THREE.JS – VRHANIE TIEŇOV A IMPLEMENTÁCIA HMLY

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

C 10

Ciele cvičenia

- Three.js – Typy a implementácia tieňových máp.
- Three.js – Implementácia tieňov na objekty bez potomkov.
- Three.js – Implementácia tieňov na objekty s potomkami.
- Three.js – Implementácia hmly.

1. *THREE.JS* – PRÍPRAVA BALÍČKA

- **Úloha:** Stiahnite si balíček „**PG_10C23_Threejs_Vrhanie_tienov_a_hmla.zip**“ z portálu Moodle KPI a predmetu Počítačova grafika.
- Obsah balíčka skopírujte do vášho projektu aby štruktúra vyzerala nasledovne:
  - `WebGL_getStart`
    - `> css`
    - `> js`
    - `>> datGUI`
    - `>> threejs`
    - `>> ThreeShadow.js`
    - `>models`
    - `>> car`
    - `>>> Pony_cartoon.obj`
    - `>>> Pony_cartoon.mtl`
    - `>>> Body_dDo_d_orange.jpg`
    - `>>> . . .`
    - `> texture`
    - `index.html`

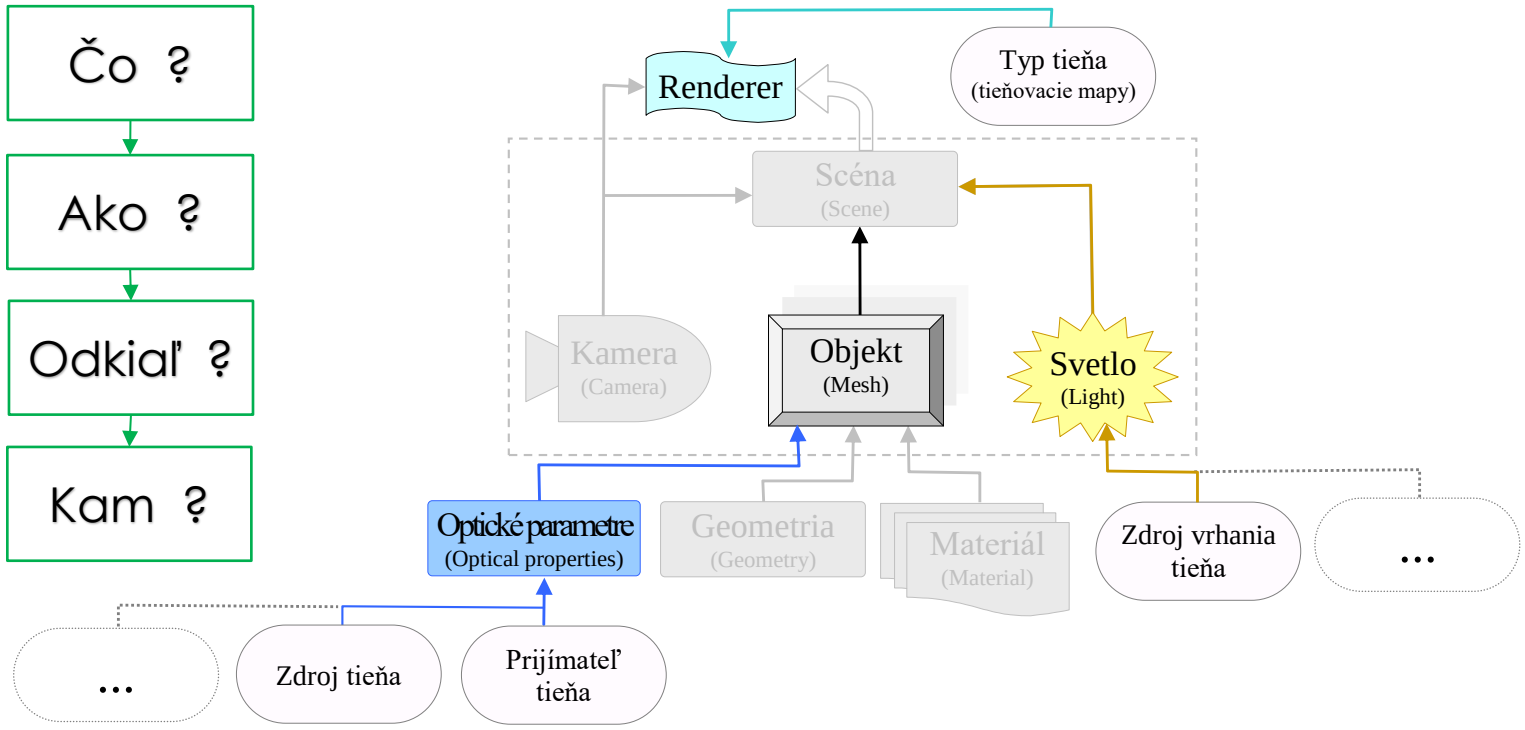
1. THREE.JS – SPUSTENIE BALÍČKA

1. Otvorte si skript „**ThreeShadows.js**“ a html súbor „index.html“ vo vašom vývojovom prostredí.
2. Skontrolujte súbor „**index.html**“ .
   Pokiaľ sa v elemente `<div id="canvas1"> . . . </div>` nenachádza skript referujúci na súbor „**Three ThreeShadows.js**“, doplňte ho nasledujúcim riadkom kódu :

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

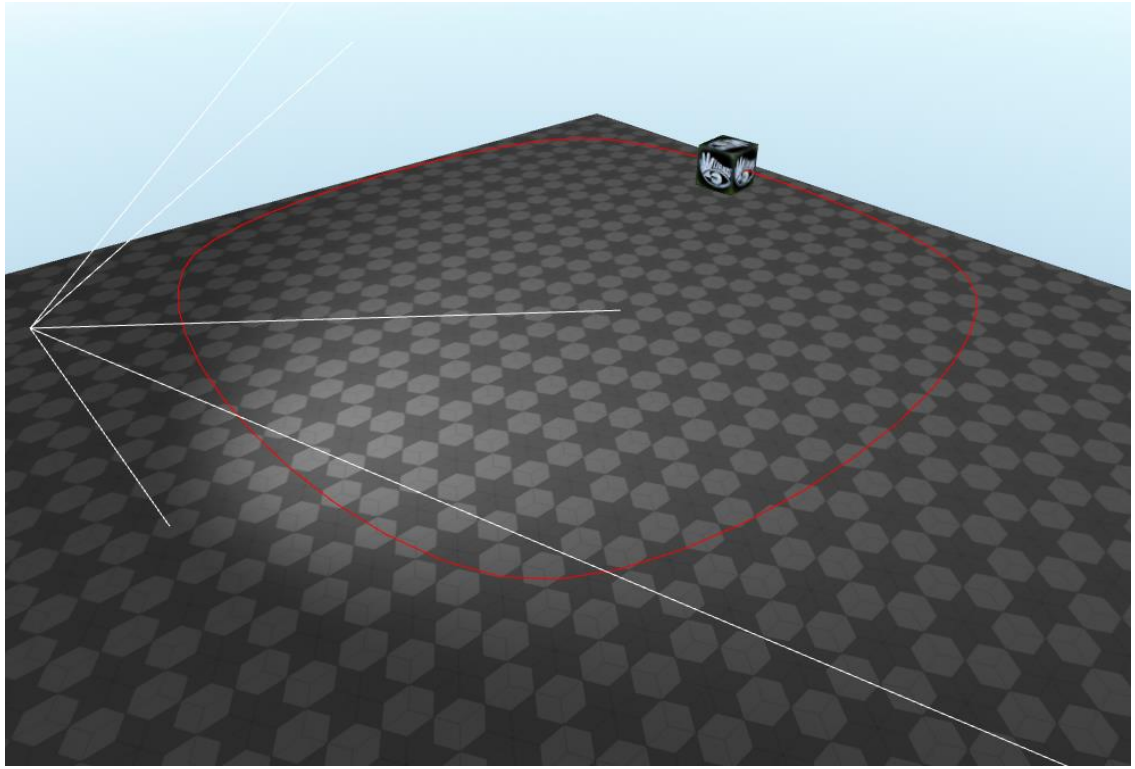
1. THREE.JS – PROCES VRHANIA TIEŇOV

- Proces vrhania tieňov v Three.js je založený na spojení viacerých parametrov a to najmä typov tieňovania a použitých tieňovacích máp, použitých typov svetiel (nie každý typ svetla podporuje vrhanie tieňov) a objektov ako zdrojov alebo prijímateľov tieňa.



1. *THREE.JS* – SPUSTENIE POČIATOČNEJ SCÉNY

- Po importovaní balíčka a správneho nasadenia skriptu v indexe, by počiatočná scéna mala vyzerať nasledovne (vychádzajúc z predošlého cvičenia):



2. THREE.JS – SHADOWMAP TYPY (BASIC)

- THREE. BasicShadowMap

- Dáva nefiltrované tieňové mapy (Najrýchlejší render výpočet avšak najnižšia kvalita)

2. *THREE.JS* – SHADOWMAP TYPY (PCF)

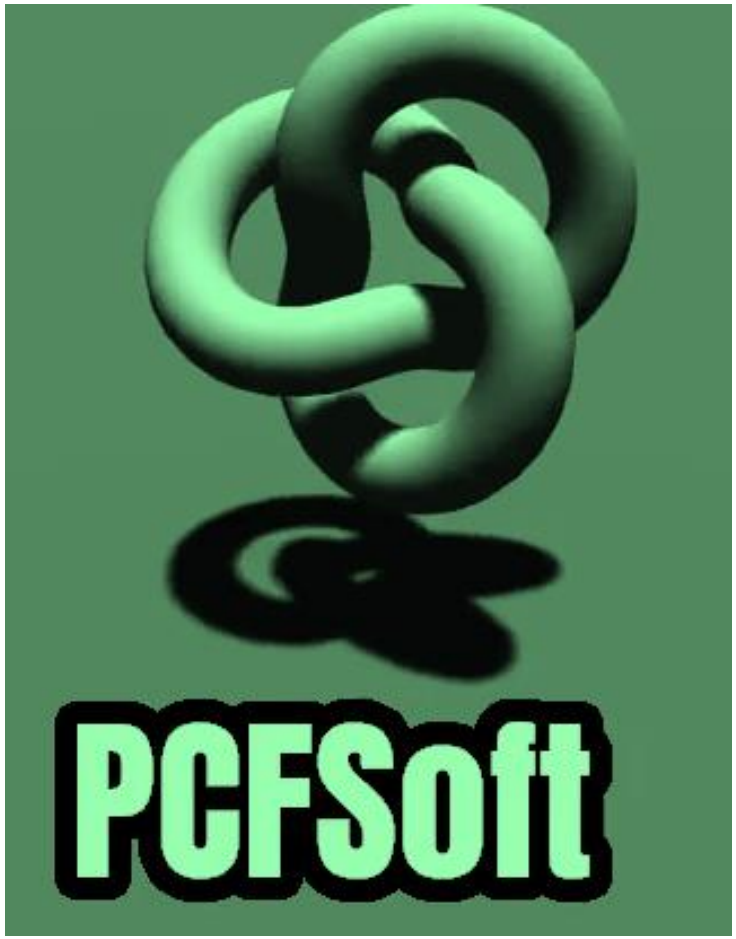
- THREE. PCFShadowMap

- Filtruje tiene pomocou Percentage-Closer Filtering (PCF) algoritmu, čím filtruje škálu zafarbenia postupne.

2. *THREE.JS* – SHADOWMAP TYPY (PCFSOFT)

- THREE. PCFSoftShadowMap

- Filtruje tiene pomocou Percentage-Closer Filtering (PCF) algoritmu s vyšším percentom prechodu zafarbenia. Najmä ak sa používa nízka kvalita tieňových máp.



2. *THREE.JS* – SHADOWMAP TYPY (VSM)

- THREE. VSMShadowMap

- Filtruje tiene pomocou Variance Shadow Map algoritmu. Rovnako využíva soft prechod a zároveň pri tomto type, všetci prijímači tieňa budú tak tiež tiene vrhať.

2. THREE.JS - *DEKLARÁCIA TIEŇOVÉHO ALGORITMU PRE RENDERER.*

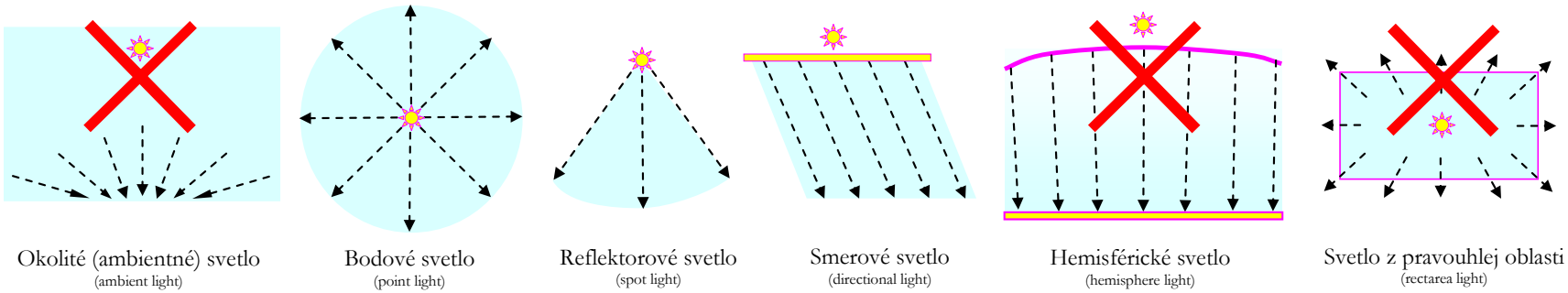
- **Úloha:** v skripte „**ThreeShadows**“ vo funkcií **init()** doplňte PCFSoft algoritmus pre renderer následovne:

```
renderer.shadowMap.enabled = true;
renderer.shadowMap.type = THREE.PCFSoftShadowMap;
```

- Riadok „**renderer**.shadowMap.enabled = true“ nám sprístupní WebGL konštantu pre renderer, ktorá pracuje s tieňovými algoritmami.
- Riadok „renderer.ShadowMap.type = THREE.PCFSoftShadowMap“ nám priradí PCFSoft algoritmický výpočet pre tieňovanie.

3. THREE.JS – *VRHANIE TIEŇOV – MOŽNOSTI SVETIEL*

- Vrhanie tieňov je možné iba na svetlách, ktoré majú zadefinovaný zdroj, vektor (smer), intenzitu a oblasť (areu). Nakoľko ambientné svetlo nie je definované priamo v prostredí ale definuje výšku emisie materiálov, toto svetlo nedokáže vrhať tiene.
- ThreeJS obsahuje ďalšie 2 typy svetiel:
  Hemisphere : https://threejs.org/docs/#api/en/lights/HemisphereLight - svetelný zdroj umiestnený priamo nad scénou s prechodom farby z farby oblohy na farbu základne.
  Rect : https://threejs.org/docs/#api/en/lights/RectAreaLight - vyžaruje svetlo rovnomerne cez tvar v pravouhlej rovine. Možno ho použiť na simuláciu svetelných zdrojov, ako sú svetlé okná alebo pásové osvetlenie.
  Ich využitie je možné prečítať na daných linkách. Tieto svetlá tiež nedokážu vrhať tieň.



3. THREE.JS – *VRHANIE TIEŇOV – OBJEKTY BEZ POTOMKOV*

- **Úloha:** v skripte „**ThreeShadows**“ vo funkcií **addLights()** pridajte možnosť vrhania tieňov pre reflektorové svetlo následovne:

```
spotlight1.castShadow = true;
```

- **Úloha:** v skripte „**ThreeShadows**“ vo funkcií **addObjects()** pridajte po vytvorení kocky ďalšiu kocku s možnosťou vrhania tieňov následovne:

```
cube = new THREE.Mesh(geometryCube, materialCube);
cube.position.set(0,0,0);
cube.castShadow = true;
scene.add(cube);
```

Pozor, teraz vám vrhá tieň iba kocka, ktorá rotuje po krivke, nakoľko tá je uložená ako posledná v parametri cube. Ak chcete aby vám vrhala tieň kocka v strede, musíte pridať cube.castShadow = true pred tým, ako sa vám pridá prvá kocka do scény.

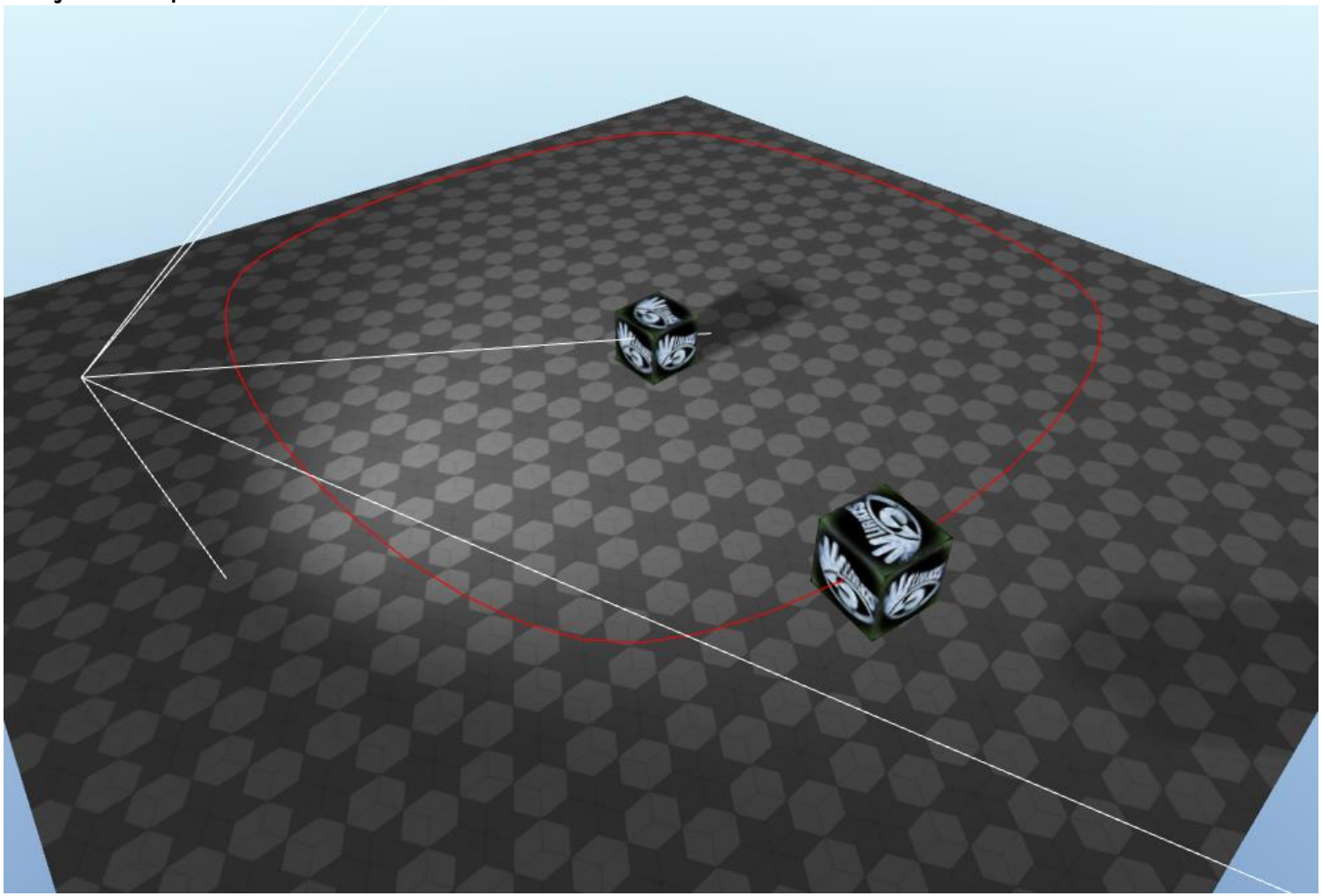
3. THREE.JS – *VRHANIE TIEŇOV - OBJEKTY BEZ POTOMKOV*

- Po spustení prostredia si je možné všimnúť, že napriek tomu, že sme pridali schopnosť vrhania tieňov pre objekty svetla a kociek, tieň sa nevytvoril. Aby sa tieňovo sfarbil objekt, na ktorý sa ma tieň vrhať, je potrebné pridať schopnosť danému objektu, aby tiene prijímal.
- **Úloha:** v skripte „**ThreeShadows**“ vo funkcií **addObjects()** pridajte možnosť prijímania tieňov pre objekt **plane** následovne:

```
plane.receiveShadow = true;
```

3. THREE.JS – *VRHANIE TIEŇOV - OBJEKTY BEZ POTOMKOV*

- Otestujte implementované riešenie



3. THREE.JS – *VRHANIE TIEŇOV - OBJEKTY S POTOMKOM*

- **Úloha:** v skripte „**ThreeShadows**“ vo funkcií **addObjects()** odstráňte inicializáciu jednej kocky.
- **Úloha:** v skripte „**ThreeShadows**“ vo funkcií **addObjects()** načítajte objekt auta pomocou funkcie **loadOBJectsPhong()** následovne:

```
loadOBJectsPhong( 0, -0.5, 0,
'models/car/Pony_cartoon.obj',
0.003, 0.003, 0.003,
'models/car/Body_dDo_d_orange.jpg',
'white');
```

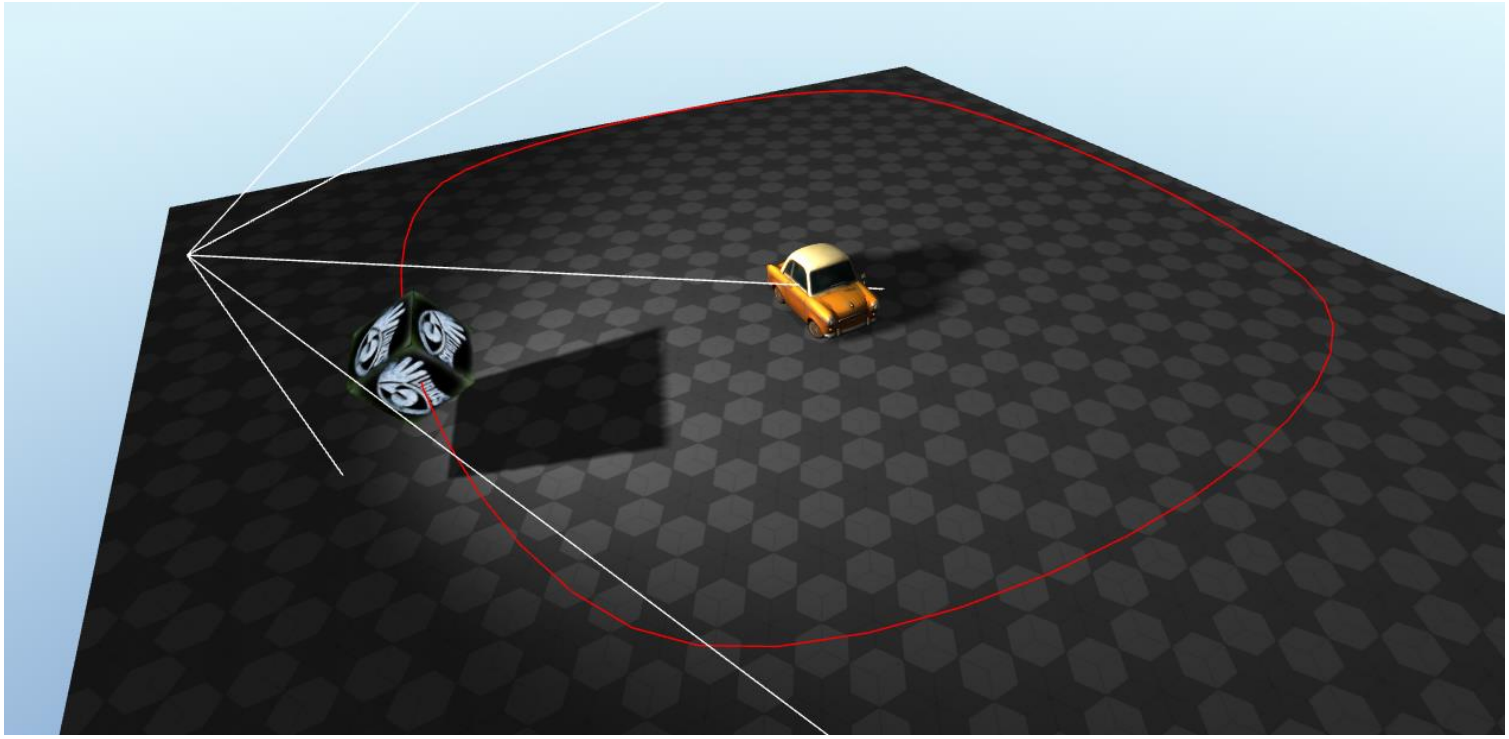
3. THREE.JS – *VRHANIE TIEŇOV - OBJEKTY S POTOMKOM*

- Objekty obsahujúce potomkov sa skladajú s viacerých objektov. Preto je potrebné, aby pre každého potomka bol samostatne nastavená schopnosť vrhania tieňov.
- **Úloha:** v skripte „**ThreeShadows**“ vo funkcií **loadOBJectsPhong()** vložte nasledujúci fragment kódu:

```
object.traverse( function ( child ) {
            if ( child instanceof THREE.Mesh ) {
                child.material.map = textureSurface;
                child.castShadow = true;
            }
        });
```

3. THREE.JS – *VRHANIE TIEŇOV - OBJEKTY S POTOMKOM*

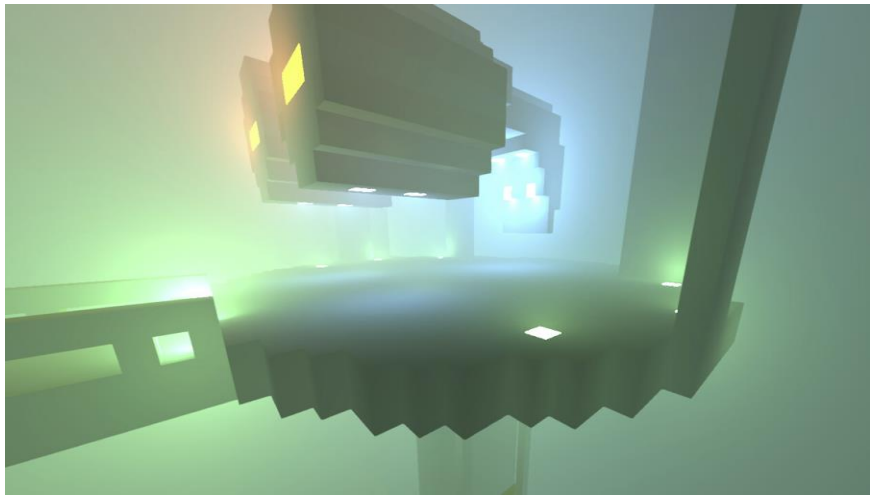
- Overte správnosť implementácie



4. THREE.JS – *HMLA*

- **THREE.Fog**(color, near, far);

1. **Color** – definuje farbu zahmlenia.
2. **Near** – definuje near clip plane, čiže minimálnu vzdialenosť kde sa má začať aplikovať hmla.
3. **Far** – definuje far clip plane, čiže maximálnu vzdialenosť, v ktorej sa hmla prestáva kalkulovať a aplikovať.



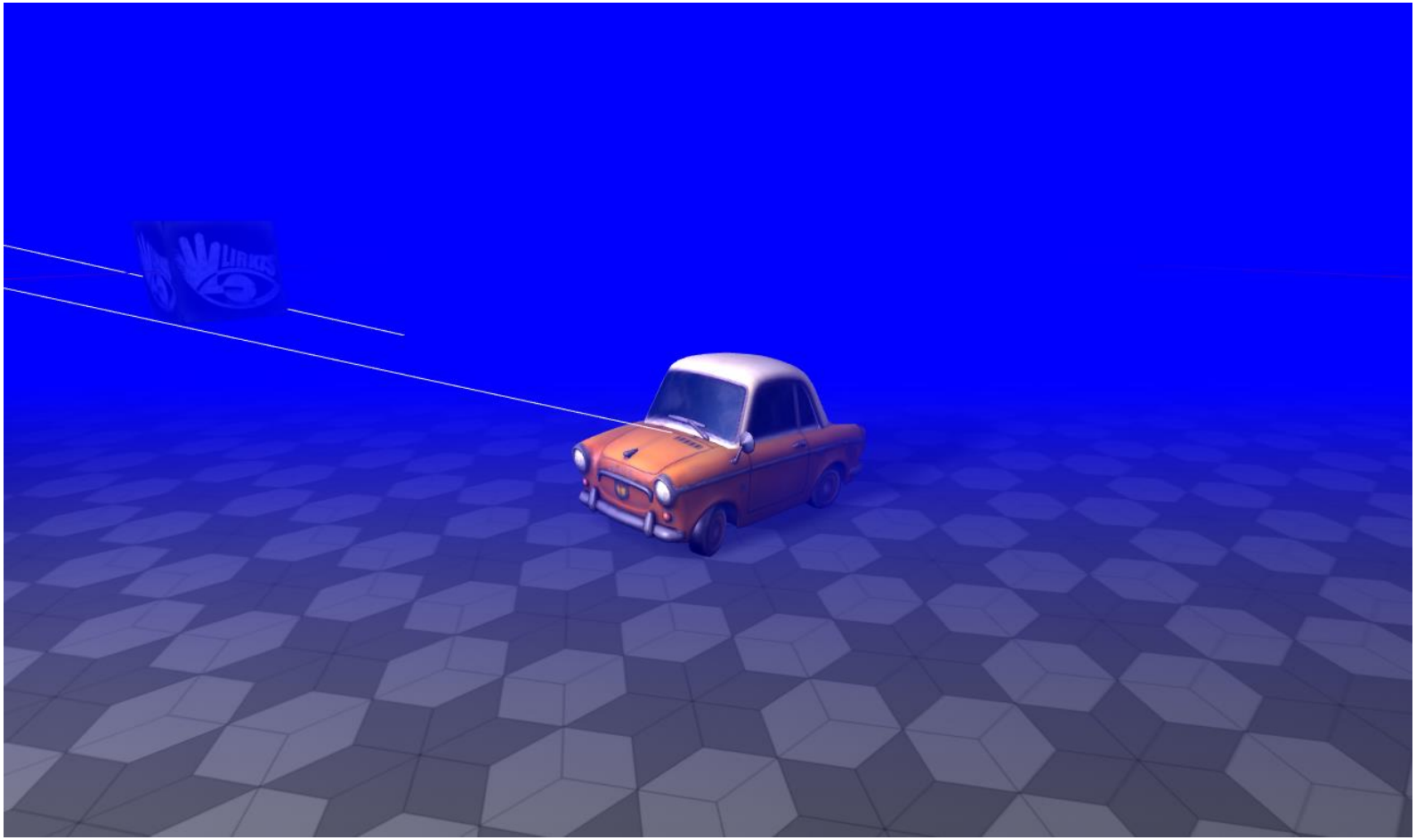
4. THREE.JS – *HMLA - IMPLEMENTÁCIA*

- **Úloha:** v skripte „**ThreeShadows**“ vo funkcií **init()** nahradte inicializáciu scény nasledujúcim fragmentom kódu

```
scene = new THREE.Scene();
    {
        const color = 0x0000FF;
        const near = 1;
        const far = 10;
        scene.fog = new THREE.Fog(color, near, far);
    }
```

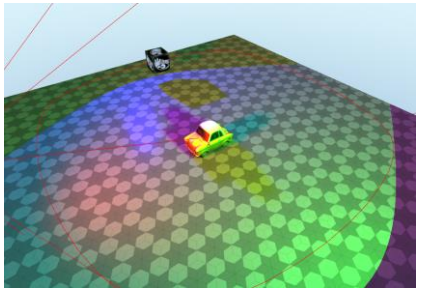
4. THREE.JS – *HMLA- OVERENIE*

- Overte implementované riešenie



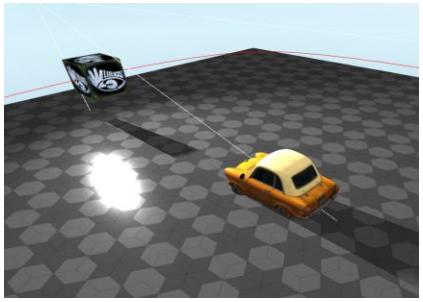
DOPLŇUJÚCE ÚLOHY

1. **Implementujte 3 reflektorové svetla podľa RGB s nastavením tieňov, intezitou 4 so smerovaním na objekt auta.**



2. **Nastavťe rozlíšenie tieňa na nižšie hodnoty pre svetlo.**

   Spotlight1.shadow.mapSize.width = 16;
   Spotlight1.shadow.mapSize.heigh = 16;

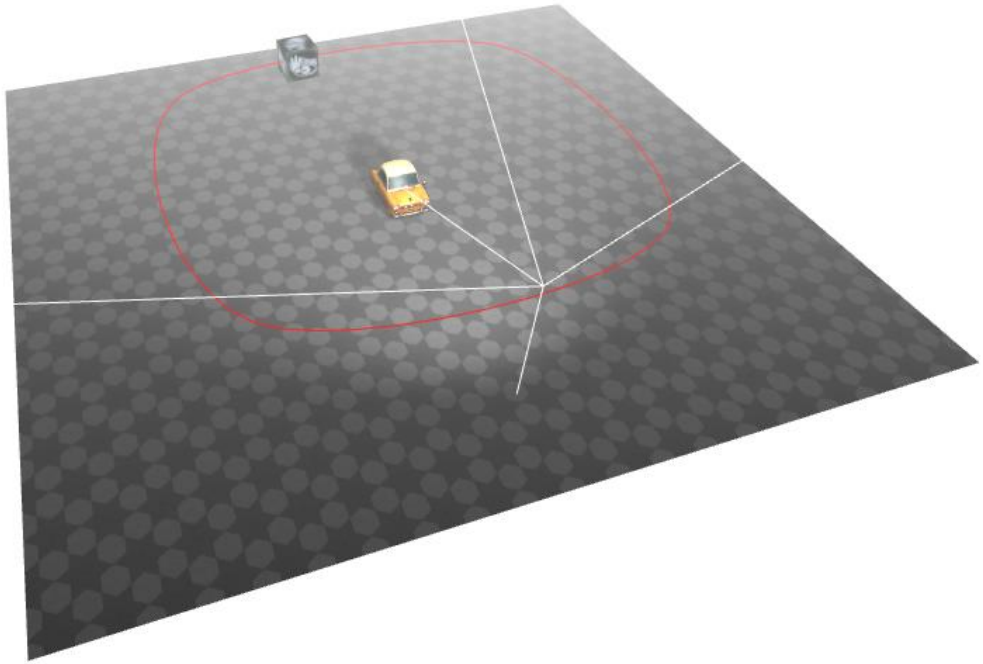


3. **Vymeňte objekty v scéne tak, že kocka bude umiestnená v strede scény a pohybovať sa bude po krivke model auta. Rozmiestnite reflektorové svetlá podľa bodu 1 tak, že sledovať ho bude len jedno svetlo a ostatné dve nasmerujte do scény podľa vlastného uváženia. Všetky objekty budú vrhať aj prijímať tieň.**

DOPLŇUJÚCE ÚLOHY

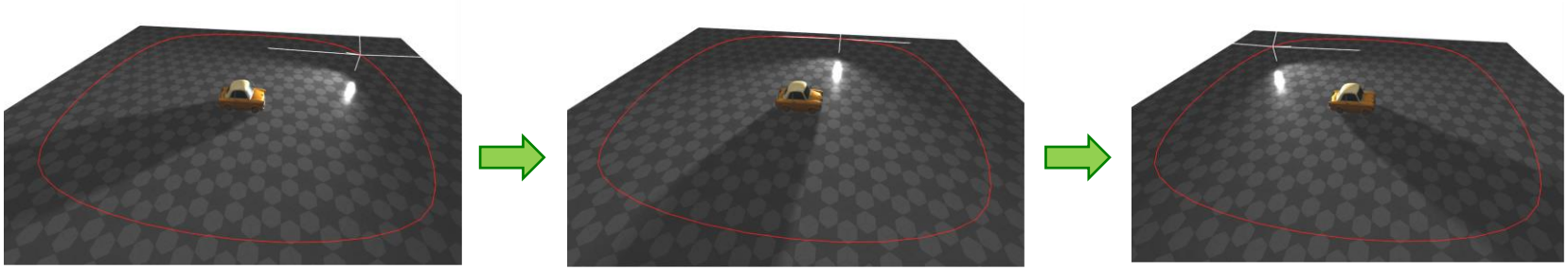
4. Vyskúšajte meniť niektoré z atribútov hmly:

- **Zmena vzdialenosti hmly :** `near = 10; far = 50;`
- **Farba hmly:** `color = 0xFFFFFF;`



ÚLOHY NA SAMOSTATNÉ RIEŠENIE

- Odstráňte zobrazenie kocky a ponechajte iba zobrazenie auta. Rotujte príslušné reflektorové svetlo po definovanej krivke a sledujte zmenu tieňa auta. Proces sledujte aj pri zapnutej hmle.



- Pridajte si výpis FPS do konzoly alebo na obrazovku. Zmeňte algoritmus tieňovanie pre renderer na Basic, PCF a VSM. Porovnajte kvalitu spracovania tieňa ako aj komplexitu prepočtu pre renderer.

Q & A

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

Katedra počítačov a informatiky, FEI TU v Košiciach