WebGL – Úvod do 2D a 3D
grafiky, transformácie, používateľské rozhranie a V/V prvky

doc. Ing. Branislav Sobota, PhD.
Ing. Marián Hudák, Ing. Lenka Bubeňková
Katedra počítačov a informatiky, FEI TU v Košiciach

C 03

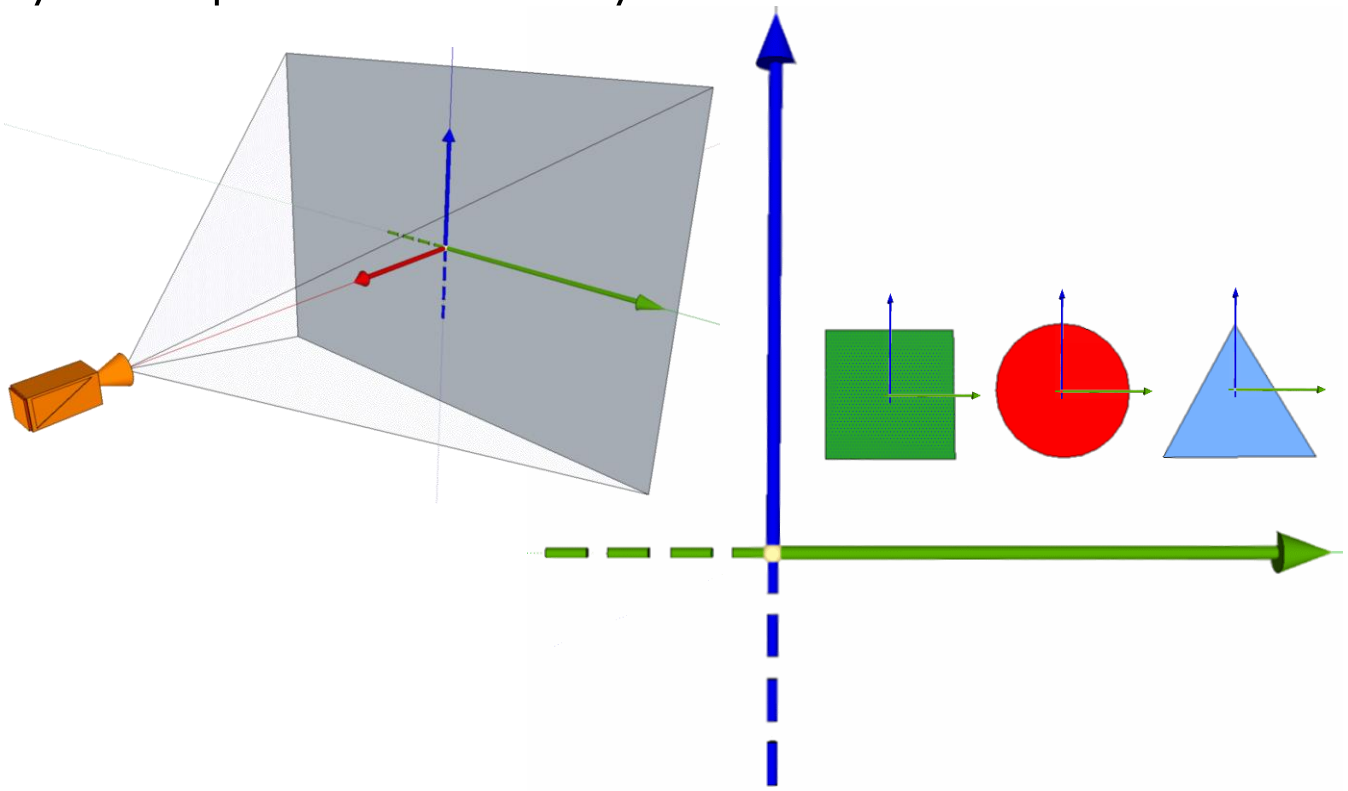
Ciele cvičenia

- WebGL - Pracovný priestor a objekty
- WebGL - Transformácie – matice, knižnica gl-matrix
- WebGL - Transformácia posunutia, otočenia a zmeny mierky
- WebGL – Základné používateľské rozhranie - príprava
- WebGL – Použitie bežca, tlačidla a ovládanie vstupom z klávesnice

1. WebGL - Pracovný priestor a objekty

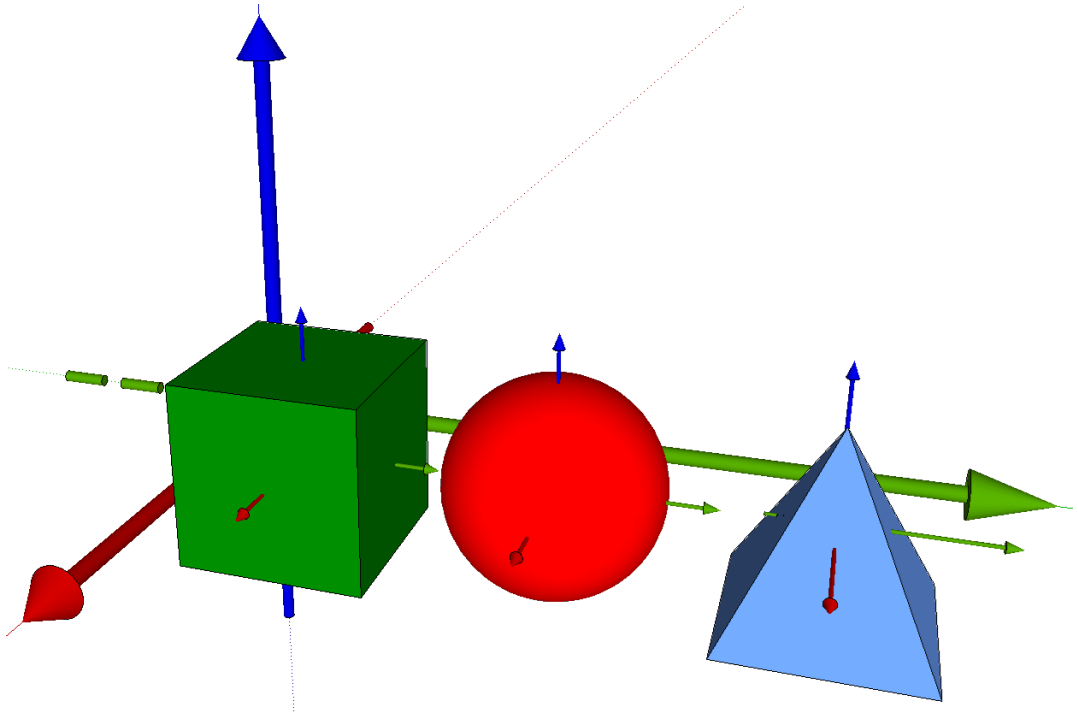
Pracovný priestor 2D

Základný pracovný priestor vo WebGL je definovaný ako na nasledovnom obrázku. V skutočnosti WebGL pracuje stále v 3D priestore a 2D priestor je vytvorený len fixáciou jednej roviny voči polohe kamery.



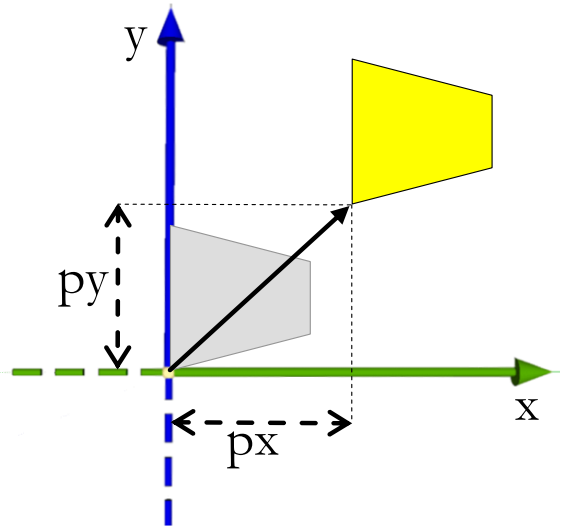
1. WebGL - Pracovný priestor a objekty
Pracovný priestor 3D

- Trojrozmerný priestor (3D) je štandardným pracovným priestorom WebGL, v ktorom sa uskutočňujú všetky transformácie. Základný pracovný 3D priestor (USS) je definovaný tak, ako ukazuje nasledujúci obrázok.



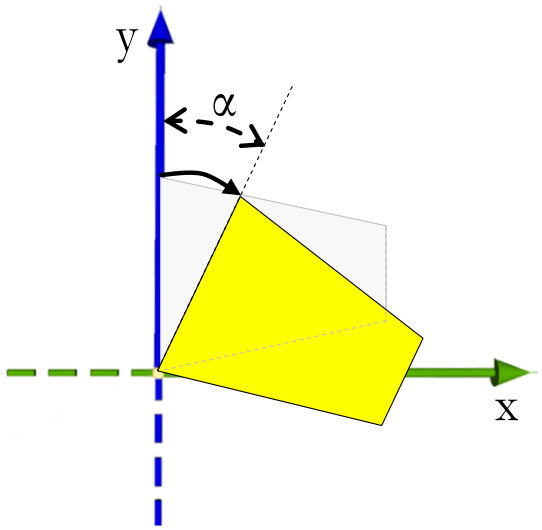
2. WebGL - Transformácie - matice
Transformačné matice 2D

Posunutie



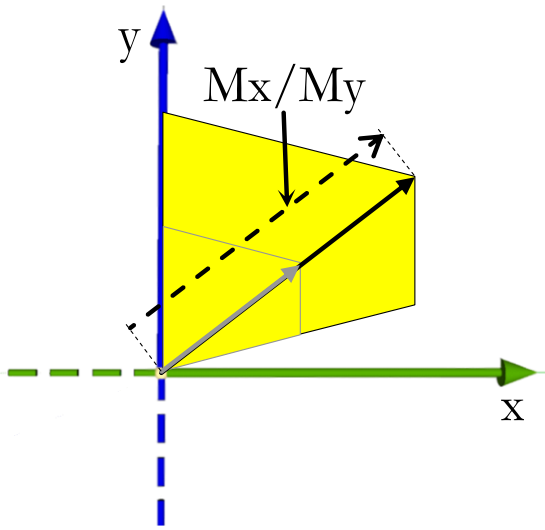
$$\mathbf{T}_P = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ px & py & 1 \end{bmatrix}$$

Rotácia



$$\mathbf{T}_O = \begin{bmatrix} \cos(\alpha) & \sin(\alpha) & 0 \\ -\sin(\alpha) & \cos(\alpha) & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

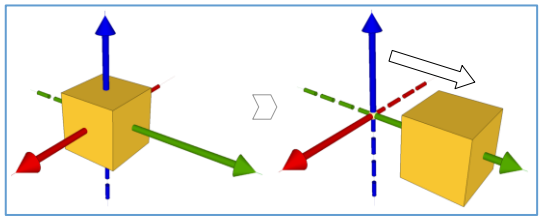
Zmena mierky



$$\mathbf{T}_M = \begin{bmatrix} M_x & 0 & 0 \\ 0 & M_y & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

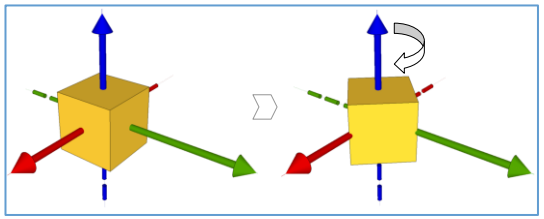
2. WebGL - Transformácie - matice
Transformačné matice 3D

Posunutie



$$\mathbf{T}_P = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ px & py & pz & 1 \end{bmatrix}$$

Rotácia

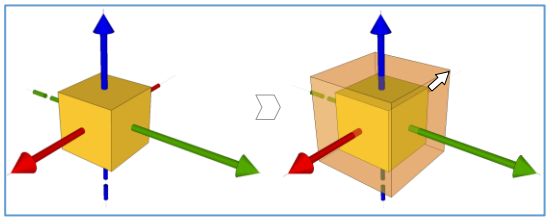


$$\mathbf{T}_{Ox} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\alpha & \sin\alpha & 0 \\ 0 & -\sin\alpha & \cos\alpha & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$\mathbf{T}_{Oz} = \begin{bmatrix} \cos\alpha & \sin\alpha & 0 & 0 \\ -\sin\alpha & \cos\alpha & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$\mathbf{T}_{Oy} = \begin{bmatrix} \cos\alpha & 0 & \sin\alpha & 0 \\ 0 & 1 & 0 & 0 \\ -\sin\alpha & 0 & \cos\alpha & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Zmena mierky



$$\mathbf{T}_M = \begin{bmatrix} M_x & 0 & 0 & 0 \\ 0 & M_y & 0 & 0 \\ 0 & 0 & M_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

2. WebGL - Knižnica gl-matrix
Knižnica pre podporu transformácií

- Stiahnite si balíček „**WebGL_Transformation_a_GUI.zip**“ z portálu Moodle predmetu Počítačová grafika. Obsah balíčka skopírujte do vášho projektu aby štruktúra vyzerala nasledovne:

1. Skontrolujte či adresár „`glmatrix`“ je skopírovaný priamo do adresára projektu s názvom „`js`“.
2. V súbore „index.html“ pridajte odkaz na skript knižnice :

```
<script src="js/glmatrix/gl-matrix.js"></script>
```

3. Okrem knižnice „`glmatrix`“ skontrolujte nakopírovanie príslušných `*.js` súborov v adresári „`js`“ nachádzajúceho sa vo vašom projekte.

2. WebGL – balíčky gl-matrix a jstranslation

- Štruktúra projektu po vložení obsahu z balíčka :

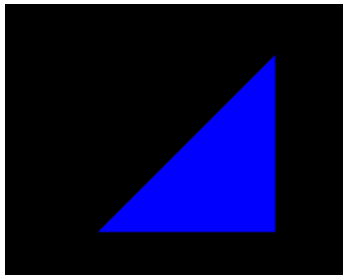
  - WebGL_getStart
    - > css
    - >> style.css
    - > js
    - >> glmatrix
    - >> TriangleMov.js
    - >> TriangleRot.js
    - >> TriangleScl.js
    - >> GUI_Cube_sliders.js
    - >> GUI_Cube_sliders_alter.js
    - models
    - texture
    - Index.html

3. WEBGL - TRANSFORMÁCIA POSUNU (TRANSLÁCIA)

- V súbore „**index.html**“ upravte odkaz na skript vizualizácie posunu trojuholníka :

```
<script src="js/TriangleMov.js"></script>
```

- Pozrite si nasledujúci riadok (č.208) v skripte **„TriangleMov.js“ :**
- `mat4.translate(modelViewMatrix,modelViewMatrix,[0.0, 0.0, -6.0]);`



- **Funkcia *translate* pracuje so vstupnými parametrami pre otáčanie v 3D :**
  - **modelViewMatrix** - počiatočná transformačná matica objektu mat4 (odkiaľ sa začína a použijú sa iniciálne hodnoty pre transformáciu)
  - **modelViewMatrix** - koncová transformačná matica objektu mat4 (kam sa zapisujú hodnoty po transformácii)
  - [x, y, z]; - súradnice posunu
- **Vstupné hodnoty môžu byť zadávané v intervale <-1.0, 1.0>, kde stred súradnicového systému je daný hodnotou 0.0**

3. WebGL - Transformácia posunu (translácia)

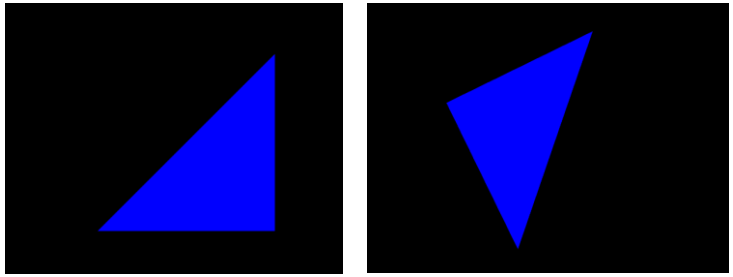
- **Úloha:** Vyskúšajte aplikovať posun v smeroch : *nahor*, *nadol*, *doľava*, *doprava*.

3. WebGL - Transformácia otočenia (rotácia)

- V súbore „**index.html**“ upravte odkaz na skript vizualizácie posunu trojuholníka :

```
<script src="js/TriangleRot.js"></script>
```

- Pozrite si nasledujúci riadok (č.211 - 214) v skripte **„TriangleRot.js“ :**
- `mat4.rotate(modelViewMatrix,modelViewMatrix, cubeRotation, [0, 0, 1]);`



- **Funkcia *translate* pracuje so vstupnými parametrami pre posun v 3D :**
  - **modelViewMatrix** - počiatočná transformačná matica objektu mat4 (odkiaľ sa začína a použijú sa iniciálne hodnoty pre transformáciu)
  - **modelViewMatrix** - koncová transformačná matica objektu mat4 (kam sa zapisujú hodnoty po transformácii)
  - [x, y, z]; - nastavenie rotácie pre príslušnú os

3. WebGL - Transformácia otočenia (rotácia)

- **Úloha:** Vyskúšajte aplikovať rotáciu postupne okolo jednotlivých ďalších osí.

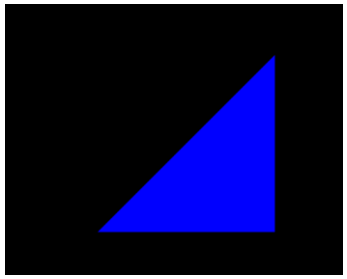
3. WebGL - Transformácia Zmeny mierky (Škálovanie)

- V súbore „**index.html**“ upravte odkaz na skript vizualizácie posunu trojuholníka :

```
<script src="js/TriangleScl.js"></script>
```

- Pozrite si nasledujúci riadok (č.212) v skripte **„TriangleScl.js“ :**
-

```
mat4.scale(modelViewMatrix,modelViewMatrix, [1, 1.5, 1]);
```



- **Funkcia *scale* pracuje so vstupnými parametrami pre škálovanie v 3D :**
  - **modelViewMatrix** - počiatočná transformačná matica objektu mat4 (odkiaľ sa začína a použijú sa iniciálne hodnoty pre transformáciu)
  - **modelViewMatrix** - koncová transformačná matica objektu mat4 (kam sa zapisujú hodnoty po transformácii)
  - [x, y, z]; - nastavenie veľkosti škálovania pre príslušnú os

3. WebGL - Transformácia Zmeny mierky (Škálovanie)

- **Úloha:** Do implementácie v riadku 212 upravte hodnoty metódy tak, aby ste objekt škálovali v smere každej osi.

4. WebGL – Základné používateľské rozhranie

- *WebGL* umožňuje ovládať vlastnosti objektov prostredníctvom používateľských rozhraní.
- Vstupy môžu byť riadené štandardným ovládaním pomocou klávesnice alebo myšky, prípadne kombináciou vstupných prvkov/elementov vytvorených v HTML.

4. WebGL – Základné používateľské rozhranie

1. Vo vývojárskom prostredí si otvorte súbor „**index.html**“ v ktorom prepíšete cestu skriptu :

```
<script src="js/GUI_Cube_sliders.js"></script>
```

Odkaz na knižnicu *glmatrix* nemažte,
tá bude potrebná pre prácu
s transformáciami !

5. WebGL –Použitie bežca (slider)

2. Do súboru vložte nasledujúce riadky.
   Vkladajte ich za ukončeným elementom canvas.

```
<link rel="stylesheet" type="text/css"
href="css/style.css">
<h3> Rotation Y </h3>
<div class="slidecontainer">
 <input type="range" min="0" max="1000" value="180"
class="slider" id="rotYrange"
oninput="sliderRyChange(this.value)" >
</div>
<h3> Translate X </h3>
<div class="slidecontainer">
 <input type="range" min="-1000" max="1000" value="0.0"
class="slider" id="translateXrange"
oninput="sliderTxChange(this.value)" >
</div>
```

5. WebGL –Použitie bežca (slider) -Získavanie údajov z elementu *bežec* (slider)

- **Úloha:** Precvičte prácu so získavaním údajov z elementu *bežec*.

1. Otvorte si javascript „**GUI_Cube_sliders.js**“ (**GUI_Cube_sliders_alter.js**).
2. Prezrite si metódy implementované pre získavanie údajov z elementu *slider*.

```
var sliderRyVal = 360; //nastavenie počiatočného uhla
......
function sliderRyChange(val){
//funkcia pre získavanie údajov z HTML elementu bežca
   sliderRyVal = val /150;
}
```

alebo **alter**nativa

```
var sliderRyVal = 360; //nastavenie počiatočného uhla
......
function sliderRyChange(val) {
    document.getElementById('rotYrange').innerHTML = val;
    sliderRyVal = document.getElementById('rotYrange').innerHTML /
150;
}
```

5. WebGL –Použitie bežca (slider) -Získavanie údajov z elementu bežec (slider)

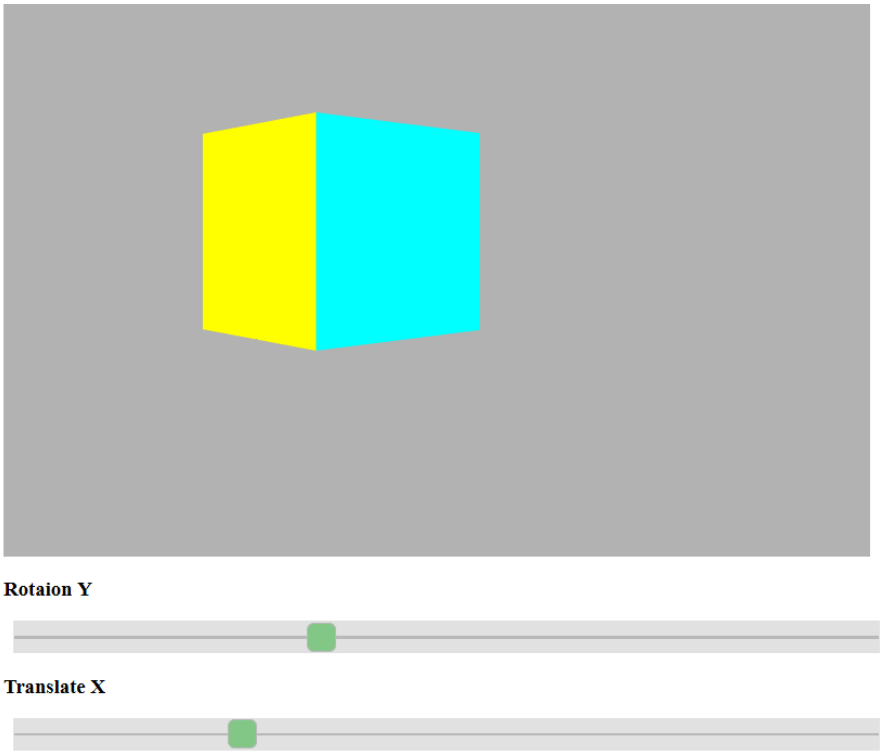
3. Následne si prezrite implementáciu riadkov 263-270.

```
mat4.rotate(modelViewMatrix, modelViewMatrix,
    sliderRyVal, // uhol otočenia podľa získaných udajov
    [0, 1, 0]);  // rotácia okolo osi Y
```

5. WebGL –Použitie bežca (slider) -Získavanie údajov z elementu bežec (slider)

4. Aktuálny stav uložte a spustite vizualizáciu v prostredí webového prehliadača.

   Posúvaním bežcov *Rotation Y* a *Translate X* dochádza k príslušným transformáciám nad zobrazenou kockou.



5. WebGL –Použitie tlačidla (button)

- **Úloha:** Precvičte pridanie elementu/ov *tlačidlo* (button) do skriptu.

1. Vo vývojárskom prostredí si otvore súbor „**index.html**“.
2. Do súboru vložte nasledujúce riadky. Vkladajte ich za ukončenou implementáciou elementov *bežcov* (slider).

```
<p></p>
<h3> Background Color </h3>
<button class="button" onclick="setBckColor(0.0, 1.0, 0.0)"
>Green</button>
<button class="button button2" onclick="setBckColor(0.0, 0.0,
1.0)">Blue</button>
<button class="button button3" onclick="setBckColor(1.0, 0.0,
0.0)">Red</button>
<button class="button button4" onclick="setBckColor(0.7, 0.7,
0.7)">Gray</button>
<button class="button button5" onclick="setBckColor(0.0, 0.0,
0.0)">Black</button>
```

5. WebGL –Použitie tlačidla (button)

- Pozrite si implementáciu v súbore **CSS/style.css** pre parametre *button* elementov.

5. WebGL –Použitie tlačidla (button)
Získavanie údajov z elementu *tlačidlo* (button)

- **Úloha:** Otvorte si javascript „**GUI_Cube_sliders.j**s“

1. Prezrite si metódy implementované pre získavanie údajov z elementu *button*.

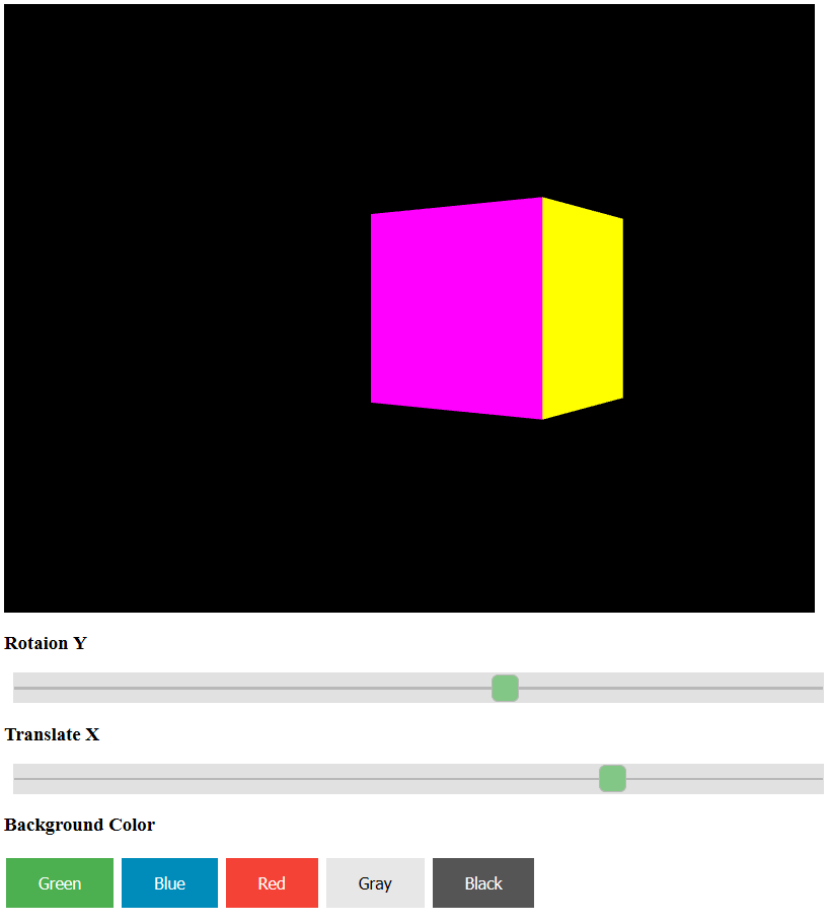
```
var r = 0.0; var g = 0.0; var b = 0.0;
//definovanie úvodných hodnôt RGB
...
function setBckColor(inr,ing,inb) {
r = inr; g = ing; b = inb;
}
```

2. Sledujte spôsob, ktorým je implementované získavanie údajov z elementu *tlačidlo* (button). Následne si prezrite implementáciu riadkov 4, 8 – 12 a riadok 226.

```
gl.clearColor(r, g, b, 1.0); //nastavenie farby pozadia
```

5. WebGL –Použitie tlačidla (button)
Získavanie údajov z elementu *tlačidlo* (button)

4. Aktuálny stav uložte a spustite vizualizáciu v prostredí webového prehliadača.



5. WebGL – Použitie ovládania vstupom z klávesnice

- **Úloha:** Otvorte si javascript „**GUI_Cube_sliders.js**“. Pre otestovanie vstupu po stlačení klávesu “r” vložte na koniec súboru „GUI_Cube_sliders.js“ nasledujúci fragment :

```
document.addEventListener('keypress', (event) => {
 const keyName = event.key;
 if(keyName == "r" )
  { r= 0.4;   }
//nastavenie hodnoty červenej zložky farby  (RED) na 0.4
});
```

- **Úloha:** Skript upravte tak, aby pri stlačení klávesov “g” a “b” sa vykonala zmena zelenej a modrej zložky na určitú nastavenú hodnotu, ktorú si určte samostatne.

DOPLŇUJÚCE ÚLOHY

- Do implementácie rotácie upravte hodnoty metódy tak, aby ste objektom dodatočne rotovali aj okolo osi x.
- Zmeňte použitie *bežca* (slider) „*Rotation Y*“ na rotáciu kocky okolo osi x.
- Pri stlačení klávesov “w”, “a”, “s” a “d” bude vykonaná zmena posunu (translácie) objektu kocky v definovaných osiach.

ÚLOHY NA SAMOSTATNÉ RIEŠENIE

- **Vytvorte zloženú transformáciu :**
  - *posunov* (translácií) objektu vo všetkých smeroch.
  - *otočení* (rotácií) objektu okolo každej osi,
    + vyskúšajte zmenu rýchlosti rotácie.
  - *zmeny mierky* objektu (škálovania) rôznu v smere každej osi.
- Použite *bežec* (slider) pre posun („***Translate X***“) pre nastavenie škálovania kocky vo všetkých osiach rovnako.
- Implementujte klávesy podľa vlastného výberu na vykonanie rotácií kocky okolo príslušných **osí *x*, *y*, *z***.
- Pokúste sa následne preimplementovať celý skript z ovládania **kocky na ovládanie ihlanu**.
- Po úspešnej preimplementácii na ihlan, sa pokúste ihlan vytvoriť farebne tak, aby každý vrchol mal inú farbu.

Q & A

branislav.sobota@tuke.sk
lenka.bubenkova@tuke.sk

Katedra počítačov a informatiky, FEI TU v Košiciach