.
- Przykład obliczeń:

$$\begin{bmatrix} 1.0 & 0.0 & 0.0 & T_x \\ 0.0 & 1.0 & 0.0 & T_y \\ 0.0 & 0.0 & 1.0 & T_z \\ 0.0 & 0.0 & 0.0 & 1.0 \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ z \\ 1.0 \end{bmatrix} = \begin{bmatrix} x + T_x & y + T_y & z + T_z & 1.0 \end{bmatrix}$$

Macierz przekształceń – obrót wokół osi x

$$\mathbf{R}_{\mathbf{x},\gamma} = \begin{bmatrix} 1.0 & 0.0 & 0.0 & 0.0 \\ 0.0 & \cos\gamma & -\sin\gamma & 0.0 \\ 0.0 & \sin\gamma & \cos\gamma & 0.0 \\ 0.0 & 0.0 & 0.0 & 1.0 \end{bmatrix}$$

- Współrzędne w osi wokół której następuje obrót nie ulegają zmianie!
- Przykład obliczeń:

$$\begin{bmatrix} 1.0 & 0.0 & 0.0 & 0.0 \\ 0.0 & \cos\frac{\pi}{2} & -\sin\frac{\pi}{2} & 0.0 \\ 0.0 & \sin\frac{\pi}{2} & \cos\frac{\pi}{2} & 0.0 \\ 0.0 & 0.0 & 0.0 & 1.0 \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ z \\ 1.0 \end{bmatrix} = \begin{bmatrix} x & -z & y & 1.0 \end{bmatrix}$$

Macierz przekształceń – obrót wokół osi y i z

$$\mathbf{R}_{\mathbf{y},\theta} = \begin{bmatrix} \cos\theta & 0.0 & \sin\theta & 0.0 \\ 0.0 & 1.0 & 0.0 & 0.0 \\ -\sin\theta & 0.0 & \cos\theta & 0.0 \\ 0.0 & 0.0 & 0.0 & 1.0 \end{bmatrix}$$

$$\mathbf{R}_{\mathbf{z},\phi} = \begin{bmatrix} \cos\phi & -\sin\phi & 0.0 & 0.0 \\ \sin\phi & \cos\phi & 0.0 & 0.0 \\ 0.0 & 0.0 & 1.0 & 0.0 \\ 0.0 & 0.0 & 0.0 & 1.0 \end{bmatrix}$$

Macierz przekształceń – obrót wokół wektora

Obrót o kąt skierowany ϕ (reguła prawej dłoni) wokół wektora $\mathbf{v}$.

$$\mathbf{v}^T = \begin{bmatrix} v_x & v_y & v_z & 0.0 \end{bmatrix}, \quad \|\mathbf{v}\| = 1$$

$$\begin{bmatrix} v_x^2(1-\cos\phi)+\cos\phi & v_x v_y(1-\cos\phi)-v_z\sin\phi & v_x v_z(1-\cos\phi)+v_y\sin\phi & 0.0 \\ v_x v_y(1-\cos\phi)+v_z\sin\phi & v_y^2(1-\cos\phi)+\cos\phi & v_y v_z(1-\cos\phi)-v_x\sin\phi & 0.0 \\ v_x v_z(1-\cos\phi)-v_y\sin\phi & v_y v_z(1-\cos\phi)+v_x\sin\phi & v_z^2(1-\cos\phi)+\cos\phi & 0.0 \\ 0.0 & 0.0 & 0.0 & 1.0 \end{bmatrix}$$

Lewa górna część (3x3) powyższej macierzy może zostać wyrażona w prostszy sposób:

$$\mathbf{v}\,\mathbf{v}^\mathbf{T}\,(1-\cos\phi) \;+\; \mathbf{I}\cos\phi \;+\; \begin{bmatrix} 0.0 & -v_z & v_y \\ v_z & 0.0 & -v_x \\ -v_y & v_x & 0.0 \end{bmatrix} \sin\phi$$

Uwaga!

Wektor $\mathbf{v}$ określa oś obrotu, przechodzącą przez początek układu współrzędnych!

Macierz patrzenia

Chcemy opisać położenie kamery jednoznacznie w przestrzeni:

- współrzędne położenia kamery $\mathbf{e} = (e_x, e_y, e_z)$,
- współrzędne punktu zainteresowania $\mathbf{p} = (p_x, p_y, p_z)$,
- wektor orientacji, wskazujący górę dla kamery $\mathbf{u} = (u_x, u_y, u_z)$.

Kierunek patrzenia można opisać wektorem $\mathbf{f} = \frac{\mathbf{p}-\mathbf{e}}{\|\mathbf{p}-\mathbf{e}\|}$; $\mathbf{s} = (\mathbf{f} \times \mathbf{u})$.

Rzut wektora $\mathbf{u}$ na płaszczyznę prostopadłą do $\mathbf{f}$ to $\mathbf{u}' = (\mathbf{f} \times \mathbf{u}) \times \mathbf{f}$.

$$\mathbf{M_{lookAt}} = \begin{bmatrix} s_x & u'_x & f_x & -e_x \\ s_y & u'_y & f_y & -e_y \\ s_z & u'_z & f_z & -e_z \\ 0.0 & 0.0 & 0.0 & 1.0 \end{bmatrix}$$

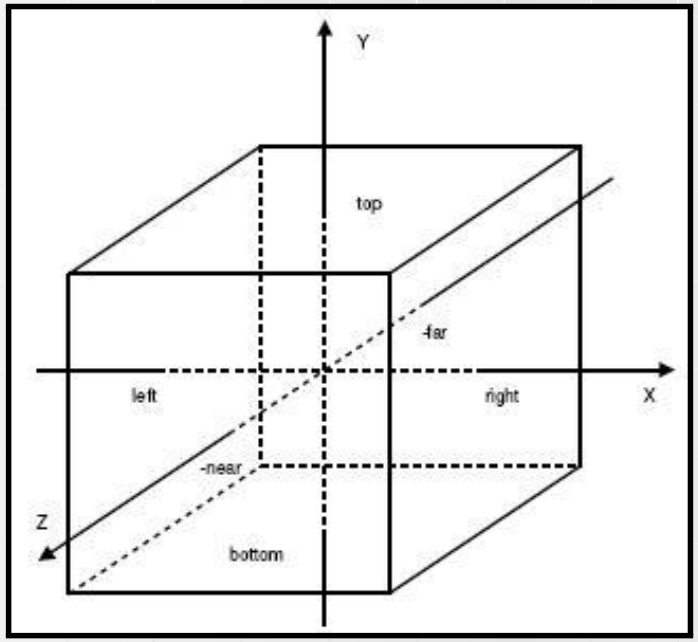
Oznaczenia: s – side vector, u – up vector, f – forward vector, e – eye position.

Więcej: https://www.khronos.org/registry/OpenGL-Refpages/gl2.1/xhtml/gluLookAt.xml

Macierz rzutowania prostopadłego

$$\mathbf{P}_{\mathbf{ortho}} = \begin{bmatrix} \frac{2}{right-left} & 0.0 & 0.0 & -\frac{right+left}{right-left} \\ 0.0 & \frac{2}{top-bottom} & 0.0 & -\frac{top+bottom}{top-bottom} \\ 0.0 & 0.0 & -\frac{2}{far-near} & -\frac{far+near}{far-near} \\ 0.0 & 0.0 & 0.0 & 1.0 \end{bmatrix}$$

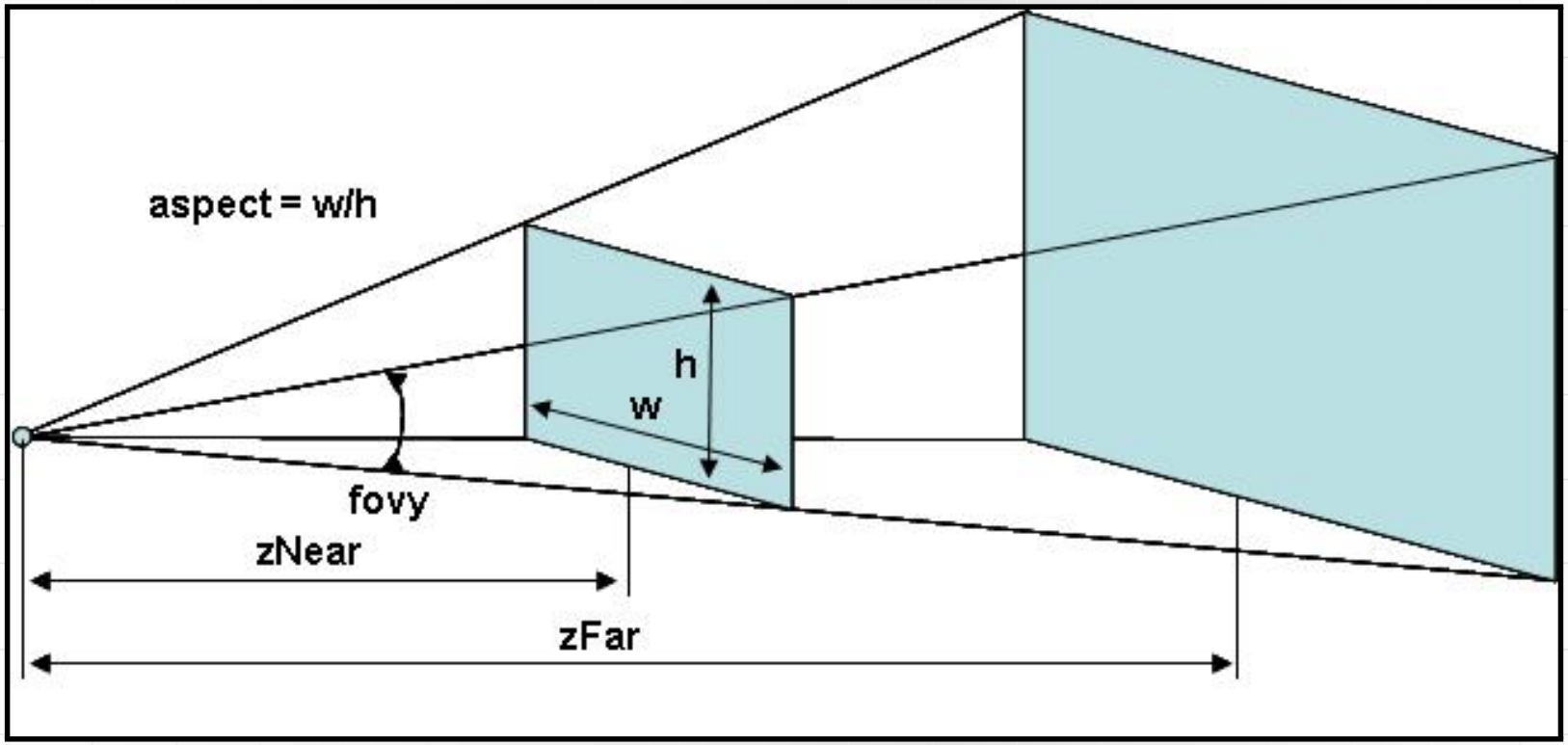
Macierz nie opisuje wprost przekształcenia nieafinicznego, a bryłę widzenia!



Źródło: http://cpp0x.pl/kursy/Kurs-OpenGL-C++/Definiowanie-sceny-3D/104

Rzutowanie perspektywiczne

Obszar pomiędzy `zNear` a `zFar` to tak zwana bryła widzenia.



Przy rzutowaniu perspektywicznym to tak zwany stożek ścięty (ang. *frustum*).

Źródło: http://www.zsk.ict.pwr.wroc.pl/zsk/dyd/intinz/gk/lab/cw_4_dz/

Macierz rzutowania perspektywicznego

$$\mathbf{P}_{\mathbf{frustum}} = \begin{bmatrix} \frac{2 \cdot near}{right-left} & 0.0 & \frac{right+left}{right-left} & 0.0 \\ 0.0 & \frac{2 \cdot near}{top-bottom} & \frac{top+bottom}{top-bottom} & 0.0 \\ 0.0 & 0.0 & -\frac{far+near}{far-near} & -\frac{2 \cdot far \cdot near}{far-near} \\ 0.0 & 0.0 & -1.0 & 0.0 \end{bmatrix}$$

W szczególności to przekształcenie może być zadane przez:

- pole widzenia jako kąt *fovy*,
- proporcji obrazu $aspect = \frac{width}{height}$,
- odległości bliższej *near* i dalszej *far* przestrzeni przycięcia;
- wtedy obliczamy $top = near * \tan(fovy \cdot \frac{\pi}{360})$ oraz $bottom = -top$, $right = top * aspect$, $left = -right$.

Politechnika Wrocławska

Realizacja w Legacy OpenGL (1/3)

- Zasadniczo prawdziwe jest następujące wyrażenie:

$$\vec{p}_{wyjściowe} = M_{transformacji} \cdot \vec{p}_{wejściowe}$$

- Współrzędne wyjściowe to faktycznie rysowane położenie wierzchołka.
- Współrzędne wejściowe stanowią argumenty funkcji `glVertex()`.
- Zmianę macierzy transformacji uzyskuje się za pomocą szeregu funkcji...

Realizacja w Legacy OpenGL (2/3)

- Identyczność / cofnięcie wszystkich przekształceń:

```
glLoadIdentity()
```

- Skalowanie:

```
glScalef(S_x, S_y, S_z)
```

- Translacja / przesunięcie o wektor:

```
glTranslatef(T_x, T_y, T_z)
```

- Obrót wokół osi X:

```
glRotatef(angle, 1.0, 0.0, 0.0)
```

- Obrót wokół osi Y:

```
glRotatef(angle, 0.0, 1.0, 0.0)
```

- Obrót wokół osi Z:

```
glRotatef(angle, 0.0, 0.0, 1.0)
```

Kąt `angle` wyrażony jest w stopniach! (Co jest raczej nietypowe w informatyce...)

Politechnika Wrocławska

Realizacja w Legacy OpenGL (3/3)

- Obrót wokół osi wyznaczonej przez wektor $[V_x, V_y, V_z]$ i punkt $[0, 0, 0]$:

```
glRotatef(angle, V_x, V_y, V_z)
```

- Przekształcenie patrzenia / przemieszczenie kamery na scenie:

```
gluLookAt(eyeX, eyeY, eyeZ, centerX, centerY, centerZ, upX, upY, upZ)
```

- Rzutowanie ortogonalne:

```
glOrtho(left, right, bottom, top, zNear, zFar)
```

- Rzutowanie perspektywiczne:

```
gluPerspective(fovy, aspect, zNear, zFar)
```

- Określenie rozmiaru rzutni w pikselach:

```
glViewport(x, y, width, height)
```

Nowości w przykładowym programie (1/3)

- Wprowadzono szereg zmiennych pomocniczych.

```
1| viewer = [0.0, 0.0, 10.0]
2|
3| theta = 0.0
4| pix2angle = 1.0
5|
6| left_mouse_button_pressed = 0
7| mouse_x_pos_old = 0
8| delta_x = 0
```

  - Zmienna `viewer` przechowuje informacje o położeniu obserwatora.
  - Zmienna `theta` zawiera wartość kąta obrotu.
  - Zmienna `pix2angle` to czynnik skalujący na potrzeby obliczeń,
    - żeby maksymalny ruch myszą odpowiadał obrotowi o 360°.
  - Zmienna `left_mouse_button_pressed` zawiera stan przycisku myszy.
  - Zmienna `mouse_x_pos_old` przechowuje ostatnie położenie w poziomie.
  - Zmienna `delta_x` zawiera informację o różnicy położeń myszy.

Nowości w przykładowym programie (2/3)

- Dodano funkcje związane z obsługą zdarzeń klawiatury i myszy.

```
def keyboard_key_callback(window, key, scancode, action, mods):
    if key == GLFW_KEY_ESCAPE and action == GLFW_PRESS:
        glfwSetWindowShouldClose(window, GLFW_TRUE)
```

```
def mouse_motion_callback(window, x_pos, y_pos):
    global delta_x
    global mouse_x_pos_old

    delta_x = x_pos - mouse_x_pos_old
    mouse_x_pos_old = x_pos
```

```
def mouse_button_callback(window, button, action, mods):
    global left_mouse_button_pressed

    if button == GLFW_MOUSE_BUTTON_LEFT and action == GLFW_PRESS:
        left_mouse_button_pressed = 1
    else:
        left_mouse_button_pressed = 0
```

- W funkcji `main()` dodano wywołania do obsługi zdarzeń.

```
def main():
    # (...)
    glfwSetKeyCallback(window, keyboard_key_callback)
    glfwSetCursorPosCallback(window, mouse_motion_callback)
    glfwSetMouseButtonCallback(window, mouse_button_callback)
    # (...)
```

Nowości w przykładowym programie (3/3)

- Zmieniono parametry rzutowania – perspektywiczne w zakresie $[0.1; 300]$.
- Dodano funkcję `example_object()`, rysującą przykładowy model.
- W funkcji `render()` wykonano transformacje wierzchołków.

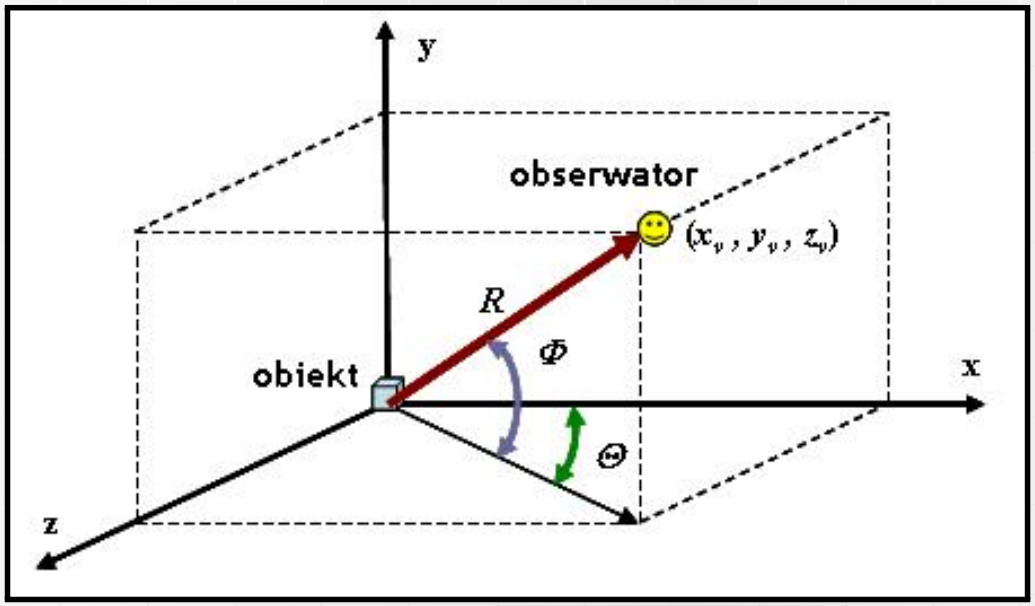
```
1  def render(time):
2      global theta
3
4      glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT)
5      glLoadIdentity()
6
7      gluLookAt(viewer[0], viewer[1], viewer[2],
8                0.0, 0.0, 0.0, 0.0, 1.0, 0.0)
9
10     if left_mouse_button_pressed:
11         theta += delta_x * pix2angle
12
13     glRotatef(theta, 0.0, 1.0, 0.0)
14
15     axes()
16     example_object()
17
18     glFlush()
```

Poruszanie kamerą dookoła obiektu

- Współrzędne kamery można określić za pomocą następujących równań,

$$x_{eye}(R,\theta,\phi) = R \cdot \cos(\theta) \cdot \cos(\phi),$$
$$y_{eye}(R,\theta,\phi) = R \cdot \sin(\phi),$$
$$z_{eye}(R,\theta,\phi) = R \cdot \sin(\theta) \cdot \cos(\phi).$$

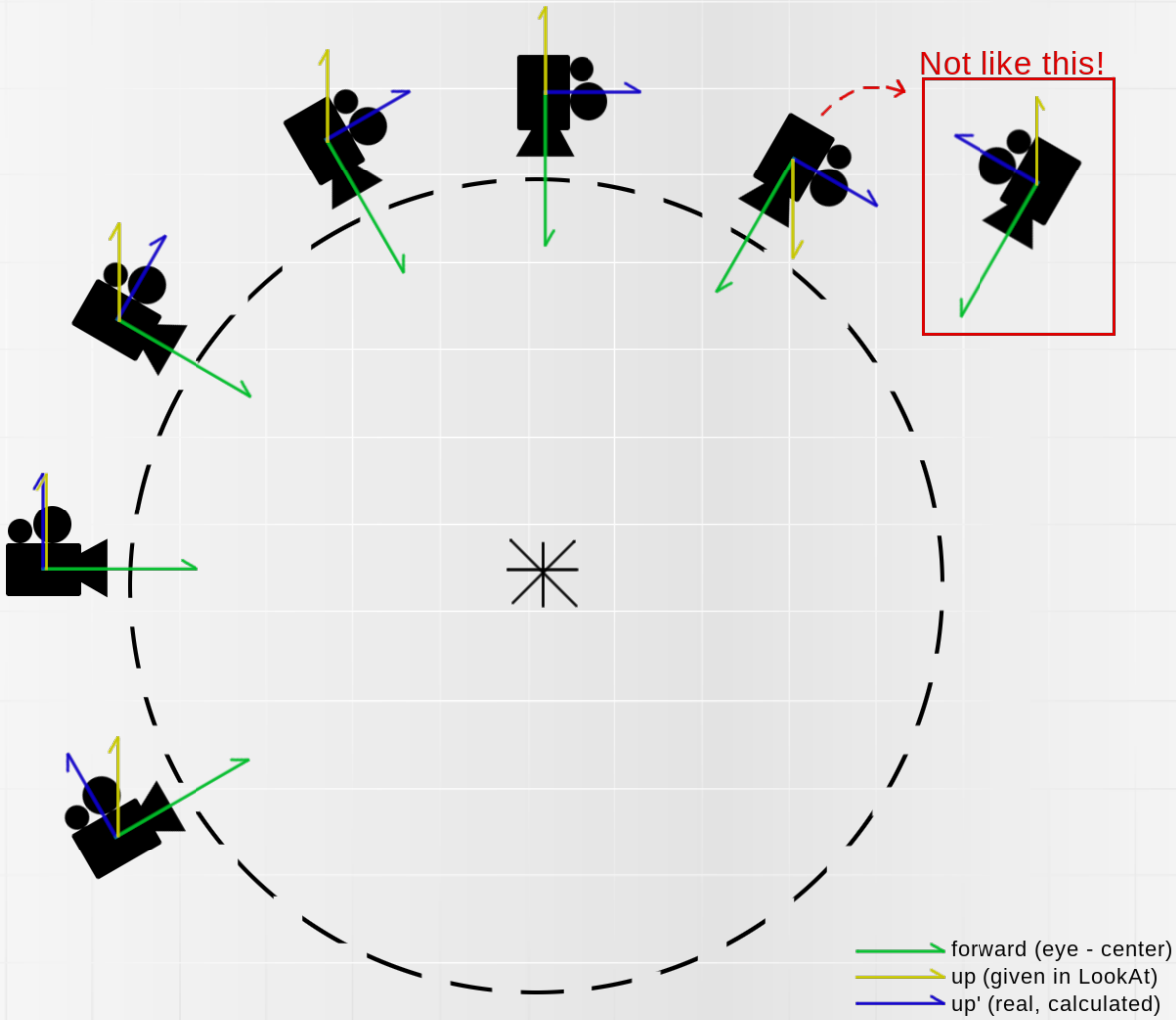
- Zakresy wartości kątów θ i ϕ to przedziały $0 \leq \theta \leq 2\pi$ oraz $0 \leq \phi \leq 2\pi$.
- Parametr θ to tak zwany kąt azymutu, zaś ϕ to tak zwany kąt elewacji.



Źródło obrazka: http://www.zsk.ict.pwr.wroc.pl/zsk/dyd/intinz/gk/lab/cw_4_dz/

Uwaga na wektor wskazujący górę

- Dla pewnego zakresu kąta elewacji, góra kamery powinna być odwrócona!



Politechnika Wrocławska

Koniec wprowadzenia.

Zadania do wykonania...

Zadania do wykonania (1)

Na ocenę **3.0** należy wprowadzić obracanie obiektu wokół osi X.

Wskazówki:

- przestudiować w jaki sposób zrealizowano obrót wokół osi Y o kąt `theta`,
- dodać zmienną pomocniczą `phi` i obsłużyć ruch myszą w pionie,
  - nadal bazować tylko na wciśniętym lewym przycisku myszy,
- intuicyjnie: obrót wokół osi X to obrót "góra-dół",
- wymagane będzie zmodyfikowanie funkcji:
  - `mouse_motion_callback()`,
  - `render()`.

Politechnika Wrocławska

Zadania do wykonania (2)

Na ocenę **3.5** należy wprowadzić obsługę drugiego przycisku myszy.

Wskazówki:

- obsługa nowego przycisku – w funkcji `mouse_button_callback()`,
- postać funkcji `mouse_motion_callback()` nie ulegnie zmianie,
- wprowadzić zmienną pomocniczą `scale`,
- w funkcji `render()`:
  - wykonać zmianę wartości zmiennej `scale`,
  - przy ruchu z wciśniętym prawym przyciskiem myszy,
- użyć wartości `scale` do przeskalowania obiektu – `glScalef()`.

Politechnika Wrocławska

Zadania do wykonania (3)

Na ocenę **4.0** należy zrealizować poruszanie kamerą wokół modelu.

Wskazówki:

- nie powinno być konieczności modyfikacji funkcji obsługujących zdarzenia,
- zakomentować funkcje `glRotatef()` i `glScalef()` w funkcji `render()`,
- wykorzystać wartości zmiennych `theta` i `phi` do obliczenia x_{eye}, y_{eye} i z_{eye},
- zdarzenia prawego przycisku myszy użyć do zmiany wartości parametru R,
- użyć wartości x_{eye}, y_{eye} i z_{eye} jako argumentów funkcji `gluLookAt()`,
- wartości *theta* i *phi* przeskalować w funkcjach *sin*() i *cos*() przez $\frac{\pi}{180}$.

Politechnika Wrocławska

Zadania do wykonania (4)

Na ocenę **4.5** należy usprawnić poruszanie kamerą wokół modelu.

Wskazówki:

- wprowadzić ograniczenia w zakresie przybliżania/oddalania kamery,
- zapewnić poprawność przejść kamery wokół modelu,
  - w szczególności zwrócić uwagę co dzieje się "nad" i "pod" obiektem,
  - pomoce może być użycie modułu do ograniczenia wartości kątów do przedziału $[0; 2\pi]$,
- wprowadzić możliwość przełączania między trybem obracania obiektu i trybem poruszania kamerą – na przykład za pomocą klawiatury.

Politechnika Wrocławska

Zadania do wykonania (5)

Na ocenę **5.0** należy zrealizować zadanie dodatkowe.

Wskazówka:

- wybrać jeden z przykładów zaproponowanych jako "zadania domowe",
  - dokument znajduje się na stronie prowadzącego,
  - uściślenie odnośnie Zadania 4.5 (Trójkat Sierpińskiego):
    - ruch ma działać analogicznie, jak w opisie poniżej,
- alternatywnie można wykonać swobodny pierwszoosobowy ruch kamery,
  - cel: ruch podobny do znanego z gier pierwszoosobowych,
  - wprowadzić zmienną pomocniczą dla kierunku patrzenia,
  - klawisze [w][s][a][d] posłużą do zmiany położenia obserwatora,
  - ruch myszką powinien wpływać na kierunek patrzenia,
  - punkt zainteresowania / kierunek patrzenia można wyznaczyć analogicznie do przypadku z kamerą poruszaną dookoła obiektu.