THREE.JS – ÚVOD, TVORBA SCÉNY A VKLADANIE OBJEKTOV

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

C 05

Ciele cvičenia

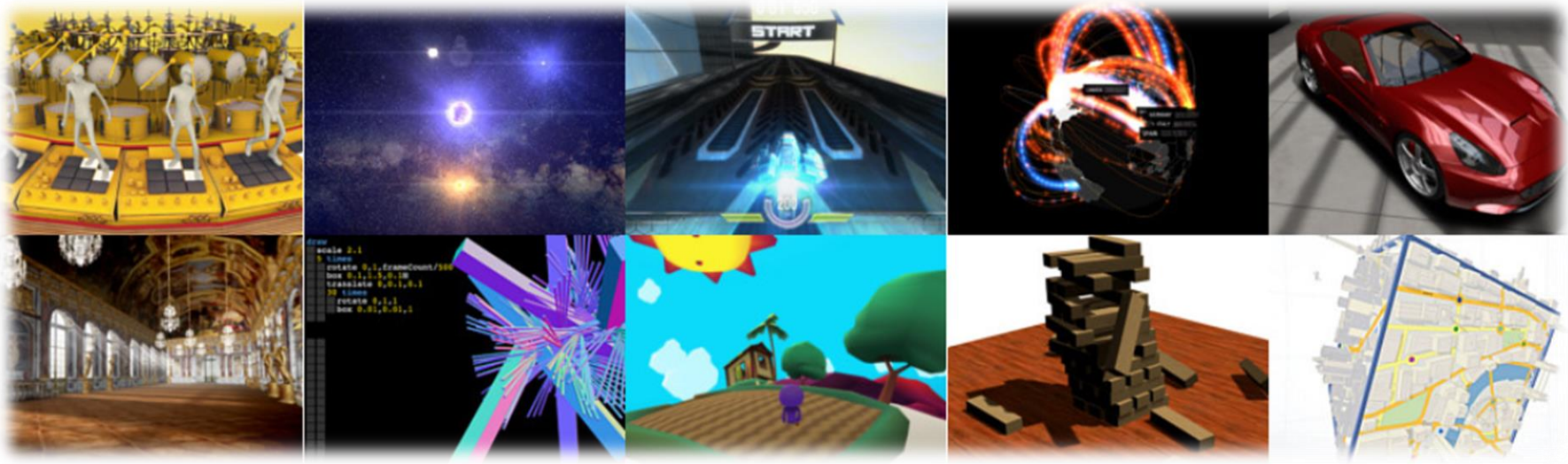
- Three.js - úvod a inštalácia balíčka
- Three.js - tvorba scény – komponenty, inicializácia a renderer
- Three.js - implementáciu 3D objektov
- Three.js – implementácia ovládania kamery - OrbitControls
- Three.js – implementácia okolitého prostredia – Skybox

1. *THREE.JS* – PRÍPRAVA BALÍČKA

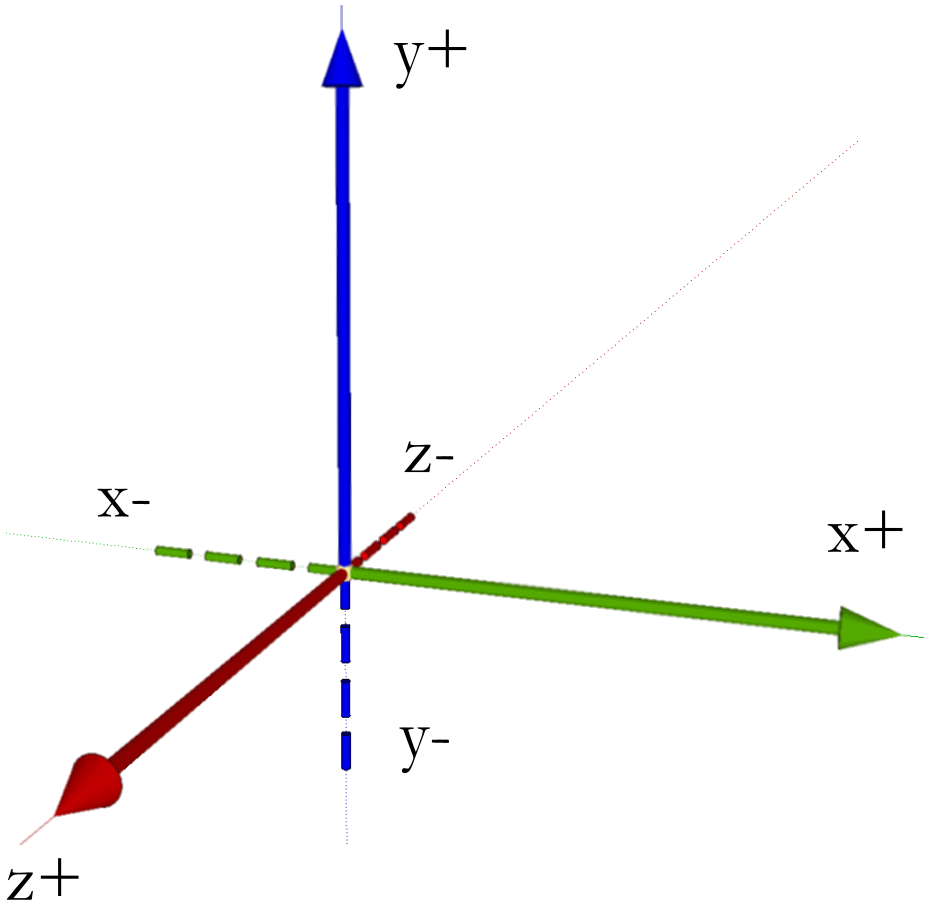
- **Úloha:** Stiahnite si balíček „**Threejs_Uvod_Scena_Objekty.zip**“ z portálu Moodle KPI a predmetu Počítačova grafika.
- Obsah balíčka skopírujte do vášho projektu aby štruktúra vyzerala nasledovne:
  - `WebGL_getStart`
    - `> css`
    - `> js`
    - `>> threejs`
    - `>> ThreeScene.js`
    - `models`
    - `> texture`
    - `>> box.jpg`
    - `>> lirkis.jpg`
    - `>> carbon.png`
    - `>> sky.jpg`
    - `index.html`

1. *THREE.JS –ÚVOD*

- *Three.js* je knižnica pre podporu implementácie 3D prostredí. Jej implementácia je v jazyku Javascript a tvorí pevné puto so systémom *WebGL*. Three.js má v súčasnosti široké uplatnenie a používa ho viacero systémov a aplikácií.

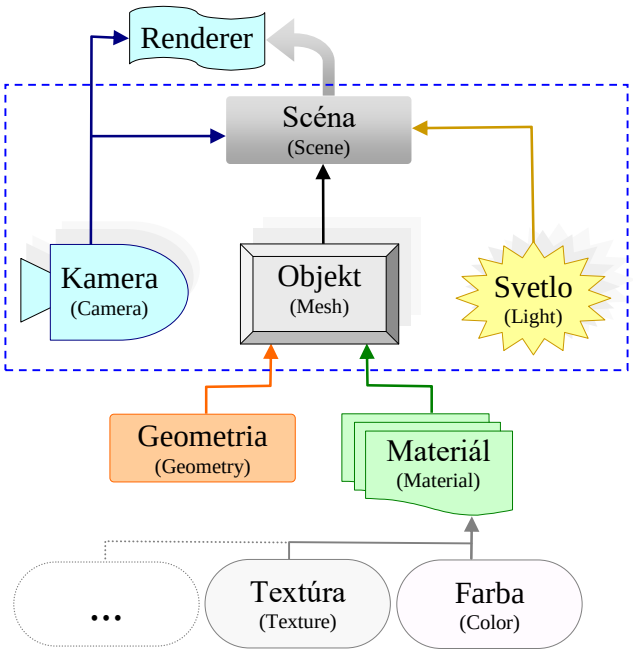


1. THREE.JS –SÚRADNICOVÝ SYSTÉM



2. THREE.JS – TVORBA SCÉNY – KOMPONENTY, INICIALIZÁCIA A RENDERER

- **Celková scéna v *Three.js* obsahuje nasledujúce komponenty:**
  - Renderer
  - Kamera
  - Scéna
  - Mesh (Objekt) = Geometria + Materiál
  - Svetlo - (implementácia bude vysvetlená neskoršie)



2. THREE.JS - IMPLEMENTÁCIA SCÉNY – INICIALIZÁCIA A RENDERER

- **Úloha:** Otvorte si skript „ThreeScene.js“ a html súbor „index.html“ vo vašom vývojovom prostredí.
- **Úloha:** inicializujte scénu a jej renderer pomocou *Three.js*

1. V skripte **„ThreeScene.js“** pridajte do metódy **init()** **implementáciu kamery:**

```
camera = new THREE.PerspectiveCamera(
                70,
                window.innerWidth / window.innerHeight,
                0.01,
                1000);
camera.position.set(0, 0, 5);
```

2. THREE.JS - IMPLEMENTÁCIA SCÉNY – INICIALIZÁCIA A RENDERER

2. Následne vytvorte v metóde **init()** *renderer*:

```
renderer = new THREE.WebGLRenderer({ antialias: true } );
renderer.setSize( window.innerWidth, window.innerHeight );
document.body.appendChild( renderer.domElement );
```

3. Na konci metódy **init()** vytvorte objekt scény :

```
scene = new THREE.Scene();
```

4. V skripte „**ThreeScene.js**“ pridajte do metódy **render()** implementáciu pre vykresľovanie pohľadu a scény:

```
requestAnimationFrame( render );
renderer.render( scene, camera );
camera.lookAt(scene.position);
```

Metóda „**requestAnimationFrame(render)**“ zavolá pri zmene každej snímky metódu render.
Pomocou tejto metódy je možné vytvárať vizualizačnú slučku.

3. THREE.JS - *IMPLEMENTÁCIA 3D OBJEKTOV*

- **Úloha:** Vložte do scény základnú „podlahovú“ plochu.

1. V skripte **„ThreeScene.js“** pridajte na konci metódu **addObjects()**
2. Pridajte do metódy **addObjects()** **implementáciu plochy:**

```
var geometryPlane = new THREE.PlaneGeometry( 10, 10, 4, 4 );
var materialPlane = new THREE.MeshBasicMaterial( {
                        color: 0x747570,
                        side: THREE.DoubleSide} );
plane = new THREE.Mesh( geometryPlane, materialPlane );
plane.position.set(0, -0.5, 0);
plane.rotation.x = Math.PI / 2;
scene.add( plane );
```

3. THREE.JS - *IMPLEMENTÁCIA 3D OBJEKTOV*

2. **Aby bola plocha `plane` v scéne viditeľná, zavolajte** v skripte „ThreeScene.js“ metódu **addObjects();**
   na konci metódy init();
3. **Otestujte správnosť implementácie.**
   Po zobrazení, by mal byť viditeľný výstup podobný ako na nasledujúcom obrázku.

3. THREE.JS - PRIDANIE OBJEKTOV KOCIEK DO SCÉNY

- **Úloha:** Pridajte do scény objekt kocky s textúrou.

1. V skripte **„ThreeScene.js“** pridajte do metódy **addObjects()** **implementáciu kocky cube1:**

```
var geometryCube = new THREE.BoxGeometry( 1, 1, 1 );
var cubeTexture = new THREE.ImageUtils.loadTexture('texture/box.jpg' );
var materialCube = new THREE.MeshBasicMaterial( {
    map: cubeTexture} );
cube1 = new THREE.Mesh( geometryCube, materialCube );
cube1.position.set(0, 0, 0);
scene.add( cube1 );
```

2. Otestujte implementáciu:

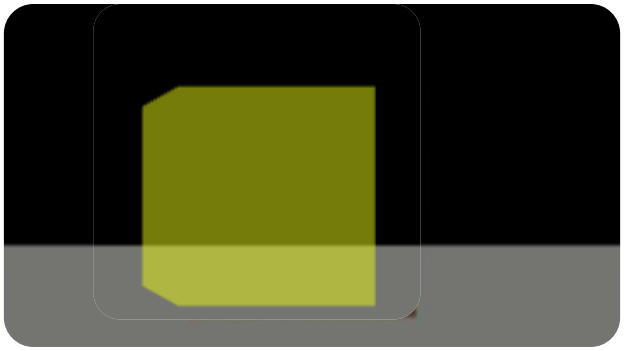
3. THREE.JS - PRIDANIE OBJEKTOV KOCIEK DO SCÉNY

- **Úloha:** Pridajte do scény objekt kocky s priesvitného materiálu.

1. V skripte **„ThreeScene.js“** pridajte do metódy **addObjects()** **implementáciu kocky cube2:**

```
var geometryOpacityBox = new THREE.BoxGeometry( 1, 1, 1 );
var materialOpacityBox = new THREE.MeshBasicMaterial( {
    color: 0xEAF913,
    transparent: true,
    opacity: 0.5
  } );
cube2= new THREE.Mesh(geometryOpacityBox,materialOpacityBox);
cube2.position.set(1.5, 0, 0);
scene.add( cube2 );
```

2. Otestujte implementáciu:



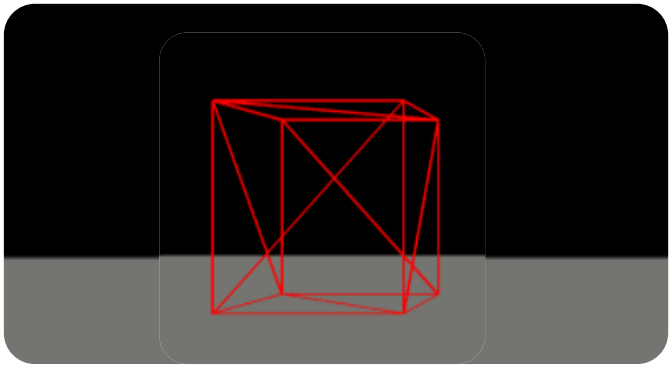
3. THREE.JS - PRIDANIE OBJEKTOV KOCIEK DO SCÉNY

- **Úloha:** Pridajte do scény objekt kocky vo forme drôtového modelu.

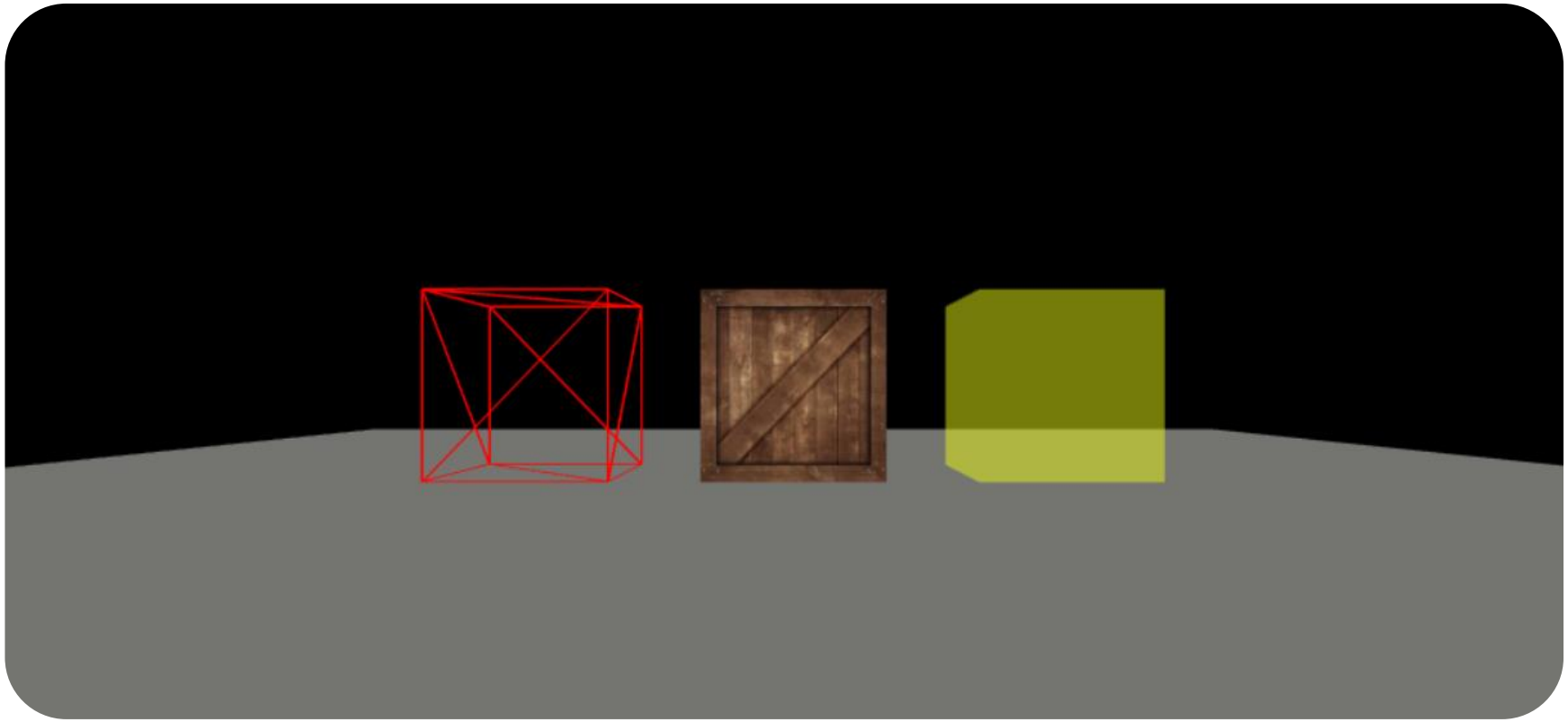
1. V skripte **„ThreeScene.js“** pridajte do metódy **addObjects()** **implementáciu kocky cube3:**

```
var geometryWiredBox = new THREE.BoxGeometry( 1, 1, 1 );
var materialWiredBox = new THREE.MeshBasicMaterial({
    color: 'rgb(255,0,0)',
    wireframe: true,
    transparent: true});
cube3 = new THREE.Mesh( geometryWiredBox, materialWiredBox );
cube3.position.set(-1.5, 0, 0);
scene.add( cube3 );
```

2. Otestujte implementáciu:



3. THREE.JS – CELKOVÝ VIZUÁL SCÉNY



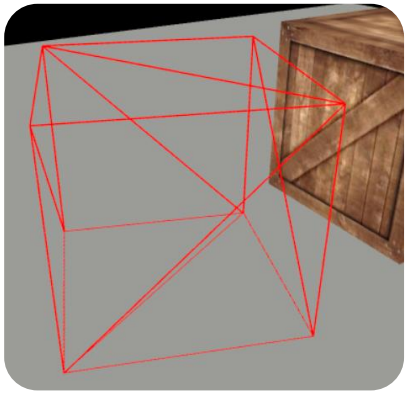
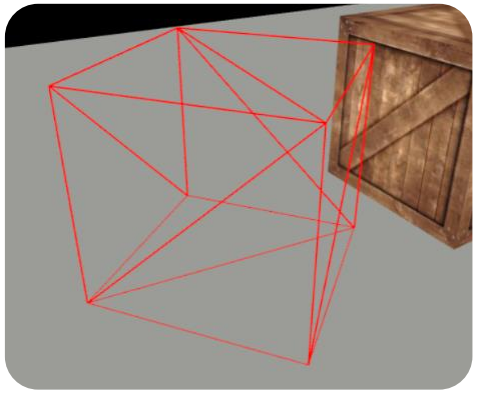
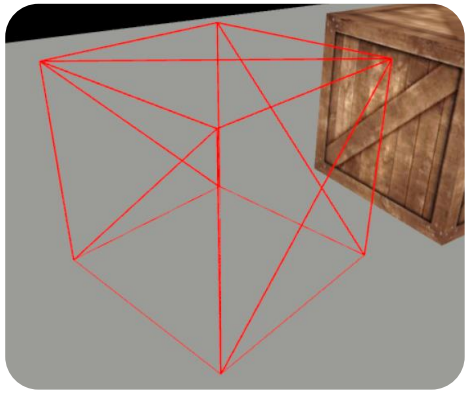
3. THREE.JS - ROTÁCIA OBJEKTU KOCKY V SLUČKE

- **Úloha:** Implementujte rotáciu objektu kocky v slučke okolo osi y s krokom *0,02*.

1. Pridajte do scény v skripte **„ThreeScene.js“** v metóde **render() implementáciu :**

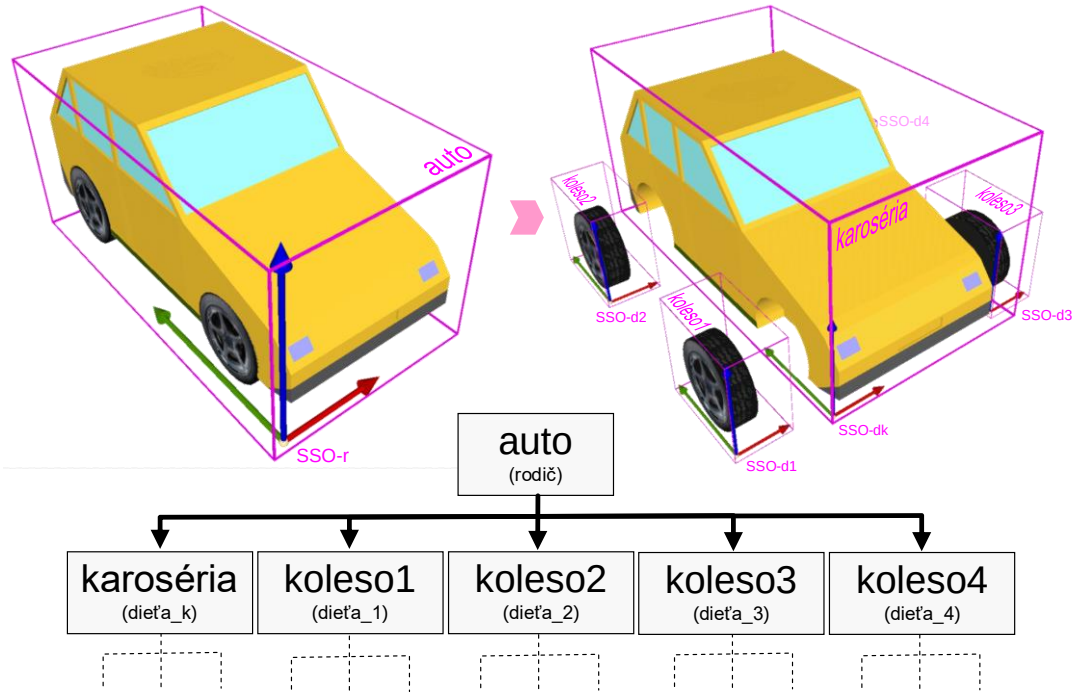
```
cube3.rotation.y += 0.02;
```

2. Následne otestujte správnosť implementácie. **Kedže je metóda render() volaná cyklicky**, aj **transformácia rotácie bude vykonávaná cyklicky** teda **v slučke** s krokom.

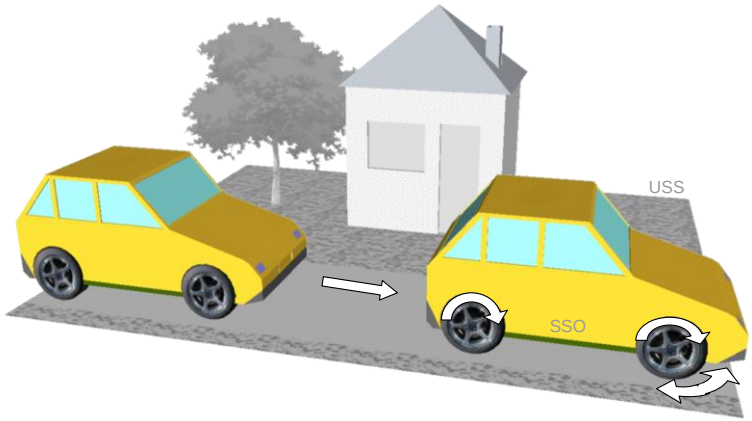
3. THREE.JS - VZŤAH RODIČ ↔ POTOMOK

- Pri tvorbe objektov je častokrát vhodné vytvárať objekty hierarchicky. Napríklad objekt automobilu môže byť definovaný ako skupina kolies a karosérie.
  Rodič je **nad-objektom** objektu **potomka**.
  Potomok dedí všetky **transformácie vykonané na rodičovi**.



3. THREE.JS - VZŤAH RODIČ ↔ POTOMOK

- Je potrebné zabezpečiť napríklad samostatnú rotáciu kolies ale zároveň aj pohyb automobilu ako celku.



- Príklad pre vytvorenie vzťahu ***RODIČ*** – ***POTOMOK*** medzi objektmi **cube3** ↔ **cube2** (v **render()**).

```
cube3.add(cube2); //cube2 bude potomkom cube3
cube2.position.set(3, 0, 0); //nastaví sa pozícia cube2 od cube3
cube3.rotation.y += 0.02; //rotovaním cube3 sa automaticky rotuje aj cube2
cube2.rotation.z += 0.02; //rotovanie len cube2
```

4. THREE.JS - *IMPLEMENTÁCIA OVLÁDANIA KAMERY* - *ORBITCONTROLS*

- OrbitControls je implementácia ovládania štandardne pomocou myši (s kolieskom). Pod súčasným štandardným ovládaním sa myslí:
  - Držanie ľavého tlačidla myši a pohybom kurzora (rotácia).
  - Držanie pravého tlačidla myši a pohybom kurzora (pohyb, posun).
  - Scrollovanie myšou t.j. pohyb kolieskom sa vykoná priblíženie/vzdialenie (zoom).

1. V skripte „ThreeScene.js“ pridajte na koniec metódy init() implementáciu *OrbitControls*:

```
controls = new THREE.OrbitControls(camera, renderer.domElement );
```

2. Následne pridajte na koniec metódy render() riadok pre kontinuálne riadenie vstupu ovládania :

```
controls.update();
```

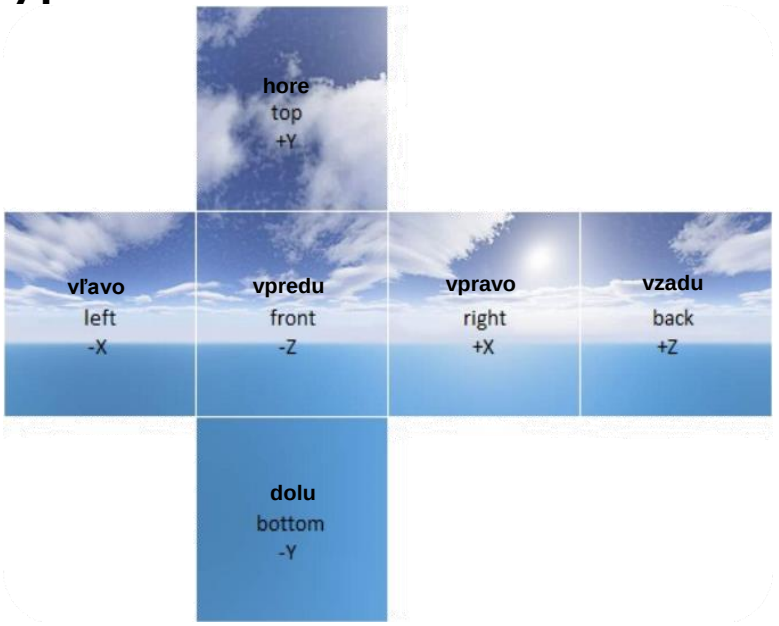
3. Po vložení *OrbitControls* overte správnosť implementácie. Spustite si vizualizáciu a ovládajte pohyb kamery.

5. THREE.JS - *IMPLEMENTÁCIA OKOLITÉHO PROSTREDIA* – *SKYBOX*

- Často je potrebné riešiť vizualizáciu okolia. Najčastejšie sa jedná o doplnenie scény o oblohu alebo napr. pohorie alebo horizont mesta či mora v pozadí výhľadu kamery. *Three.js* disponuje metódou *Skybox*, pomocou ktorej je možné jednoducho tento efekt dosiahnuť.

  Základný ***Skybox* môže byť dvojakého typu**:

guľa (sphere)



kocka (cube)

5. THREE.JS - *IMPLEMENTÁCIA OKOLITÉHO PROSTREDIA – SKYBOX*

- **Úloha:** Implementujte na aktuálnu scénu objekt **Skybox**, čím dodáte do scény zobrazenie oblohy v okolí.

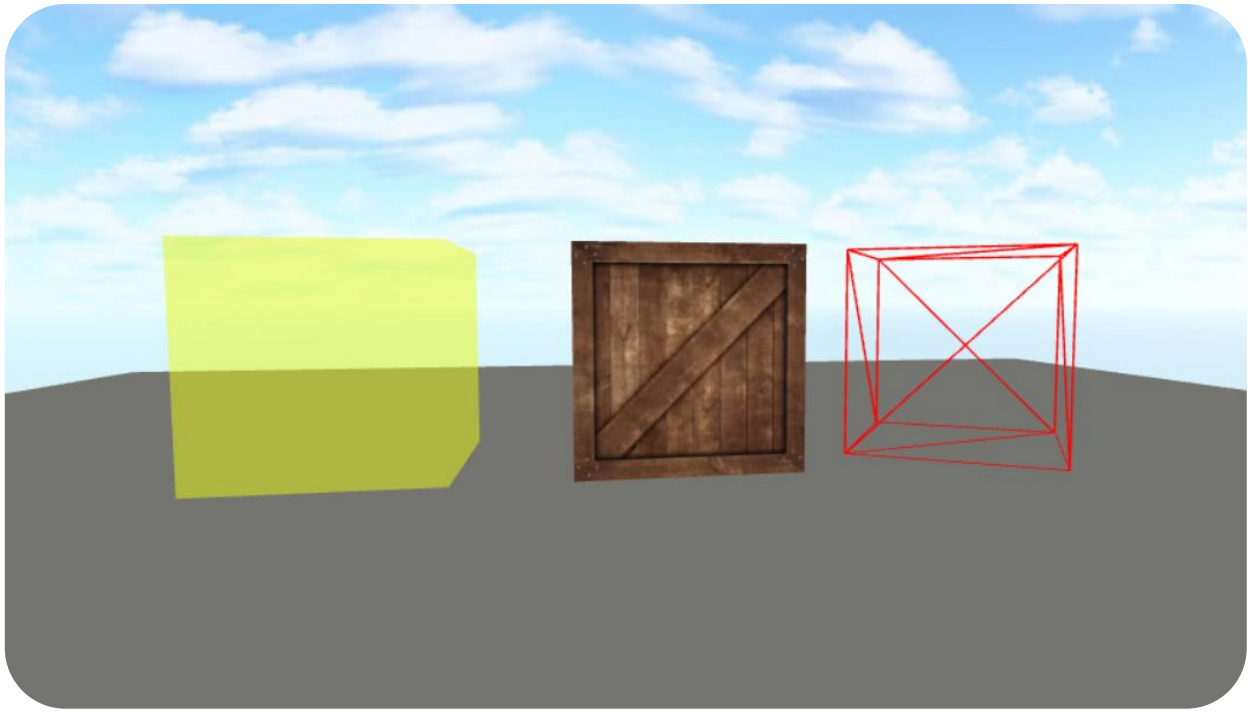
1. V skripte **„ ThreeScene.js**“ pridajte v metóde **addObjects()** **implementáciu primitíva gule :**

```
var geometrySphere = new THREE.SphereGeometry( 100, 100, 100 );
var cubeTexture = new THREE.ImageUtils.loadTexture(
                      'texture/sky.jpg' );
var materialSphere = new THREE.MeshBasicMaterial( {
    map: cubeTexture,
    transparent: true,
    side: THREE.DoubleSide} );
sphere = new THREE.Mesh( geometrySphere, materialSphere );
sphere.position.set(0, 0, 0);
scene.add( sphere );
```

5. THREE.JS - *IMPLEMENTÁCIA OKOLITÉHO PROSTREDIA – SKYBOX*

- **Úloha:** Implementujte na aktuálnu scénu objekt **Skybox**, čím dodáte do scény zobrazenie oblohy v okolí.

2. Po vložení primitíva overte správnosť implementácie. Vizuál scény po pridaní ***Skyboxu*** (oblohy) by mal vyzerať podobne ako na nasledujúcom obrázku.



DOPLŇUJÚCE ÚLOHY

- **Vložte do scény so *Skybox-om* ďalšiu kocku** (cube4) **s použitím materiálu textúry loga LIRKIS** z predošlého cvičenia.
- Implementujte **rotáciu** niektorého z **objektov okolo viac ako jednej osi naraz.**
- Implementujte rotáciu objektov **cube4-cube3** pri ich vzťahu ***Rodič↔Potomok*** okolo viac ako jednej osi naraz.
- Predveďte hotové implementácie cvičiacemu.

Úlohy na samostatné riešenie

- Implementujte rotáciu oblohy.
- Vyskúšajte vložiť do scény rôzne typy primitív.
- *Odkaz:* https://threejs.org/manual
  (https://threejsfundamentals.org/threejs/lessons/threejs-primitives.html)

Q & A

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

Katedra počítačov a informatiky, FEI TU v Košiciach