Univerzita Pardubice
Fakulta elektrotechniky a informatiky

Zobrazování informací prostřednictvím HUD v herním engine Three.js

Filip Mička

Bakalářská práce
2022

Univerzita Pardubice
Fakulta elektrotechniky a informatiky
Akademický rok: 2021/2022

ZADÁNÍ BAKALÁŘSKÉ PRÁCE
(projektu, uměleckého díla, uměleckého výkonu)

Jméno a příjmení:	**Filip Mička**
Osobní číslo:	**I19119**
Studijní program:	**B2646 Informační technologie**
Studijní obor:	**Informační technologie**
Téma práce:	**Zobrazování informací prostřednictvím HUD v herním engine Three.js**
Zadávající katedra:	**Katedra informačních technologií**

Zásady pro vypracování

V teoretické části bakalářské práce bude seznámení s herním engine Three.js a možnostmi zobrazování informací prostřednictvím HUD (heads-up display).
V implementační části bakalářské práce bude vytvořena webová stránka v rozsahu cca 10 ukázek implementace zobrazování informací prostřednictvím HUD. Práce se zaměří na nejčastěji potřebné zobrazování informací v počítačových hrách:

- Zdraví / uzdravení (Health / healing).
- Zbraň / munice / nabíjení (Weapon / ammunition / charging).
- Nabídky (Menus).
- Kontextové informace (Context-sensitive information).
- Kompas / ukazatel navigace (Compass / quest arrow).
- Lišta kompasu / ukazatel navigace (Compass bar / quest arrow).
- Minimapa (Mini-map).
- Dynamický zaměřovač / kurzor / nitkový kříž (Dynamic reticle / cursor / cross-hair).
- Herní pohled optikou / optické zvětšení (Game scope / optical magnification).

Webová stránka bude mít podobu přehledného výukového materiálu doplněná o obrazovou přílohou a komentované ukázky zdrojových kódů v jazycích HTML5, CSS a JavaScript v některém prototypovacím prostředí (JSFiddle, CodePen).

Rozsah pracovní zprávy: **min. 30 stran**
Rozsah grafických prací:
Forma zpracování bakalářské práce: **tištěná/elektronická**

Seznam doporučené literatury:

SUKIN, Isaac. *Game Development with Three.js*. 1st edition. Birmingham: Packt Publisher, 2013. ISBN 978-1-78216-853-9.
DIRKSEN, Jos. *Learn Three.js: programming 3D animations and visualizations for the web with HTML5 and WebGL*. 3rd edition. Birmingham: Packt Publisher, 2018. ISBN 978-1-78883-328-8.
LEWY, Blue. *DISCOVER three.js:* Welcome to the Missing Manual for three.js! [online]. 2020 [cit. 2021-10-27]. Dostupné z: https://discoverthreejs.com/.
MICROSOFT. *HUD Components Overview of common HUD components* [online]. 2021 [cit. 2021-10-27]. Dostupné z: https://gameuxmasterguide.com/2019-05-09-HUDComponents/.

Vedoucí bakalářské práce: **Ing. Zbyněk Kopecký**
Katedra informačních technologií

Datum zadání bakalářské práce: **17. prosince 2021**
Termín odevzdání bakalářské práce: **13. května 2022**

L.S.

Ing. Zdeněk Němec, Ph.D. v.r.
děkan

Ing. Jan Panuš, Ph.D. v.r.
vedoucí katedry

V Pardubicích dne 28. února 2022

Prohlašuji:

Tuto práci jsem vypracoval samostatně. Veškeré literární prameny a informace, které jsem v práci využil, jsou uvedeny v seznamu použité literatury.

Byl jsem seznámen s tím, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorský zákon, zejména se skutečností, že Univerzita Pardubice má právo na uzavření licenční smlouvy o užití této práce jako školního díla podle § 60 odst. 1 autorského zákona, a s tím, že pokud dojde k užití této práce mnou nebo bude poskytnuta licence o užití jinému subjektu, je Univerzita Pardubice oprávněna ode mne požadovat přiměřený příspěvek na úhradu nákladů, které na vytvoření díla vynaložila, a to podle okolností až do jejich skutečné výše.

Beru na vědomí, že v souladu s § 47b zákona č. 111/1998 Sb., o vysokých školách a o změně a doplnění dalších zákonů (zákon o vysokých školách), ve znění pozdějších předpisů, a směrnicí Univerzity Pardubice č. 9/2012, bude práce zveřejněna v Univerzitní knihovně a prostřednictvím Digitální knihovny Univerzity Pardubice.

V Pardubicích dne 17. 8. 2022

Filip Mička

PODĚKOVÁNÍ

Tímto bych chtěl poděkovat svému vedoucímu Ing. Zbyňku Kopeckému za cenné rady, trpělivost a pomoc při vytváření této práce. Dále bych chtěl poděkovat celé své rodině za to, že mi byla oporou nejen při vypracovávání této práce, ale i po celou dobu mého studia.

ANOTACE

Cílem bakalářské práce je nejprve seznámení s herním enginem Three.js a představení možností zobrazování informací pomocí HUD. Výstupem praktické části bakalářské práce je webová stránka, která slouží jako přehledný výukový materiál dané problematiky. Na stránce je uvedeno 9 ukázek nejčastěji zobrazovaných informací (zdraví, munice, zaměřovač, …). Stránka obsahuje obrazové přílohy a komentované ukázky zdrojových kódů v jazycích HTML, CSS a JavaScript, které jsou zasazeny do prostředí CodePen.

KLÍČOVÁ SLOVA

HUD, průhledový display, Three.js, HTML, CSS, JavaScript, WebGL, herní engine, zobrazování informací ve hrách, výuková webová stránka

TITLE

Displaying information using HUD in Three.js game engine

ANNOTATION

The main purpose of the bachelor's thesis is to apprise with game engine Three.js and to present possibilities of displaying information using HUD in Three.js. Practical part of this thesis is educational website. There are 9 illustrations of the most used HUD (health, munition, reticle, …). There are image attachments and commented code in languages HTML, CSS and JavaScript which is presented by CodePen environment.

KEYWORDS

HUD, heads-up display, Three.js, HTML, CSS, JavaScript, WebGL, game engine, displaying information in games, educational website

OBSAH

SEZNAM OBRÁZKŮ

SEZNAM ZDROJOVÝCH KÓDŮ

SEZNAM ZKRATEK

HUD	Heads-up display
3D	3 Dimensions
2D	2 Dimensions
API	Application Programming Interface
WebGL	Web Graphics Library
HTML	Hypertext Markup Language
CSS	Cascading Style Sheets
RGB	Red, Green, Blue
HSL	Hue, Saturation, Luminosity
PBR	Physically Based Rendering Materials
PNG	Portable Network Graphics
SVG	Scalable Vector Graphics
URL	Uniform Resource Locator
CD	Compact Disc

ÚVOD

Mnoho lidí zřejmě neví, co se za zkratkou HUD skrývá, ale jistě se s HUD prvky setkali v běžném životě. Příkladem může být osobní automobil, kde nás mohou informovat o aktuální rychlosti nebo například letadlo, kde mohou sloužit k zobrazení letové hladiny či náklonu letadla. Největší oblibě HUD prvků se ovšem těší počítačové hry. V současné době existuje nepřeberné množství herních titulů a nejspíše žádná počítačová hra se neobejde bez jisté formy přenosu herních informací právě prostřednictvím HUD prvků. Zde se takto například zobrazují informace o množství zdraví hráče, munice nebo také zaměřovač zbraně.

Herní engine Three.js se v současné době těší poměrně velké oblibě a s problematikou tvorby HUD prvků se tak setkává nemalé množství herních vývojářů, kteří by jistě uvítali možnost inspirace v ukázkách HUD prvků.

Cílem této práce je nejprve v teoretické části seznámit čtenáře s problematikou HUD prvků používaných ve hrách, představit základní principy knihovny Three.js a popsat, jaké jsou možnosti pro realizaci HUD prvků v knihovně Three.js.

Cílem praktické části této práce je vypracování ukázek HUD prvků v herním enginu Three.js. Následně vytvořit přehlednou webovou prezentaci s těmito ukázkami, ve které bude možné ukázky prohlížet pomocí některého z prototypovacích prostředí.

1 HUD V HERNÍM PROSTŘEDÍ

Význam zkratky HUD pochází z anglického slovního spojení Heads-Up Display. Tento termín sahá historií až do druhé světové války, kdy se takto začal označovat průhledový display v letadlech. Výhoda tohoto zobrazení informací spočívá v tom, že je pilot vidí v dráze letu a může se tak více soustředit na samotný let. [2]

Následně koncem 80. let se začal tento způsob zobrazování informací používat i v automobilovém průmyslu. [3]

***Obrázek 1**: HUD prvky v automobilu (zdroj [4])*

HUD prvky jsou významnou součástí téměř všech počítačových her již od jejich počátku. Hráči předkládají důležité informace takovým způsobem, aby zároveň těmito informacemi nebyl příliš rušen od samotného herního zážitku. [5]

Bez HUD prvků by hráč neměl informace o aktuálním stavu hry a například při hraní akčních žánrů, kde je využíváno střelných zbraní, by míření bez zaměřovacího kříže bylo jistě mnohem komplikovanější.

1.1 Definice HUD prvku v herním odvětví

Za HUD prvek lze považovat herní informaci, která je zobrazena hráči během hraní. Nejčastěji takto bývá zobrazena informace o aktuální stavu zdraví, munici, pozici na mapě nebo poloze zaměřovače. HUD prvky mají ve většině případů statickou pozici na obrazovce tak, že v hráči navozují dojem pohledu průhledovým displejem. [5]

1.2 Základní vlastnosti HUD prvků ve hrách

Ve většině případů má hra do jisté míry v hráči vzbudit pocit skutečnosti. HUD prvky by tedy v žádném případě neměly působit rušivým dojmem. Měly by být minimalistické, ale zároveň dobře čitelné. Hráči by měly být zobrazovány pouze informace, které skutečně potřebuje a neměl by být zbytečně rušen irelevantními informacemi. [5]

Styl HUD prvků zobrazovaných uživateli je také nutné přizpůsobit žánru hry. Pokud se jedná o sci-fi žánr, použití výrazných HUD prvků nebude nijak na škodu. Naopak do hry skvěle zapadnou. Ve hrách cílených na realistický zážitek ze hry, je nutné zvolit co nejmenší počet HUD prvků v decentním designu. [5]

Je také nutné brát ohled na to, do jakého prostředí bude hra zasazena, aby HUD prvky nesplývaly s okolím. [5] Při volbě výrazné barvy mohou HUD prvky působit rušivým dojmem, proto se lze ve hrách setkat často s různými odstíny a různou průhledností bílé barvy, která je dostatečně kontrastní, ale zároveň nepůsobí nikterak rušivě.

1.3 Ukázky použití HUD prvků v moderních hrách

***Obrázek 2**: Ukázka decentních HUD prvků ve hře The Last of Us Part II (zdroj [6])*

***Obrázek 3**: Ukázka výrazných HUD prvků ve hře Apex Legends (zdroj [7])*

1.4 Typy HUD prvků

HUD prvky lze rozdělit do několika kategorií, dle informace, kterou zobrazují. Pro každý prvek se používá jiné grafické zpracování a umístění. HUD prvek by měl být zpracován tak, aby hráč na první pohled věděl, jakou informaci zobrazuje.

1.4.1 Zdraví

Pro zobrazení informace o stavu zdraví lze využít několik možností, případně jejich kombinaci.

Jednou z možností je zobrazovat stav zdraví pomocí ikon, často se využívá ikony srdce. Při změně stavu zdraví je možné odebírat celé ikony, nebo pro spojitější změnu stavu zdraví podbarvovat jinou barvou ikony po částech. Další možností je zobrazit stav zdraví pomocí čísla, které indikuje procentuální stav zdraví. Často využívanou možností je zobrazení zdraví pomocí lišty zdraví (health bar), kdy je hodnota zdraví vykreslena pomocí pruhu. [8]

Kromě aktuální hodnoty zdraví je také důležité nějakým způsobem hráči zobrazit úbytek nebo nárůst stavu zdraví. To je možné realizovat například změnou podbarvení HUD prvku. [8]

Další důležitou informací pro hráče, kterou je vhodné zobrazit, je maximální stav zdraví. Při použití lišty zdraví je možné tuto informaci zobrazit pomocí světlejší barvy pozadí. Hodnota aktuálního stavu zdraví je zobrazena tmavší barvou. Při úbytku zdraví má pozadí stejnou délku a mění se pouze délka tmavější části pruhu v popředí. [8]

HUD prvek indikující stav zdraví bývá umístěn v jednom z rohů obrazovky. Nejčastěji pravý nebo levý dolní roh. [8]

1.4.2 Zaměřovač

Zaměřovač ve hrách indikuje například směr, kterým při výstřelu poletí kulka ze zbraně. U zaměřovače je důležité zvolit správnou barvu. Ta by měla být kontrastní s okolním prostředím tak, aby nedošlo ke splývání s okolím. [8]

Ve hrách bývají zaměřovače vykreslovány nejčastěji v různých podobách nitkových křížu nebo pomocí kružnice s tečkou uprostřed. [8]

Zaměřovač lze vytvořit dynamický, kdy při pohybu dochází k rozmíření (zvětšení) zaměřovače a tak i snížení přesnosti při střelbě hráče.

Dále je možné hráči pomocí zaměřovače zobrazit zásah protihráče (například pomocí změny barvy) nebo různých animací zaměřovače (například v podobě pootočení zaměřovacího kříže). [8]

Střelné zbraně jsou někdy vybaveny optickým zaměřovačem se zvětšením, kdy hráči většinou pomocí pravého tlačítka myši je umožněn pohled touto optikou. Pro tento typ zaměřovače je možné použít ruzné textury pro navození pocitu pohledu skrz optický zaměřovač.

Zaměřovač bývá umístěn přirozeně uprostřed obrazovky. [8]

1.4.3 Kompas

Tento HUD prvek slouží pro zlepšení orientace hráče na mapě. Na kompasu se kromě směru, kterým hráč míří, může zobrazovat také pozice některých objektů ve hře, o kterých hráč potřebuje vědět. Mezi tyto objekty patří například nepřátelé nebo různé úkoly. [8]

Kompas může být ve hře vykreslen pomocí lišty kompasu, kde je zobrazeno několik po sobě jdoucích hodnot na kompasu, ve středu lišty je zpravidla hodnota směru hráče. Přesná hodnota směru hráče bývá ještě vyznačena například pomocí malého trojúhelníčku, nebo šipky. Pozice důležitých objektů bývají zobrazeny pomocí různých ikon či geometrických tvarů. Lišta kompasu bývá ve středu horní části obrazovky. [8]

Další možností zobrazení kompasu je použití obrázku kompasu. Tento obrázek je následně otáčen dle orientace hráče na mapě. Hodnotu směru hráče opět určuje identifikátor, například v podobě šipky. Toto zobrazení kompasu je vhodné umístit do některého z rohů obrazovky.

1.4.4 Munice

Informace o stavu munice v zásobníku hráče bývá zobrazena číselnou hodnotou nebo pomocí ikon náboje. Zde je také vhodné zobrazit maximální stav zásobníku, nebo celkový stav zásob munice hráče. Poblíž stavu munice by měla být také zobrazena zbraň, kterou hráč aktuálně používá. Dále je možné hráče informovat o prázdném zásobníku například červeným podbarvením tohoto HUD prvku. [8]

Stav munice je vhodné umisťovat do některého z rohů obrazovky nebo případně do dolní střední části obrazovky. V některých hrách (zejména sci-fi žánru) bývá stav munice zobrazen na display zbraně, kterou hráč drží. [8]

1.4.5 Minimapa

Minimapa slouží pro zlepšení orientace hráče na mapě. Je zde zobrazena část mapy, ve které se hráč právě pohybuje, ukazatel hráče a důležité cíle v okolí hráče. Minimapa by měla být zjednodušená, aby nebyl hráč zatěžován zbytečnými informacemi. [8]

Tento HUD prvek je vhodné umístit do levého nebo pravého dolního rohů obrazovky. Není vhodné vytvářet minimapu příliš velkou. [8]

1.4.6 Kontextové informace

Tyto informace jsou zobrazeny hráči při určité události ve hře. Příkladem kontextové informace může být zobrazení výzvy pro stisknutí klávesy, když se hráč nachází u dveří. Kontextová informace je zde zobrazena pro otevření i zavření dveří stejná. Obeně lze říci, že stejná klávesa

je použita pro různé akce. Přínosem pro hráče tedy je, že si nemusí pamatovat několik kláves pro podobné akce. [9]

Hráči je zobrazena požadovaná klávesa, která je někdy doplněna krátkým textem, který danou událost vystihuje.

Tyto HUD prvky ve hrách bývají většinou zobrazovány poblíž středu obrazovky.

1.4.7 Menu

Menu býva hráči většinou zobrazeno hned po spuštění hry. Mělo by proto na hráče udělat dobrý dojem. Samotný vzhled menu by měl být přizpůsoben žánru hry. Pro hráče by orientace v menu měla být snadná. Jako pozadí bývá někdy využíváno různých scén ze hry. [10]

Menu bývá tvořeno několika možnostmi v podobě tlačítek (téměř vždy obsahuje možnosti continue, options a exit).

Umístění menu ve hrách bývá různé, ale nejčastěji se lze setkat s umístěním uprostřed obrazovky.

2 HERNÍ ENGINE THREE.JS

Three.js je 3D knihovna, která je založena na JavaScriptu. Umožňuje vytvářet bohaté 3D scény, které jsou zobrazovány pomocí webového prohlížeče. Šířena je pod licencí open source. Používání knihovny Three.js ušetří velké množství kódu, oproti použití samotného API WebGL. [11, s. 8]

Například krychli je možné v knihovně Three.js vytvořit pouze pomocí tří řádků kódu. Pomocí dalších dvou řádků kódu lze krychli přidat do scény a scénu vykreslit vykreslovačem. [12]

2.1 Vytvoření Three.js projektu

Nejprve je nutné vytvořit základní HTML dokument. V hlavičce dokumentu je vhodné vytvořit část `style`, aby bylo možné pracovat s kaskádovými styly nebo případně pro lepší přehlednost naimportovat CSS soubor. V těle dokumentu je následně vytvořena část `script`, která bude obsahovat samotný kód pro práci s knihovnou Three.js. Opět je pro lepší přehlednost možné kód neumisťovat přímo do HTML dokumentu, ale vytvořit samostatný soubor s kódem JavaScriptu a poté ho naimportovat v HTML dokumentu. Výstup vykreslovače (element `canvas`) je následně často vkládán v kódu JavaScriptu přímo do těla dokumentu a vykreslován do celého okna prohlížeče. Proto je potřebné tělu dokumentu (element `body`) nastavit vlastnost `margin` na hodnotu 0. [13, s. 9-10]

Dále je nutné importovat knihovnu Three.js do aplikace. To lze následujícími způsoby.

První možností, jak importovat knihovnu Three.js do projektu, je pomocí elementu `script` v HTML dokumentu, kde jako hodnota atributu `src` je uvedena cesta ke zdrojovému kódu, dané verze knihovny. [11, s. 15]

```
<script src="https://unpkg.com/three@0.140.2/build/three.js">
</script>
```

***Zdrojový kód 1**: Import knihovny (zdroj [11, s. 15], upraveno autorem)*

Knihovnu Three.js lze také importovat pomocí importovacích map jako modul. Zde je nutné nejdříve vytvořit v HTML dokumentu element `script`, který má nastaven atribut `type` na hodnotu `"importmap"`. V kódu JavaScriptu je následně uvedena importovací mapa. Dále je nutné vytvořit další element `script`, který má nastavený atribut type na hodnotu `"module"`. Zde je pomocí příkazu `import` naimportován modul knihovny ze zdroje dle importovací mapy. [1]

```
<script type="importmap">
```

```
    {
        "imports": {
            "three": "https://unpkg.com/three@0.140.2/build/three.module.js"
        }
    }
</script>

<script type="module">
    import * as THREE from 'three';
</script>
```

***Zdrojový kód 2**: Import modulu knihovny (zdroj [1], upraveno autorem)*

Některé komponenty Three.js nejsou součástí knihovny a je nutné je importovat zvlášť. Příkladem může být komponenta `OrbitControls`, kterou je možné naimportovat opět například pomocí tagu `script` v HTML dokumentu s cestou nastavenou ke zdrojovému kódu komponenty. [13, s. 210]

2.2 Webový server

Vykreslovač WebGL nemůže pracovat přímo s lokálním úložištěm. Pokud je využito zdrojů z lokálního uložiště (například obrázky), je nutné používat webový server. [14]

Pro tyto účely je možné zvolit například aplikaci Web Server for Chrome, kde je nutné pouze vybrat složku s projektem a poté webový server spustit pomocí přepínače. Pokud je HTML soubor pojmenován jako index, při kliknutí na jednu z URL adres webového serveru (pod tlačítkem pro výběru složky) se spustí daná stránka v prohlížeči. [15]

Při použití vývojového prostředí Microsoft Visual Studio Code je také možné využít rozšíření v podobě nástroje Live Server. Po jeho instalaci je možné web server spouštět přímo ve vývojovém prostředí nad otevřeným projektem pomocí tlačítka Go Live v pravém dolní rohu. [16]

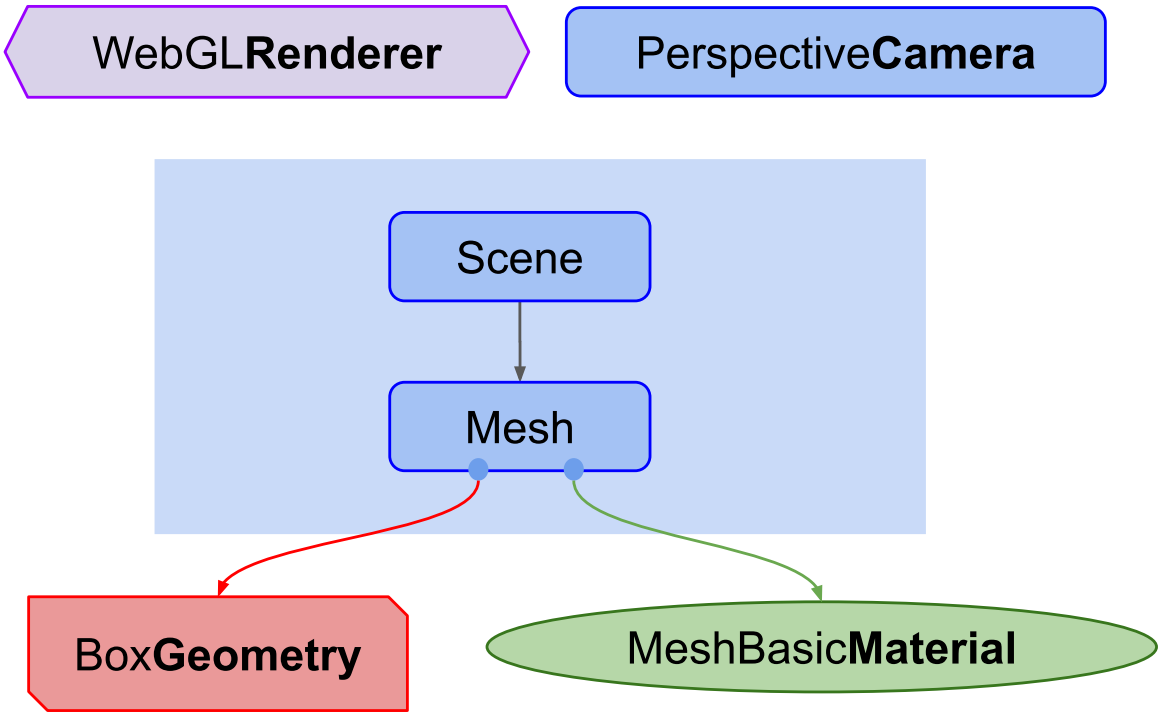
2.3 Struktura Three.js aplikace

Pro vytvoření aplikace pomocí Three.js je nutné vytvořit několik objektů. Mezi nejdůležitější a nezbytné patří objekty vykreslovače, scény a kamery. [17]

Objekt třídy `WebGLRenderer` (vykreslovač) slouží pro převedení 3D scény, která je zachycena kamerou do 2D obrazu. Tento 2D obsah je následně vykreslen na HTML element `canvas`. [12]

Objekt třídy `Scene` slouží k určení toho, co bude vykreslováno. Scéna tvoří stromovou strukturu. Do scény mohou být dále přidávány objekty třídy `Mesh`, `Object3D`, `Group` a `Light`. [12]

Objekt třídy `Camera` je objekt, který slouží k zachycení scény a určuje jaká část scény bude vykreslena. Odlišností tohoto objektu je, že nemusí být nutně přidán do scény. [12]



***Obrázek 4**: Základní struktura aplikace Three.js (zdroj [12])*

2.4 Vykreslovač

Při využití objektu třídy `WebGLRenderer` je scéna vykreslována pomocí vykreslovače WebGL. Konstruktor tohoto objektu nemá žádné parametry. Vykreslovači je nutné určit velikost výstupu, kam bude vykreslovat. Metoda `setSize` prvním parametrem určuje délku a druhým parametrem šířku. Většinou se vykresluje do celého okna, proto lze jako parametry uvést `window.innerWidth` pro délku a `window.innerHeight` pro výšku. Dalším nezbytným úkonem s vykreslovačem je předání výstupu vykreslovače vlastností `domElement` (element `canvas`) HTML elementu. Tím je nejčastěji samotné tělo dokumentu. [13, s. 12]

Je možné nastavit barvu pozadí vykreslování tak, aby pozadí nebylo úplně černé, jako je tomu ve výchozím stavu. To lze pomocí metody `setClearColor`, kde je jako první parametr zadána

barva a jako druhý parametr je zadána průhlednost. Průhlednost se zadává v hodnotách od 0.0 (průhledná), do 1.0 (neprůhledná). [13, s. 12]

Pro vykreslení scény je nutné u objektu vykreslovače volat metodu `render`. Tato metoda vyžaduje dva parametry. První je objekt scény, která bude vykreslována. Druhý parametr je objekt kamery, pomocí které bude scéna snímána. Jelikož scéna nebývá většinou statická, ale dochází v ní například ke změnám pozice objektů nebo kamery, je ji nutno neustále překreslovat. Pro tyto účely je vodné vytvořit a následně volat funkci například s názvem `animate` nebo `render`, která bude obsahovat kód, který se provede vždy při překreslení scény. Do těla funkce následně umístit volání metody `render` objektu vykreslovače. Dále v této funkci lze využít metodu `requestAnimationFrame`, kde jako parametr je uveden název vytvořené funkce (`animate` nebo `render`). Metoda `requestAnimationFrame` zajistí, aby při překreslení obrazovky uživatele došlo k volání vytvořené funkce a scéna tak byla neustále překreslována. [17]

2.5 Scéna

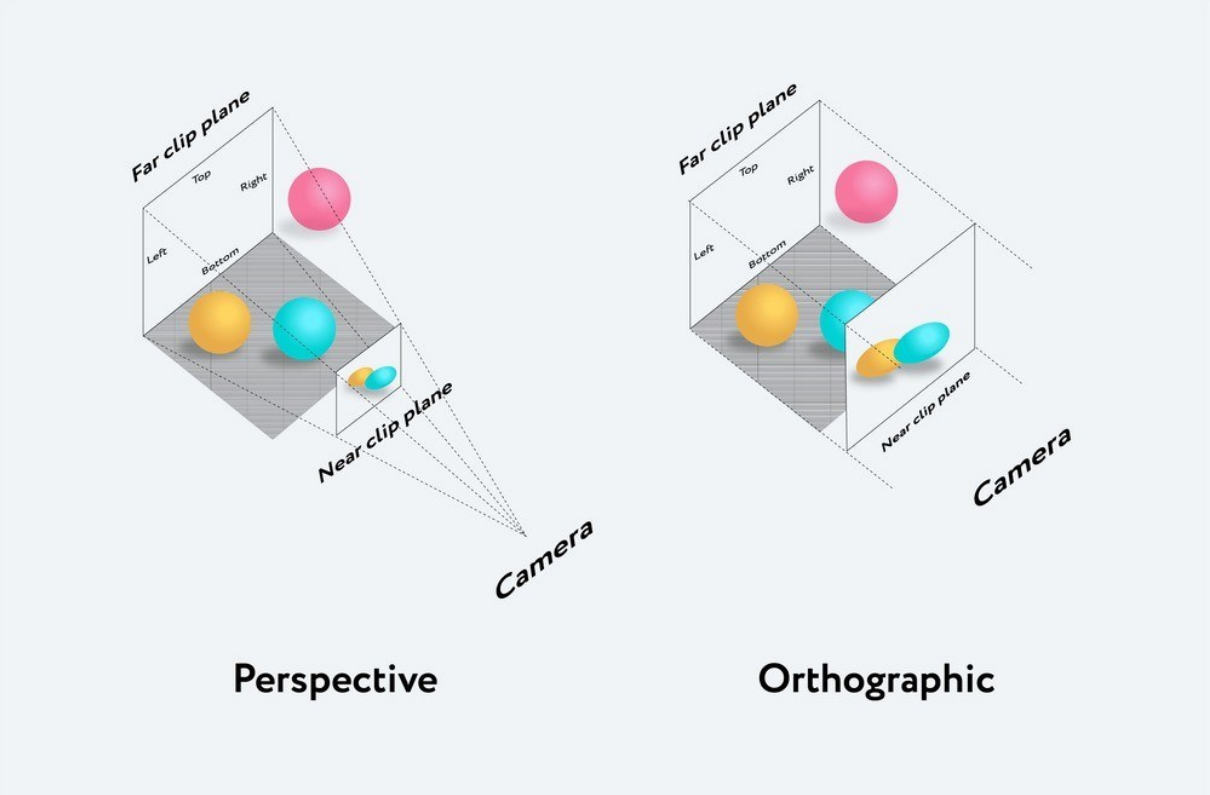
Scénu je v Three.js možné vytvořit pomocí třídy `Scene`. Při vytváření objektu není potřebný žádný parametr. Objekty do scény je možné přidat pomocí metody `add`, kde v parametru je uveden přidávaný objekt do scény. Stějně lze i objekty odebírat metodou `remove`. Případně lze také přidat do scény více objektů pomocí metody `add`, pouze tyto objekty je nutné oddělit v parametru čárkou. Pokud je objekt přidán do scény, jeho výchozí pozice ve scéně je na souřadnicích (0, 0, 0). [17]

2.6 Kamera

Three.js obsahuje dva základní typy kamer, a to ortografickou kameru (třída `OrtograpicCamera`) nebo perspektivní kameru (třída `PerspectiveCamera`). Perspektivní kamera zachovává perspektivu scény, a tak pokud ve scéně jsou umístěny objekty v různé vzdálenosti od kamery, vzdálenější objekty budou vykresleny jako menší a objekty blíže kameře budou vykresleny jako vetší. Při použití ortografické kamery nezáleží, jak daleko leží objekty ve scéně od kamery a jsou vykreslovány všechny ve stejné velikosti. [18 s. 57-59]

Pro člověka je přirozené vnímat okolní svět perspektivně, a proto se i v Three.js častěji využívá perspektivní kamera.

Pro ovládání kamery lze v three.js použít objekty několika tříd, například třídu `OrbitControls`. [11, s. 32]



***Obrázek 5**: Porovnání ortografické a perspektivní kamery (zdroj [19])*

2.6.1 Perspektivní kamera

Záběr scény perspektivní kamerou je definován komolým čtyřbokým jehlanem. Při vytváření objektu perspektivní kamery (třída `PerspectiveCamera`) je potřebné určit čtyři parametry, pomocí kterých lze ovlivnit tvar jehlanu. [18, s. 60-61]

Parametr `fov` určuje zorné pole kamery, pod tímto pojmem si lze představit úhel jisté výseče ze scény. Hodnota parametru je zadávána ve stupních a jeho defaultní hodnota je 45 stupňů. Dalším parametrem je `aspect`, který definuje, v jakém poměru bude délka a šířka podstavy jehlanu. Nejčastěji se využívá podíl délky a šířky okna. Parametr `near` určuje v jaké vzdálenosti od kamery začne být scéna vykreslována. Parametr `far` naopak určuje, do jaké vzdálenosti od kamery bude scéna vykreslována. [18, s. 60-61]

2.6.2 Ortografická kamera

Záběr scény ortografickou kamerou je určen kvádrem. Při vytváření objektu ortografické kamery (třída `OrthographicCamera`) je potřebné pomocí šesti parametrů určit rozměry a pozici kvádru. [18, s. 61-62]

Parametry `left`, `right`, `top` a `bottom` určují vzdálenosti daných stran obdelníku od středu obdelníku (kamery). Pokud například nastavíme vlastnost `left` na hodnotu 200, tak objekty

ležící dále než 200 jednotek od kamery nebudou vykresleny. Pomocí těchto parametrů jsou definovány dva shodné obdélníky, kde první je umístěn do vzdálenosti dle parametru `near` od kamery a druhý do vzdálenosti dle parametru `far` od kamery. Scéna která leží mezi těmito dvěma obdélníky (v kvádru) je vykreslena. [18, s. 61-62]

2.7 Souřadnicový systém

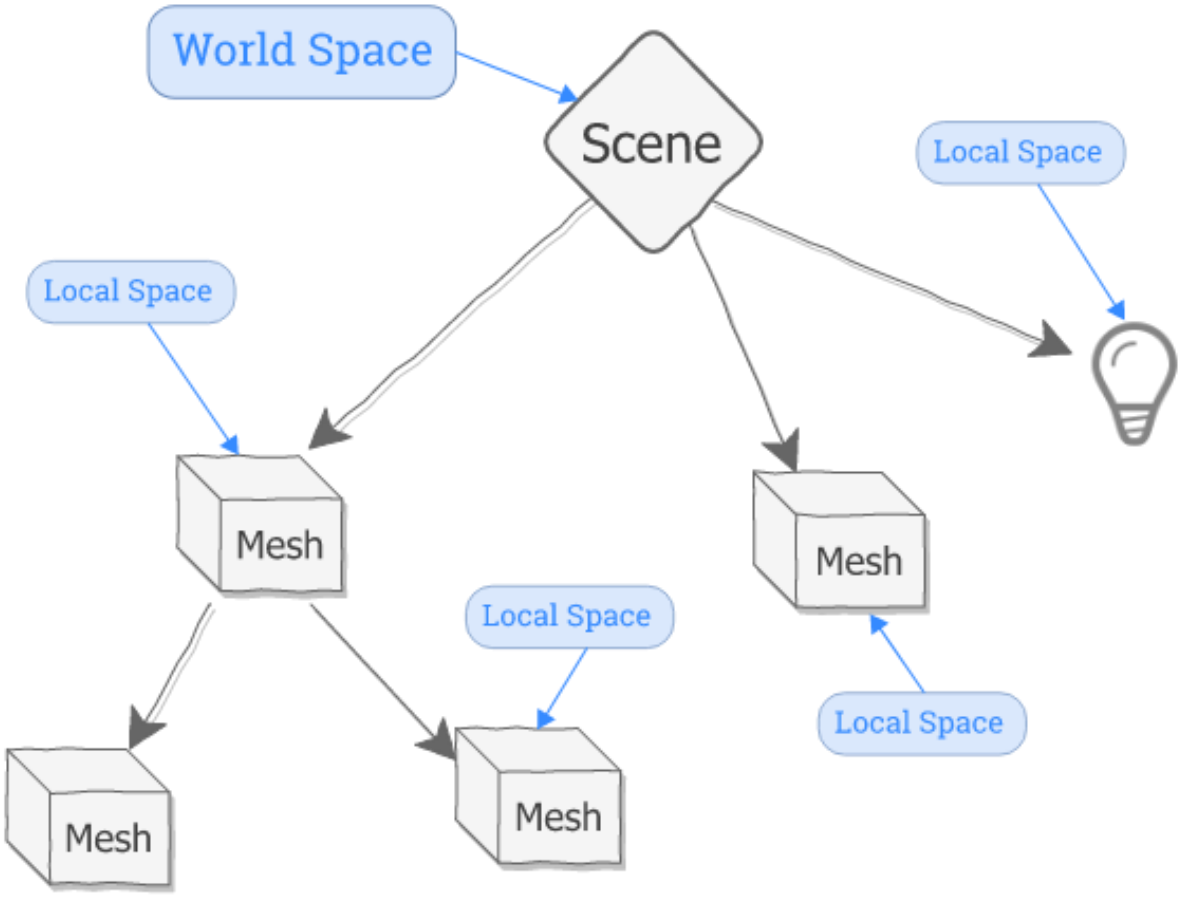
Ve 3D grafice nejpoužívanějším souřadnicovým systémem je kartézský souřadnicový systém, kdy i Three.js využívá tohoto souřadnicového systému. Kartézský souřadnicový systém se skládá ze tří os (X, Y, Z). K průniku těchto os dochází v bodě (0, 0, 0), který se nazývá počátek. Zde se osy protínají a jsou k sobě vzájemně kolmé. Díky tomuto systému lze jednoznačně určit polohu objektu v prostoru. Výsledná poloha objektu je určena vektorem o složkách X, Y a Z. [20]

2.7.1 World Space

Pod tímto pojmem si lze představit samotný prostor scény . Word Space je kartézský souřadnicový systém. Výsledné vykreslení objektů ve scéně je určeno pozicí v tomto prostoru. [20]

2.7.2 Local Space

Local Space je prostor, který je definován v rámci každého objektu, který je vložen do scény. Tento prostor je také kartézským souřadnicovým systémem. Pro přidávání a odebírání objektů do Local Space je využito stejných metod jako u World Space (Scény), jen jsou volány u daného objektu. Pokud tedy je například posunut nějaký objekt A do jehož Local Space byl vložen další objekt, dojde k posunutí obou těchto objektů. [20]



***Obrázek 6**: Obrázek grafu scény (zdroj [20])*

2.7.3 Základní transformace v Three.js

Mezi základní transformace patří změna pozice, rotace a měřítka. Objekty ve scéně nesou vlastnosti, pomocí kterých lze tyto transformace realizovat. V Three.js se pro změnu pozice využívá vlastnost `position`, pro nastavení rotace je zde vlastnost `rotation` a pro změnu měřítka vlastnost `scale`. Hodnota rotace je v Three.js zadávána v radiánech. Každou z vlastností je možno měnit ve směru X, Y a Z. Nastavit tyto vlastnosti lze samostatně nebo všechny najednou metodou `set`. [20]

Nastavení všech vlastností najednou u objektu camera: `camera.position.set(5, 5, 5);`

Nastavení vlastnosti pouze ve směru X u objektu camera: `camera.position.x = 5;`

2.8 Barvy

Color je třída, která v Three.js reprezentuje barvu. Třída Color má několik konstruktorů. Bezparametrický konstruktor použije jako výchozí barvu bílou. Dále je možné použít parametrický konstruktor, kde v parametru je možné zapsat barvu několika způsoby. První

možností je zapsat barvu pomocí šestnáctkové soustavy s prefixem 0x. Třetí možností je využití RGB nebo HSL textového řetězce. Další možností je zapsat barvu pomocí názvu z tabulky X11. Poslední možností je využít konstruktor, který požaduje 3 parametry. Kdy každý reprezentuje hodnotu R, G nebo B (nabývá hodnot od 0.0 do 1.0). [21]

Možnosti konstruktorů třídy Color: [21]

- `new THREE.Color(); // bílá barva`
- `new THREE.Color(0x000000); // černá barva`
- `new THREE.Color("rgb(0, 255, 0)"); // zelená barva`
- `new THREE.Color("rgb(0%, 100%, 0%)"); // zelená barva`
- `new THREE.Color("hsl(240, 100%, 50%)"); // modrá barva`
- `new THREE.Color( 'aqua' ); // azurová barva`
- `new THREE.Color( 0.5, 0.5, 0.5 ); // šedá barva`

2.9 Materiály

Materiály slouží k určení vzhledu objektu ve scéně. Materiál je možné vytvořit pomocí bezparametrického konstruktoru, nebo pomocí parametrického konstruktoru a určit tak požadované vlastnosti materiálu. Barvu materiálu je možně učit více způsoby viz kapitola 2.8. Dokonalost materálu vyžaduje vyšší výkon. [22]

Materiály knihovny Three.js je možné realializovat těmito třídami: [22]

MeshBasicMaterial

Dle názvu je tento materiál nejzákladnější v knihovně Three.js, ale také nejméně náročný na výkon. Jako jediný z uvedených materiálů nevyžaduje světla ve scéně. Materiál je vykreslen pouze barvou, která mu byla nadefinována.

MeshLamberMaterial

Je materiál vyžadující osvětlení, které je počítáno pro vrcholy.

MeshPhongMaterial

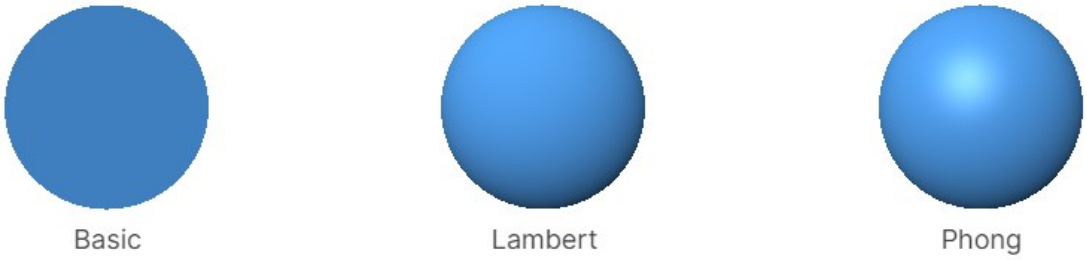
Tento materiál vyžaduje také světla ve scéně, osvětlení je ale počítáno zvlášť pro pixely. U tohoto materiálu lze navíc určit jeho lesklost (vlastnost `shininess`).

MeshToonMaterial

Je materiál, který pro stínování využívá přechodovou mapu. Na základě této mapy je vykresleno finální stínování pomocí dvou barev (cartoon vzhled).

MeshPhysicalMaterial

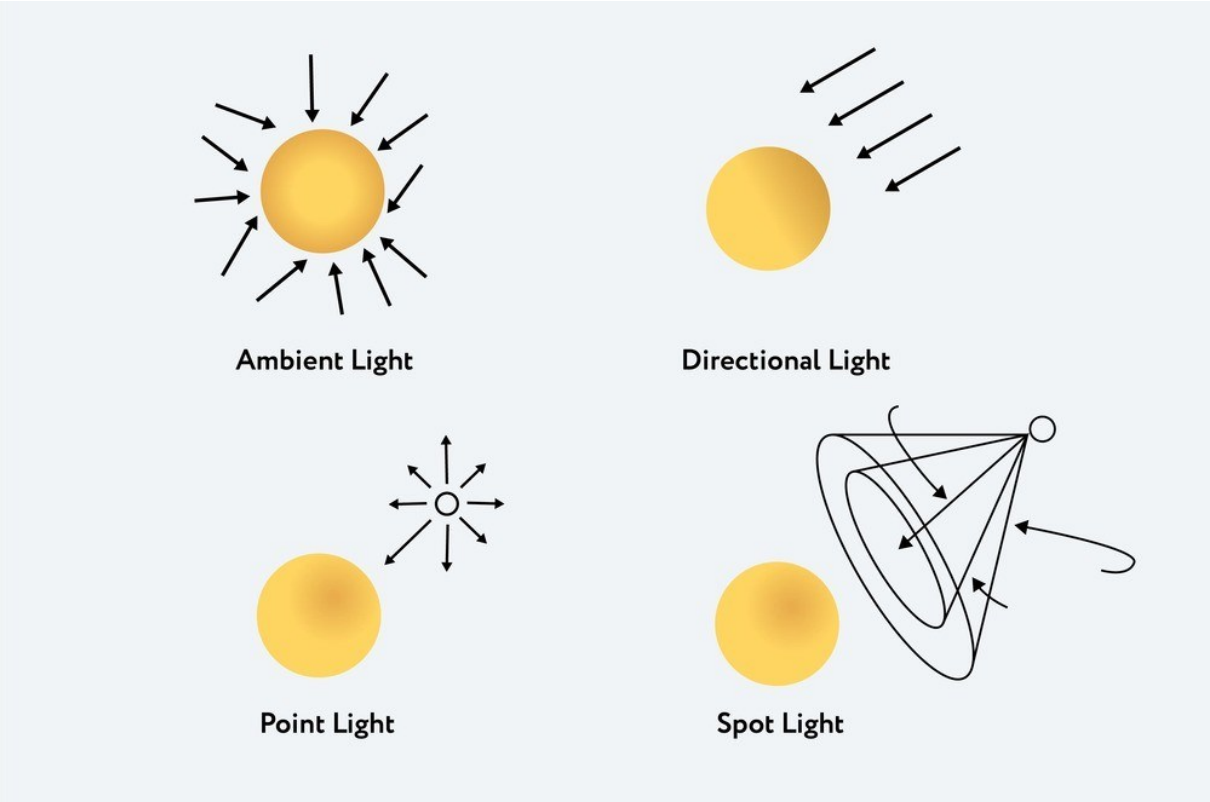
Řadí se mezi PBR materiály, které jsou cílené na to, aby působily co nejvíce realisticky. Realistického vzhledu je docíleno pomocí složitých matematických výpočtů.



***Obrázek** 7: Porovnání vzhledu základních materiálů (zdroj [22])*

2.10 Světla

V knihovně Three.js je možné vytvořit několik typů světel. U každého typu světla je potřebné nastavit barvu světla a jeho intenzitu (vlastnosti `color` a `intensity`). [23]



***Obrázek 8**: Světla v Three.js (zdroj [19])*

Světla je možné realizovat pomocí následujících tříd: [23]

AmbientLight

Základní typ světla je možné realizovat pomocí třídy AmbientLight. Výsledná barva ve scéně je určena vynásobením barvy materiálu, barvy světla a intenzity světla. Objekty ve scéně působí ploše (nejsou stínované) a jejich barva je zkreslená tímto světlem.

HemisSphereLight

Světlo vytvořené pomocí této třídy je určeno pomocí dvou barev (vlastnosti `skyColor` a `groundColor`). Výsledná barva ve scéně je určena vynásobením barvy materiálu s barvami definovanými jako `skyColor`, `nebo groundColor` podle toho, kam daná plocha objektu směřuje. I zde objekty ve scéně působí plochým dojmem.

DirectionalLight

Pomocí této třídy lze vytvořit směrové světlo. Vlastnostmi `position` a `target` je určen směr záření světla, ale není definován přímo bod ze kterého světlo vychází.

PointLight

Je třída pomocí které lze ve scéně vytvořit bodový zdroj světla. Vlastností `position` je určen bod ze kterého světlo vychází a vlastnost `distance` určuje do jaké vzdálenosti bude světlo svítit (všemi směry).

SpotLight

Pomocí této třídy je záření světla definováno kuželem, kde zdroj světla se nachází v jeho vrcholu. Pro umístění a nasměrování kuželu je nutné nastavit vlastnosti `position` a `target`. Výška kuželu je nastavena pomocí vlastnosti `distance` a určuje kam až světlo bude svítit. Pro nastavení velikosti podstavy kuželu slouží vlastnost `angle`. Vlastnost `penumbra` určuje slábnutí světla od středu kuželu. Takovéto světlo si lze představit například svícením baterkou ve tmě.

RectAreaLigth

U této třídy je zdroj světla definován obdelníkem a může tak působit jako například zářivka z realného prostředí. U obdelníku jsou určeny jeho rozměry (vlastnosti `width` a `height`) a také jeho natočení (vlastnost `rotation`).

2.11 Vytvoření těles

Pro demonstraci bude vytvářena krychle. Nejprve je nutné definovat vrcholy daného tělesa a určit tak jeho tvar. Pro krychli je vytvořen objekt třídy `BoxGeometry`, kde v konstruktoru lze nastavit rozměry daného tělesa. V knihovně Three.js existuje několik tříd pro definování vrcholů základních těles. Dále je nutné vytvořit materiál pro definování vzhledu daného tělesa. Materiál a tvar tělesa jsou sloučeny pomocí objektu třídy `Mesh`, kde v konstruktoru je jako první parametr uveden tvar a jako druhý materiál. Objekt lze následně přidat do scény a nechat ho tak vykreslit. S objektem lze následně provádět dále různé transformace. [12]

```
var cubeGeometry = new THREE.BoxGeometry( 20, 20, 5 );
var cubeMaterial = new THREE.MeshBasicMaterial( { color: 0x33cc33 } );
var cube = new THREE.Mesh( cubeGeometry, cubeMaterial );
scene.add( cube );
cube.position.y = 20;
```

***Zdrojový kód 3**: Vytvoření krychle v Three.js (zdroj vlastní)*

3 MOŽNOSTI IMPLEMENTACE HUD PRVKŮ POMOCÍ THREE.JS

První možností je využití HTML dokumentu a zobrazit HUD prvky pomocí HTML elemetů před samotnou scénou Three.js. Druhou možností je využít objektu kamery v Three.js scéně a vykreslovat HUD prvky uživateli těsně před objektem kamery.

3.1 Implementace pomocí HTML elementů

Při zobrazování HUD prvků pomocí HTML elementů je vhodné tento prvek nejprve vytvořit v těle HTML dokumentu a následně ho nastylizovat pomocí CSS stylů do požadovaného vzhledu a umístění. HUD prvek se umisťuje do běžného kontejneru. Často je realizován pomocí různě nastylizovaných textů, nebo obrázkem s průhledným pozadím (například ve formátu PNG nebo SVG). Je vhodné těmto HTML elementům nastavit atribut `id`, aby je bylo možné dále využívat ve zdrojovém kódu JavaScriptu (např. změna polohy HUD prvku, nebo obsahu textu HUD prvku).

V samotném kódu JavaScriptu lze následně získat lehce přístup k elementům HTML dokumentu. Pokud element nese atribut `id` nebo atribut `class`, je možné ho pomocí tohoto atributu dále využívat v kódu JavaScriptu jako jiné proměnné pomocí tečkové notace, například `compass.style.transform = 50`. [24]

Tomuto elementu lze tedy dále měnit různé vlastnosti, nejspíše nejčastěji měněnou vlastností při tvorbě HUD prvků, je vlastnost `style`, kdy při různých událostech ve scéně dochází k modifikaci vzhledu HUD prvku.

Události ve scéně lze zachytit pomocí různých posluchačů. [24]

Pomocí HTML elementů lze HUD prvek zobrazit také pomocí více elementů `canvas`. Například lze na jeden element `canvas` vykreslovat hlavní pohled kamerou a na druhý element `canvas` vykreslovat pohled druhé kamery hledící na scénu shora za účelem vykreslení minimapy. Podobného výsledku lze dosáhnout vykreslováním na jeden element `canvas` s použitím výřezů elementu `canvas`. Tento výřez lze nastavit vykreslovači pomocí metody `setViewport`.

```
<!DOCTYPE html>
<html lang="en">
    <head>
        <meta charset="UTF-8"/>
        <meta name="viewport" content="width=device-width,
```

```
     initial-scale=1.0"/>
    <meta http-equiv="X-UA_Compatible" content="ie=edge"/>
    <title>Three.js Compass</title>
    <style>
        body{
            margin: 0;
        }
        canvas{
            width: 100%;
            height: 100%;
        }
        #startBtn{
            position: absolute;
            color: crimson;
        }
        /*kontejner pro obrázky tvořící HUD kompas*/
        #compassBox{
            position: absolute;
            top: 40px;
            left: -50px;
            /*
            nastavení vlastnosti clip-path,
            aby byla viditelná jen výseč obrázku kompasu
            přibližný tvar výseče
              _________
              \       /
                ------
            */
            clip-path: polygon(
                20% 0,
                80% 0,
                65% 30%,
                35% 30%);
        }
        #compass{
            /*průhlednost obrázku kompasu*/
            opacity: 0.6;
        }
        #znacka{
            /*průhlednost zelené značky ukazující aktuální směr*/
            opacity: 0.7;
        }
    </style>
</head>
<body>
    <!--obrázek s šipkou-->
    <div id="compassBox">
        <img id="znacka" src="arrow_final.png" width="500">
    </div>
```

```
        <!--obrázek s kruhovým kompasem-->
        <div id="compassBox">
            <img id="compass" src="compass_final.png" width="500">
        </div>

        <button id="startBtn">START</button>
        <script src="main.js"></script>
    </body>
</html>
```

Zdrojový kód 4: *Ukázka vytvoření kompasu pomocí HTML elementů (zdroj vlastní)*

Obrázek 9*: HUD prvek vytvořený v kódu výše (zdroj vlastní)*

3.2 Implementace pomocí objektu kamery

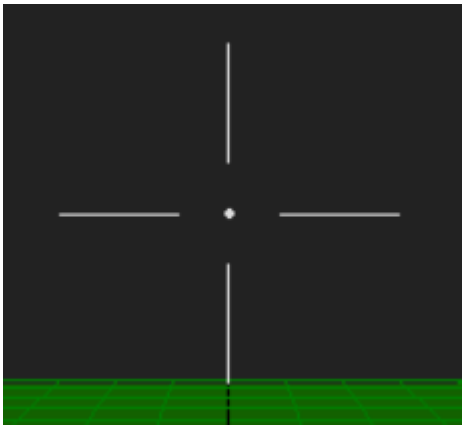
Pro zobrazení HUD prvku pomocí kamery je nejprve nutné vytvořit objekt kamery. Následně vytvořit požadovaný HUD prvek, například v podobě různé kombinace geometrických tvarů. HUD prvek je následně nutné umístit těsně před kameru. Toho lze docílit nastavením vlastnosti `postion` a předáním prvku kameře pomocí metody `add`. HUD prvek se tak bude pohybovat s kamerou a hráč ho uvidí na jednom místě obrazovky. Pokud by byl prvek umístěn příliš daleko od kamery, při přiblížení kamery k nějakému objektu ve scéně by mohlo docházet k nechtěnému překrývání HUD prvku objektem ve scéně. Výslednou velikost HUD prvku před kamerou lze ještě následně vyladit pomocí nastavení měřítka, k těmto účelům poslouží vlastnost `scale` u objektu HUD prvku.

```
var scene = new THREE.Scene();
var camera = new THREE.PerspectiveCamera(
    75,
    window.innerWidth/window.innerHeight,
    0.1,
    1000
);

scene.add(camera);
```

```
var renderer = new THREE.WebGLRenderer();
renderer.setSize( window.innerWidth, window.innerHeight );
document.body.appendChild( renderer.domElement );
renderer.setClearColor(0x222222, 1.0);

var controls = new THREE.PointerLockControls( camera, renderer.domElement );

camera.position.set( 0, 50, 100 );

var mapSize = 500;

//mřížka
var grid = new THREE.GridHelper(mapSize, 30, "black", "green");
grid.position.y = 1;
scene.add(grid);

//zem
var floorPlane = new THREE.PlaneGeometry( mapSize, mapSize );
var floorMaterial = new THREE.MeshBasicMaterial( {color: 0x136100} );
var floor = new THREE.Mesh( floorPlane, floorMaterial );
floor.rotation.x = -90 * (Math.PI/180);
scene.add( floor );


//reticle

const distanceFromCamera = -1;
scale = 1/70;

function createLine(x1, y1, z1, x2, y2, z2, mat, name)
{
    //body úsečky reticlu
    const points = [];
    points.push( new THREE.Vector3( x1, y1, z1 ) );
    points.push( new THREE.Vector3( x2, y2, z2 ) );

    //tvar úsečky z bodů
    const reticleGeometry =
    new THREE.BufferGeometry().setFromPoints( points );

    //objekt ucečky
    const line = new THREE.Line( reticleGeometry, mat );

    //přidání úsečky objektu kamery

    line.position.set(0,0,distanceFromCamera);
    line.name = name;
    line.scale.set(scale, scale, scale);
```

```
    camera.add( line );
}

//material pro reticle
const reticleMaterial = new THREE.LineBasicMaterial(
    { color: 0xffffff, opacity: 0.85, transparent: true });

//vytvoření úseček (levá, pravá, horní, spodní)
createLine(-10, 0, 0,-3, 0, 0, reticleMaterial, "left");
createLine(3, 0, 0, 10, 0, 0, reticleMaterial, "right");
createLine(0, 3, 0, 0, 10, 0, reticleMaterial, "upper");
createLine(0, -3, 0, 0, -10, 0, reticleMaterial, "lower");

//vytvoření středového bodu reticlu
const geometryDot = new THREE.CircleGeometry( 0.3, 100 );
const circle = new THREE.Mesh( geometryDot, reticleMaterial );
circle.scale.set(scale, scale, scale);
circle.name = "dot";

camera.add(circle);
circle.position.set(0,0,distanceFromCamera);
```

***Zdrojový kód 5**: Ukázka vytvoření nitkového kříže pomocí objektu kamery (zdroj vlastní)*



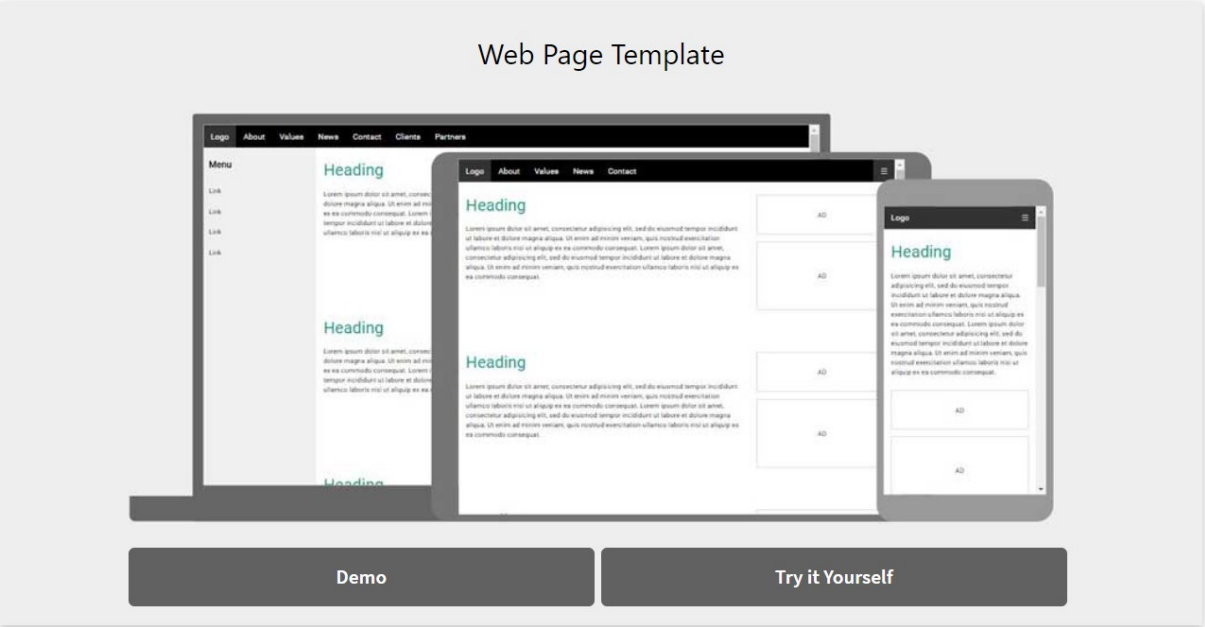
***Obrázek 10**: HUD prvek zaměřovače vytvořený v kódu výše (zdroj vlastní)*

4 PRAKTICKÁ ČÁST

4.1 Šablona pro webovou prezentaci

Pro tvorbu webové prezentace obsahující ukázky HUD prvků byla zvolena jedna z dostupných šablon CSS stylů pro tvorbu webových stránek nabízených na webu W3Schools. Zde je na výběr několik šablon určených pro tvorbu blogů, galerií nebo rezervačních stránek. Šablony jsou responzivní a jsou dostupné zdarma, včetně možnosti vlastní úpravy šablony. Šablonu si je možné prohlédnout pomocí editoru, ve kterém je možno si vyzkoušet různé modifikace šablony. [25]

Pro účely prezentace vytvořených ukázek HUD prvků byla převzata šablona s názvem Web Page Template. Šablona působí decentním dojmem a dle náhledu je vhodná pro vložení většího množství textu doplněného obrazovou přílohou. Postraní menu dobře poslouží pro realizaci navigace mezi jednotlivými ukázkami.



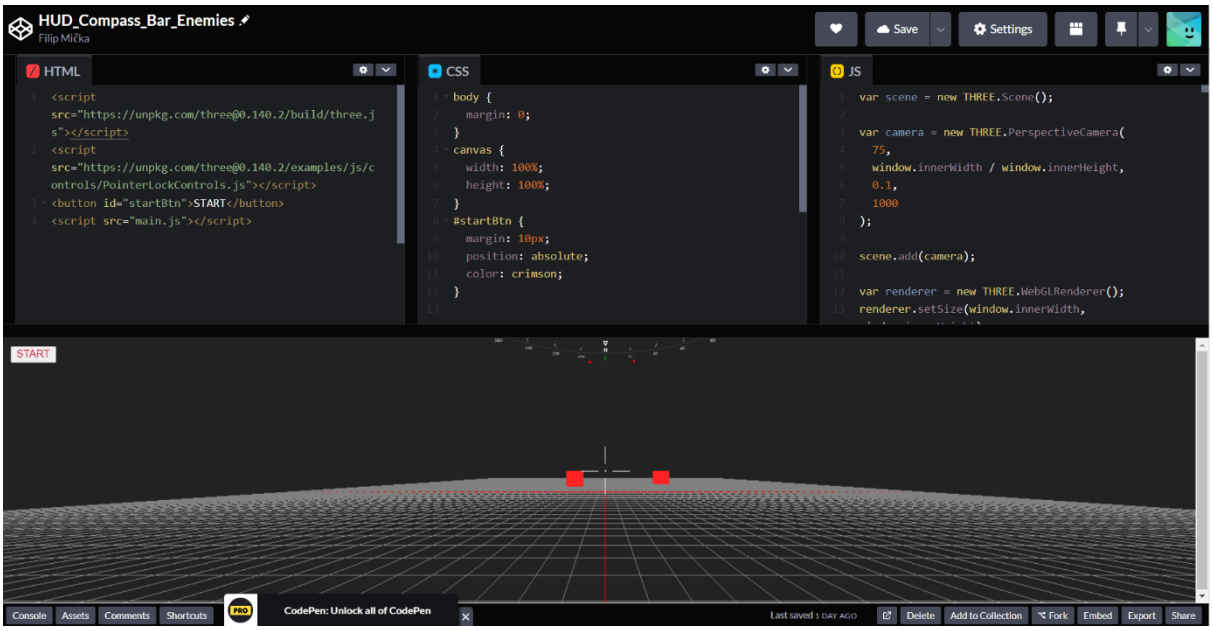
***Obrázek 11**: Náhled šablony Web Page Template (zdroj [25])*

4.2 Zpracování ukázek v prototypovacím prostředí

Prvotní tvorba ukázek byla realizována ve vývojovém prostředí Visual Studio Code. Aby bylo možné ukázky interaktivně prohlížet ve webové prezentaci, byl nutný jejich import do některého z prototypovacích prostředí. Pro tyto účely bylo zvoleno prototypovací prostředí CodePen.

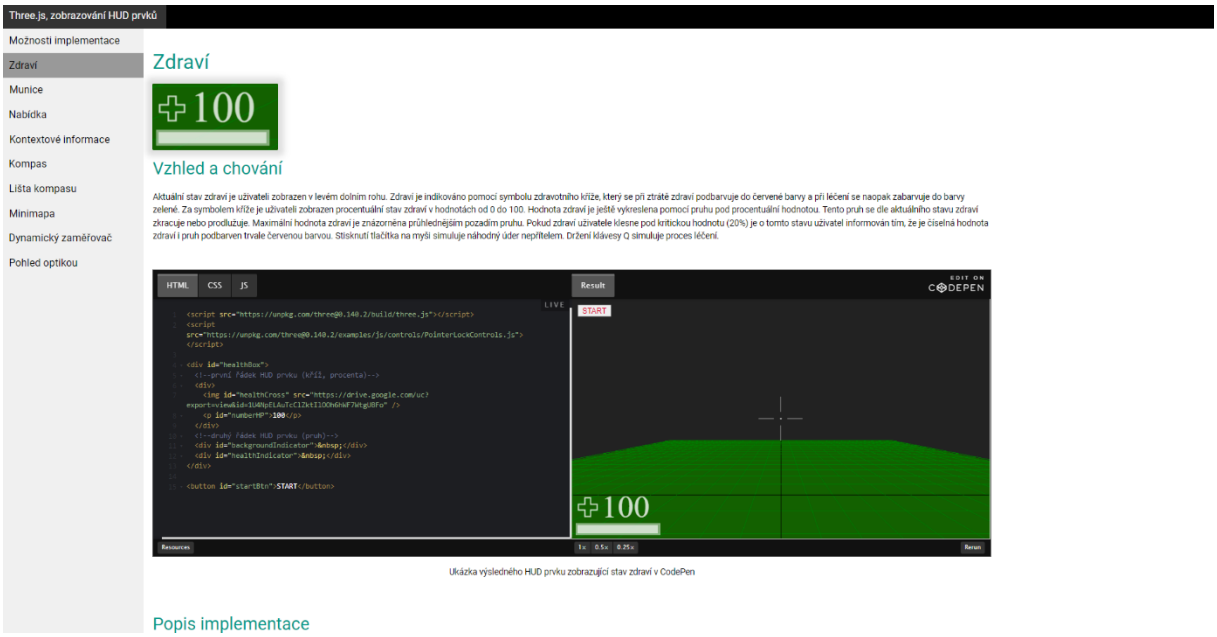
V tomto prostředí je možné realizovat projekty s využitím jazyků HTML, CSS a JavaScript. Je možné se bezplatně registrovat a následně využívat toto prototypovací prostředí. Dále je možné využít placenou verzi a získat tak některé z prémiových funkcionalit. [26]

Pro realizaci menších projektů je možné vytvářet tzv. Pens, které obsahují pouze 3 soubory (HTML, CSS a JavaScript). Vytvoření takového projektu je možné po přihlášení pomocí tlačítka v levém horním rohu hned pod logem CodePen. Následně je možné vkládat kód do textových editorů pro HTML, CSS a JavaScript v horní polovině okna. Editor pro HTML kód slouží pro implementaci sekce body, nikoliv celého HTML dokumentu. V dolní polovině okna se nachází náhled výsledné aplikace. Editor nabízí několik užitečných nastavení, jako například automatické formátování kódu při ukládání. [26]



***Obrázek 12**: Zpracování ukázky HUD prvku v CodePen (zdroj vlastní)*

Projekt lze následně sdílet pomocí tlačítka embed v pravém dolním rohu. Zde lze vygenerovat HTML kód, díky kterému lze interaktivní ukázku i s editory pro HTML, CSS a JavaScript umístit na vlastní webovou stránku. Ostatní uživatelé si tak mohou výslednou aplikaci zobrazit včetně zmíněných editorů a zkoušet si tak například různé modifikace kódu.



***Obrázek 13**: Interaktivní ukázka umístěna na vlastní webovou stránku (zdroj vlastní)*

Samotný import z vývojového prostředí Visual Studio Code byl jednoduchý, stačilo zkopírovat zdrojové kódy ukázek a vložit do editorů v CodePen. Pouze bylo nutné změnit cestu k obrázkům použitých v ukázkách. Obrázky byly umístěny na Disk Google.

4.3 Výsledná podoba webové prezentace

Prezentace je zasazena do zmiňované šablony, která byla lehce poupravena do výsledné podoby. Je vytvořena pomocí jazyků HTML, CSS a JavaScript. V levém horním rohu se nachází název webové stránky. Pod názvem webové stránky je umístěno menu, ve kterém je možnost přechodu na jednotlivé ukázky HUD prvků. První položka menu obsahuje popsání možností tvorby HUD prvků v knihovně Three.js.

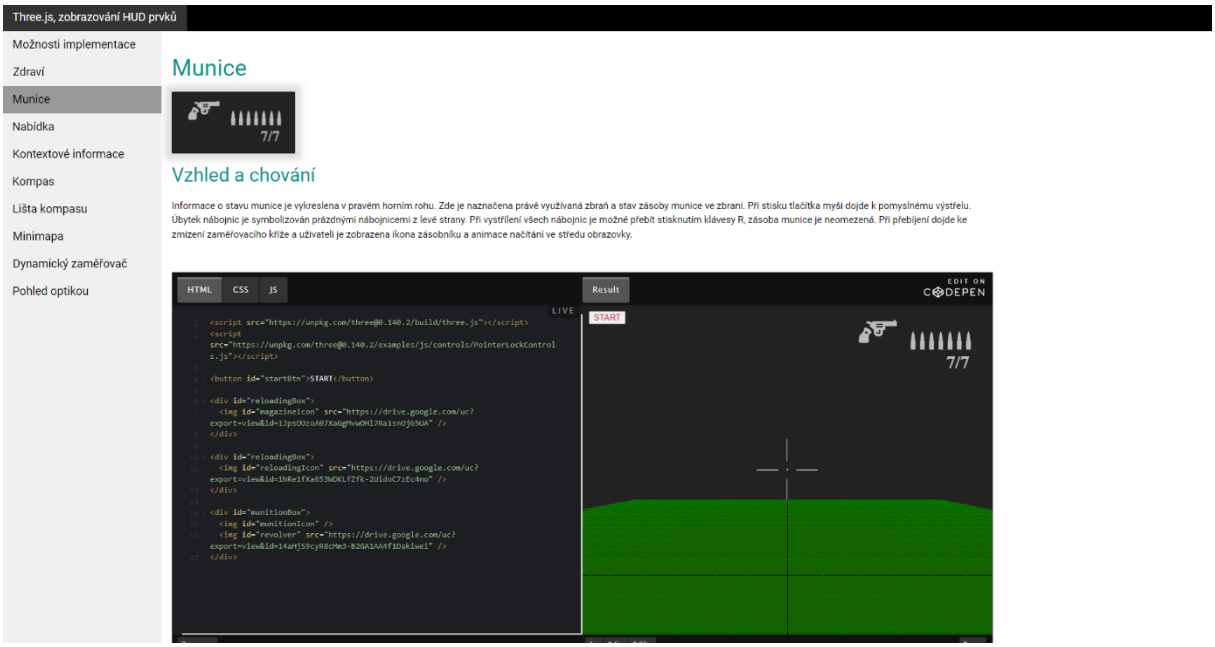
Jednotlivé stránky s ukázkami HUD prvků mají jednotný styl a jsou rozděleny do několika částí.

První část stránky je tvořena názvem HUD prvku s malým obrázkem sloužícím jako náhled konkrétního HUD prvku.

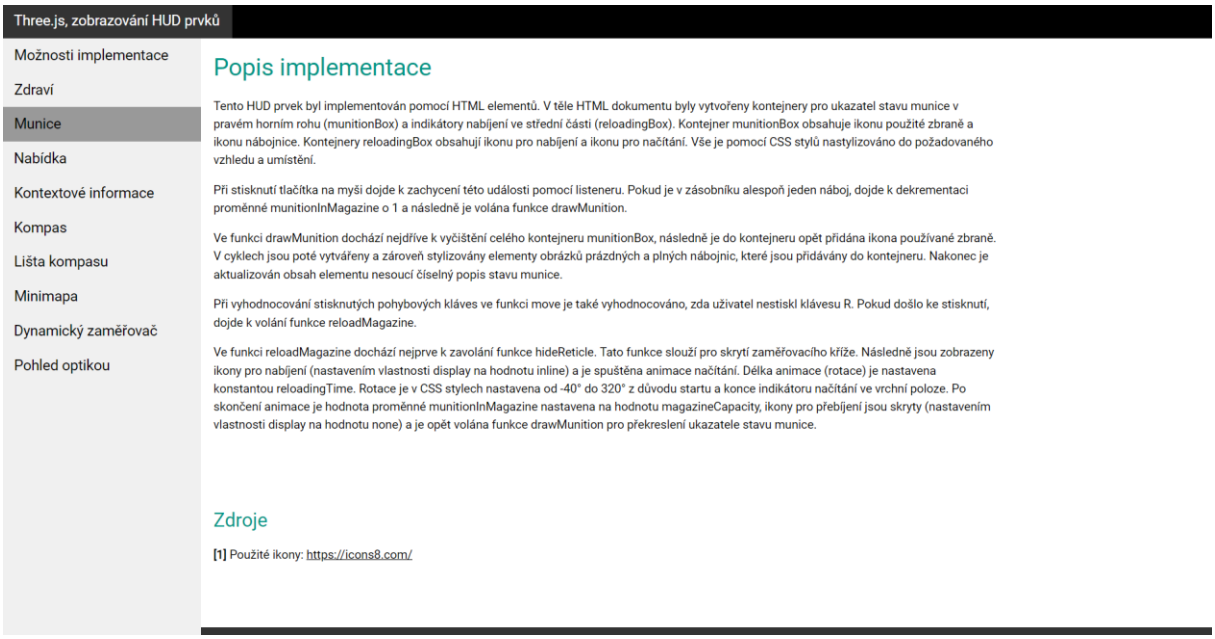
V další části stránky se nachází odstavec textu, kde je popsán vzhled a chování konkrétní ukázky HUD prvku. Pod tímto odstavcem se nachází interaktivní ukázka HUD prvku v prototypovacím prostředí CodePen. Stěžejní části kódu jsou zde komentovány tak, aby se v nich uživatel lehce orientoval.

Pod ukázkou HUD prvku je pomocí několika odstavců popsán detailněji postup implementace pro snadné pochopení, jak byla ukázka realizována. Některé ukázky jsou v této části také doplněny nákresy pro lepší představu popisované problematiky.

V poslední části stránky jsou uvedeny zdroje, které byly použity při implementaci ukázek (převzaté a různě modifikované části kódu nebo obrázky). Je zde vždy uvedeno pro co byl daný zdroj v ukázce použit s URL adresou daného zdroje. Při nevyužití cízích zdrojů tato část chybí.



***Obrázek 14**: Webová prezentace – první část (zdroj vlastní)*



***Obrázek 15**: Webová prezentace – druhá část (zdroj vlastní)*

4.4 Ukázka tří částí webové prezentace

4.4.1 Zdraví

Vzhled a chování:

Aktuální stav zdraví je uživateli zobrazen v levém dolním rohu. Zdraví je indikováno pomocí symbolu zdravotního kříže, který se při ztrátě zdraví podbarvuje do červené barvy a při léčení se naopak zabarvuje do barvy zelené. Za symbolem kříže je uživateli zobrazen procentuální

stav zdraví v hodnotách od 0 do 100. Hodnota zdraví je ještě vykreslena pomocí pruhu pod procentuální hodnotou. Tento pruh se dle aktuálního stavu zdraví zkracuje nebo prodlužuje. Maximální hodnota zdraví je znázorněna průhlednějším pozadím pruhu. Pokud zdraví uživatele klesne pod kritickou hodnotu (20 %) je o tomto stavu uživatel informován tím, že jsou číselná hodnota zdraví i pruh podbarveny trvale červenou barvou. Stisknutí tlačítka na myši simuluje náhodný úder nepřítelem. Držení klávesy Q simuluje proces léčení. To lze realizovat také v zeleném boxu.

***Obrázek 16**: HUD prvek zobrazující informaci o zdraví (zdroj vlastní)*

Popis implementace:

Pro implementaci HUD prvku, který informuje o aktuálním stavu zdraví uživatele, byl zvolen přístup implementace pomocí HTML elementů.

V těle HTML dokumentu byl vytvořen kontejner (`healthBox`), který obsahuje ikonu zdravotního kříže (`healthCross`), text nesoucí číselnou hodnotu zdraví (`numberHP`), prázdný kontejner sloužící pro vytvoření pruhu ukazatele zdraví s proměnnou délkou (`healthIndicator`) a druhý prázdný kontejner pro vytvoření pozadí pruhu (`backgroundIndicator`).

V kódu JavaScriptu je vytvořen listener na kliknutí myši. Při stisknutí tlačítka na myši dojde k vygenerování hodnoty náhodného úderu do proměnné `randomHit`. O tuto hodnotu je následně dekrementována proměnná `actualHealth`, která reprezentuje aktuální procentuální zdraví uživatele. Dále je provedena kontrola zda hodnota proměnné `actualHealth` neklesla pod 0. V případě že klesla, provede se korekce na hodnotu 0. Následně je bílý obrázek zdravotního kříže nahrazen červeným po dobu, která je nastavena ve funkci `setTimeout` a po uplynutí této doby je ikona kříže opět změněna za ikonu bílé barvy. Nakonec je volána funkce `drawHealth`. Změna ikon je provedena pomocí změny atributu `src` elementu obrázku (`healthCross`).

Ve funkci `heal` dochází k vyhodnocení , zda uživatel stiskl klávesu Q, nebo se nachází v boxu pro léčení. Při kladném vyhodnocení je nejdříve provedena kontrola zda hodnota zdraví uživatele (proměnné `actualHealth`) již nepřesáhla hodnotu 100. Pokud je hodnota zdraví nižší než 100 dojde k inkrementaci proměnné `actualHealth` o proměnnou `healthGrowCoeficient`. Následně je bílá ikona kříže změněna za zelenou a opět po uplynutí prodlevy nastavené pomocí funkce `setTimeout` je navrácena ikona bílá. Nakonec je volána funkce `drawHealth`.

Ve Funkci `drawHealth` je nejprve vyhodnoceno, zda hodnota zdraví je nižší než 20. Pokud je hodnota nižší než 20, je nastylizován pruh ukazatele zdraví a textová hodnota zdraví do červené barvy (kritická hodnota). Pokud hodnota není nižší než 20, je pruh i textová hodnota bílá. Pruh ukazatele zdraví má v CSS stylech definovanou velikost 200 px. Pomocí nastavení vlastnosti `clipPath` je pruhu (`healthIndicator`) nastavena jeho procentuální viditelná délka dle aktuální hodnoty proměnné `actualHealth`. Nakonec je aktualizováno číslo zobrazující procentuální stav zdraví (`numberHP`) pomocí změny atributu `innerHTML`.

```
<div id="healthBox">
            <!--první řádek HUD prvku (kříž, procenta)-->
            <div>
                <img id="healthCross" src="/images/kriz.png"/>
                <p id="numberHP">100</p>
            </div>
            <!--druhý řádek HUD prvku (pruh)-->
            <div id="backgroundIndicator"> </div>
            <div id="healthIndicator"> </div>
</div>
```

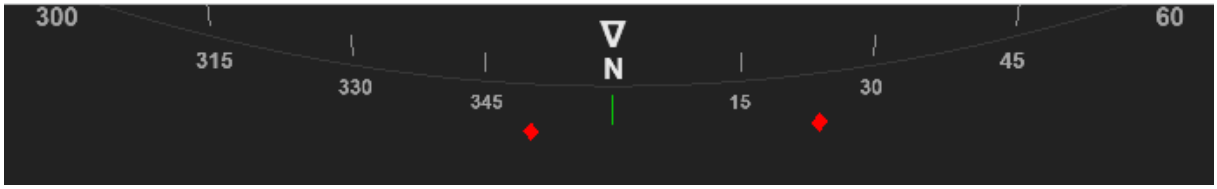
***Zdrojový kód 6**: HUD prvek reprezentující zdraví vytvořený pomocí HTML elementů (zdroj vlastní)*

4.4.2 Lišta kompasu

Vzhled a chování:

Lišta kompasu je umístěna ve středu horní části okna. Na horní části lišty se nachází znaky N, S, W a E. Tyto znaky reprezentují světové strany. Mezi těmito znaky je lišta dále rozdělena na dílky, které jsou označeny pomocí čísel pod nimi. Tato čísla slouží pro přesnější určení směru pohledu hráče a nabývají hodnot od 0 do 360, kdy hodnoty 0, 90, 180 a 270 jsou zastoupeny zmíněnými znaky světových stran. Hodnoty jsou stupňovány po 15. V nejvrchnější části lišty se nachází trojúhelníkový ukazatel indikující aktuální směr pohledu. Na liště jsou dále umístěny dvě červené značky, které zobrazují aktuální polohu červených krychlí ve scéně. Při pohybu kamerou značka ukazuje vždy směr, ve kterém se nepřátelé (krychle) nachází. Pokud dojde

k situaci, kdy je nepřítel mimo pohled kamery, jeho značka zůstává v rohu lišty tak, aby uživatel měl povědomí o poloze nepřítele.



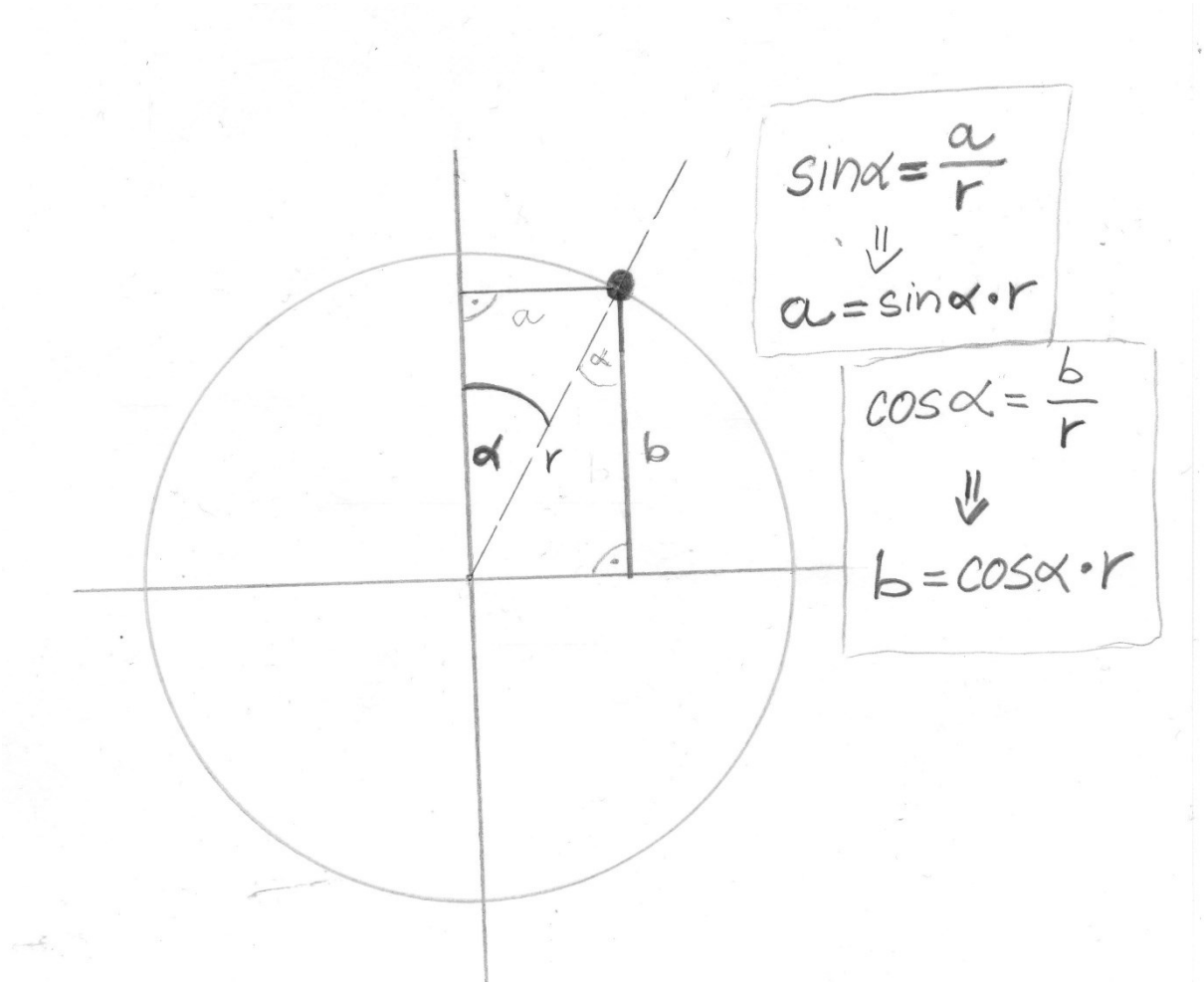
***Obrázek 17**: HUD prvek zobrazující lištu kompasu (zdroj vlastní)*

Popis implementace:

Lišta kompasu byla realizována pomocí objektu kamery ve scéně v kombinaci s geometrickým útvarem kružnice, která je umístěna nad kamerou. Tato kružnice je dále doplněna o úsečky, které znázorňují stupně.

Nejprve byla vytvořena samotná kružnice. Velikost kružnice je dána proměnnou `circleRadius`. Při běžném vytvoření kružnice pomocí `CircleGeometry` je uvnitř kružnice vidět její segmentace. V tomto případě se hodí spíše kružnice bez viditelné segmentace, proto je nutné ji vytvořit pomocí bodů této kružnice, ze kterých je následně vytvořena křivka dané kružnice.

Pomocí funkce `createLineOnCompass` byly vytvořeny hlavní zelené úsečky na kružnici. Následně v cyklu byly vytvořeny úsečky vedlejší segmentace, které jsou zbarveny bíle. V cyklu je zahrnuta podmínka, aby nedošlo k vytvoření této vedlejší segmentace kružnice přes hlavní segmentaci. Poloha úsečky je určena pomocí výpočtu polohy bodu na kružnici.



***Obrázek 18**: Nákres principu výpočtu polohy bodu na kružnici (zdroj vlastní)*

S těmito úsečkami jsou dále na kružnici vytvářeny nápisy v podobě číslic označující stupně a také zkratky světových stran (N, S, E, W). Nápisy jsou realizovány pomocí objektů třídy `Sprite` (popsáno v kapitole o kontextových informacích) a vytvářeny ve funkci `createTextSprite`.

Vše je následně předáno metodou `add` objektu kamery a umístěno do pozice nad kameru.

Pomocí funkce `createTextSprite` je také objektu kamery předán znak Δ, který je otočen o 180 stupňů, aby působil jako šipka ukazující na hodnoty stupňů na liště.

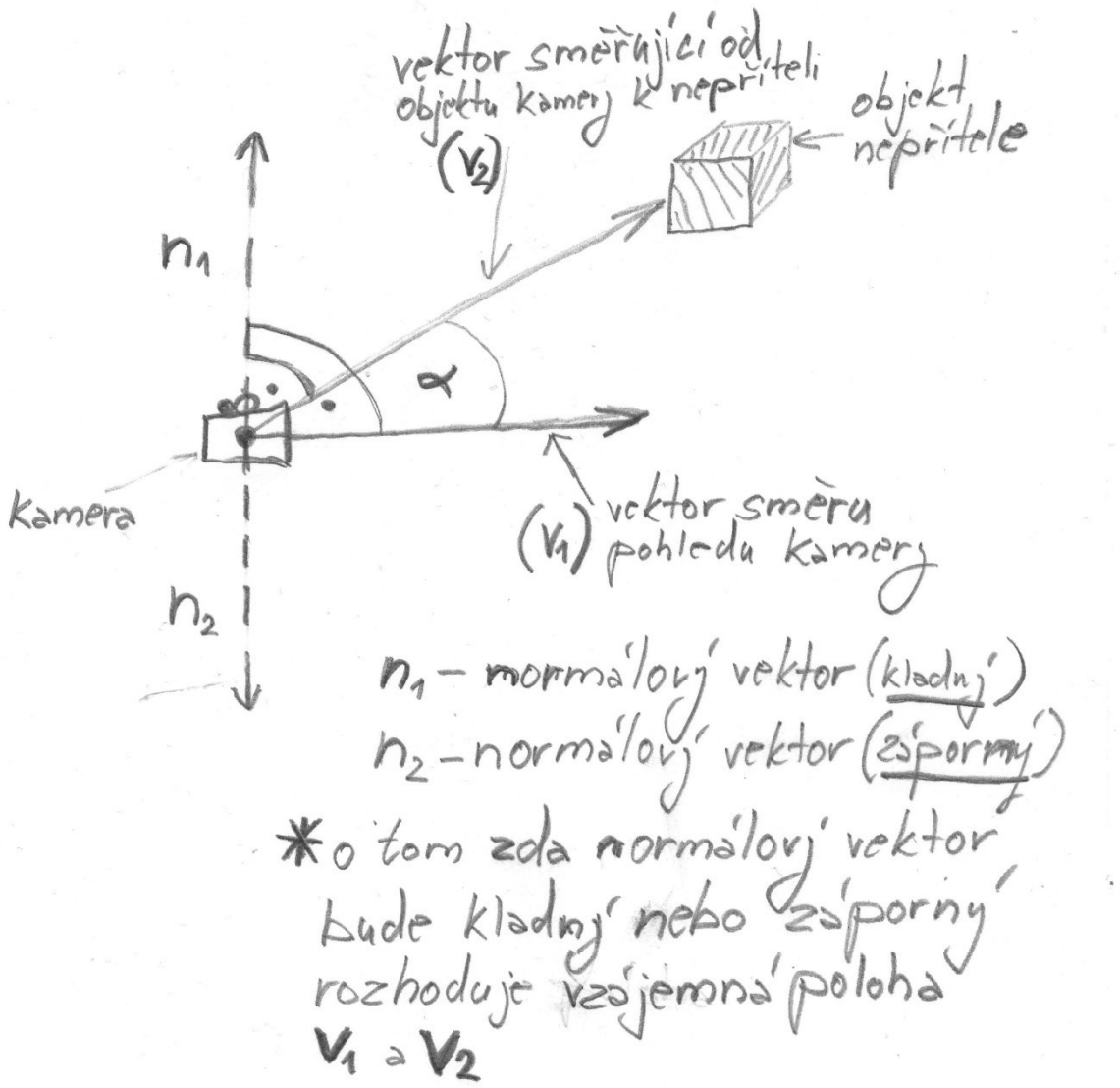
Ve funkci `animate` je při každém renderování snímku nastavena kružnici rotace dle aktuálního směru pohledu kamery. Princip funkce `calculateAngle` je vysvětlen v kapitole o kompasu, kde bylo využito stejné funkce. Rozdíl je pouze v jednotkách, kdy zde pro rotaci kružnice (objekt `finalCircle`) jsou vyžadovány radiány (bez vynásobení 180/π) a směr rotace je opačný, proto není u návratové hodnoty funkce záporné znaménko.

Funkce `drawEnemyMarks` slouží k prvotnímu vykreslení značek nepřátel. Ve funkci je v cyklu pro každý objekt ze skupiny enemies volána funkce `createEnemyMark`. Zde je vytvořena samotná značka nepřítele. Značky nepřátel jsou přidány do skupiny `enemyMarks` a ta je následně předána objektu kružnice.

Pomocí funkce `refreshEnemyMarks` jsou aktualizovány polohy ukazatelů nepřátel. Funkce je volána ve funkci `animate` při obnově snímku. Ve funkci je nejprve pomocí metody `getWorldDirection` vypočítán vektor směru pohledu kamery a tento vektor je uložen do proměnné `vectorCamera`. Dále je v cyklu pro každý objekt ze skupiny `enemies` vypočítán vektor směřující od kamery k objektu. Tento vektor je uložen do proměnné `vectorEnemyToCamera`. Oba tyto vektory mají souřadnici Y nastavenou na hodnotu 0. Tato hodnota je pro výpočet nepodstatná, jelikož je počítán úhel mezi vektory v rovině osy X a Z. Funkcí `angleTo` je vypočítán úhel mezi těmito dvěmi vektory. Pomocí tohoto samotného údaje by nebylo možné určit, zda objekt leží vlevo, či vpravo od kamery.

Jestli objekt leží vlevo, či vpravo od kamery je možné zjistit vektorovým součinem. Ten určí vektor kolmý na oba vektory (v tomto případě bude výsledný vektor rovnoběžný s osou Y). Podle toho zda je souřadnice Y výsledného vektoru kladná nebo záporná lze rozhodnout, zda objekt leží vpravo, či vlevo od kamery.

Tento údaj je ve funkci uložen do proměnné `direction` a v podmínce je rozhodnuto, podle této proměnné, zda bude hodnota úhlu kladná nebo záporná. V dalších podmínkách je následně kontrolováno, zda absolutní hodnota úhlu nepřesáhla 60 stupňů. Pokud hodnota přesáhne 60 stupňů, je hodnota úhlu změněna na 60 případně -60 pro opačný směr. Tato korekce je nutná, aby byl ukazatel nepřátel vždy viditelný i pokud nepřítel opustí zorné pole kamery. Výsledná poloha značky nepřítele na kružnici je vypočítána odečtením vypočítaného úhlu mezi vektory od úhlu směru pohledu kamery. Tato hodnota je uložena do proměnné `finalAngleOnCircle`. Nakonec je pomocí tohoto finálního úhlu vypočítána poloha bodu na kružnici a do této polohy je přemístěna značka nepřítele. Nalezení konkrétní značky nepřítele je provedeno v cyklu dle vlastnosti `name`.



***Obrázek 19**: Nákres pro výpočet úhlu pozice objektu nepřítele (zdroj vlastní)*

```
// funkce pro přepočet pozice ukazatelů nepřátel
function refreshEnemyMarks(){
    //vektor ve směru pohledu kamery
    var vectorCamera = new THREE.Vector3();
    camera.getWorldDirection(vectorCamera);
    //převedení do 2D
    vectorCamera.y = 0;

    if(enemies.isGroup){
        for(var enemy of enemies.children){
            //vektor  ve směru od kamery k objektu
            vectorEnemyToCamera = new THREE.Vector3(enemy.position.x -
camera.position.x, 0, enemy.position.z - camera.position.z);
```

```
            //úhel mezi vektory pohledu kamery a vektorem mezi kamerou a
objektem(nepřítelem)
            var uhel = vectorCamera.angleTo(vectorEnemyToCamera);

            //pomocí vektorového součinu (kolmého vektoru) zjistíme "směr
uhlu" (zda je objekt vpravo nebo vlevo)

            //počítáme pouze Y souřadnici výsledného vektoru, jelikož oba
vektory mají Y = 0 => X a Z souřadnice výsledného vektoru bude nulová
            var direction = vectorCamera.x * vectorEnemyToCamera.z -
vectorCamera.z * vectorEnemyToCamera.x;
            if(direction < 0){
                uhel = -uhel;
            }

            //nepřítel je mimo zorné pole, ale jeho ukazatel zůstane viditelny
(na kraji lišty)
            if(uhel / (Math.PI/180) > 60){
                uhel = 60 * (Math.PI/180);
            }
            if(uhel / (Math.PI/180) < -60){
                uhel = -60 * (Math.PI/180);
            }
            var finalAngleOnCircle = spherical.theta - uhel;
            //výpočet polohy bodu na kružnici
            var a = Math.sin(finalAngleOnCircle)*circleRadius;
            var b = Math.cos(finalAngleOnCircle)*circleRadius;
            //nastavení nové pozice nepříteli dle jména
            for(var mark of enemyMarks.children){
                if(mark.name == enemy.name){
                    //nastavení nové pozice ukazatele
                    mark.position.x = a;
                    mark.position.y = b;
                    //nastavení ukazatelů ke kameře
                    mark.lookAt(camera.position);
                }
            }
        }
    }
}
```

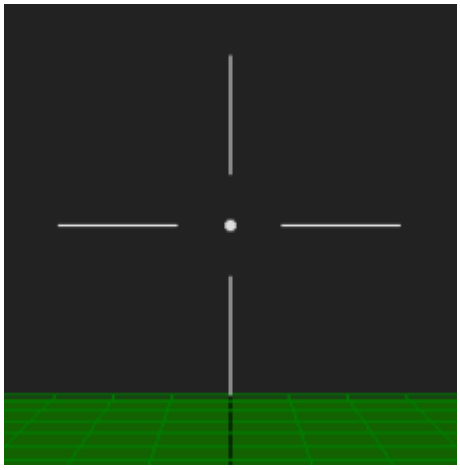
***Zdrojový kód** 7: Funkce pro překreslení pozic ukazatelů nepřátel do aktuální pozice (zdroj vlastní)*

4.4.3 Dynamický zaměřovač

Vzhled a chování:

Zaměřovač je dynamický, při pohybu po scéně dochází k rozestupu nitkových částí a mizení zaměřovací tečky uprostřed pro navození pocitu rozmíření. Nitková část konstantním pohybem

putuje do svého maximálního rozměření, kde setrvá do doby, než uživatel zastaví pohyb kamery po scéně. Při zastavení pohybu kamery po scéně se opět nitková část zaměřovače vrací zpět ke středu a objeví se i středová tečka. Zaměřovač je umístěn do středu obrazovky.



***Obrázek 20**: HUD prvek zobrazující nitkový kříž (zdroj vlastní)*

Popis implementace:

Pro implementaci zaměřovače byl zvolen přístup s použitím kamery ve scéně v kombinaci s geometrickými tvary. Nejprve byly vytvořeny 4 úsečky (`left`, `right`, `upper` a `lower`) pro nitkovou část a 1 kruh pro středovou tečku (`dot`). Tyto části byly následně umístěny před kameru a předány objektu kamery.

Při pohybu uživatele po scéně jsou jeho stisknuté klávesy pro pohyb (W, A, S, D) zaznamenávány do pole pomocí listeneru. Následně při každém renderování snímku je volána funkce `move`, ve které dochází k vyhodnocení, zda dojde k rozměření nebo zaměření. Pokud byla stisknuta pohybová klávesa, dojde k zavolání funkce `unaim`. Pokud nedošlo ke stisknutí pohybové klávesy, je volána funkce `aim`.

Ve funkci `unaim` je nastaveno středovému kruhu zaměřovače viditelnost (vlastnost `visible`) na hodnotu `false`. Pro určení maximální míry rozměření zaměřovače slouží konstanta `maxUnaim` a pro rychlost rozměření zaměřovače slouží konstanta `unaimCoeficient`. Hodnota těchto konstant byla zvolena tak, aby rozměření zaměřovače mělo přirozený pohyb. Ve funkci je dále podmínka, zda rozměření nedosáhlo maxima. Pokud nedosáhlo, dojde k posunu úseček od středu o hodnotu danou konstantou `unaimCoeficient`.

Funkce `aim` je velmi podobná funkci `unaim`, liší se pouze ve směru posunu úseček nitkového kříže, úsečky se pohybují ve směru do středu kříže a středový kruh je opět zobrazen (nastavení

vlastnosti `visible` na `true`). Pro určení rychlosti zaměření je využita konstanta `aimCoeficient`.

Pro realizaci statického zaměřovače (bez rozmíření) by způsob implementace pomocí HTML elementů byl jistě rychlejší a jednodušší. Stačilo by pouze umístit obrázek nitkového kříže do středu okna.

```
//míra rozmíření
const maxUnaim = 20 * scale;
const unaimCoeficient = 2 * scale;

//rozmíření při pohybu
function unaim(){

    //získaní úseček
    var leftLine = scene.getObjectByName( "left" );
    var rightLine = scene.getObjectByName( "right" );
    var upperLine = scene.getObjectByName( "upper" );
    var bottomLine = scene.getObjectByName( "lower" );
    var dot = scene.getObjectByName( "dot" );

    dot.visible = false;

    //pohyb úseček kříže ve směru od středu
    if(Math.abs(leftLine.position.x) < maxUnaim){
        leftLine.position.x -= unaimCoeficient;
        rightLine.position.x += unaimCoeficient;
        upperLine.position.y += unaimCoeficient;
        bottomLine.position.y -= unaimCoeficient;
    }
}
```

***Zdrojový kód 8**: Ukázka zdrojového kódu funkce unaim (zdroj vlastní)*

ZÁVĚR

Vytváření HUD prvků v knihovně Three.js je poměrně snadné. Pro jejich realizaci je ale nutné pochopit základní principy samotné knihovny. Pochopení těchto principů není až tak složité, a to hlavně díky přehledné dokumentaci knihovny. Je nutná také alespoň základní znalost jazyků HTML, CSS a JavaScript.

V teoretické části bylo dosaženo cílů nastíněním problematiky tvorby HUD prvků, zde jsou popsány především techniky grafického zpracování, které se pro HUD prvky využívají v herním prostředí. Další cíl je splněn vysvětlením základních principů knihovny tak, aby čtenář byl schopen vytvořit základní aplikaci pomocí této knihovny.

Byly zjištěny dva způsoby realizace HUD prvků. První možností je využití HTML elementů a vytvářet tak HUD prvky před samotnou scénou Three.js. Druhou možností je využití objektu kamery a vykreslovat HUD prvky před tímto objektem.

V praktické části bylo dosaženo stanovených cílů. Jsou vytvořeny ukázky nejčastěji používaných HUD prvků v herním prostředí, které jsou zasazeny do prostředí CodePen. Ukázky jsou prezentovány v přehledné webové prezentaci s popisem vzhledu, funkcionalit, implementace a možností zobrazení ukázek v CodePen. Mohou tak být užitečné pro vyvojaře her v této knihovně.

Výhodou knihovny Three.js je jistě poměrně velká členská základna a je tak možné najít spoustu užitečných rad na diskuzních fórech, jako je například three.js forum nebo Stack Overflow. Dále je možné čerpat informace z již vytvořených projektů umístěných v prostředích CodePen nebo JSFidle. Další výhodu knihovny Three.js lze spatřovat také v tom, že aplikace je spuštěna pomocí webového prohlížeče a uživateli dané aplikace odpadá nutnost jakékoliv instalace.

POUŽITÁ LITERATURA

[1] Installation. *three.js docs* [online]. © 2010-2022. [cit. 17.08.2022] https://threejs.org/docs/#manual/en/introduction/Installation.

[2] The evolution of the Head-Up Display. *BAE Systems* [online]. © 2022. [cit. 17.08.2022] https://www.baesystems.com/en/feature/our-innovations-hud.

[3] OLIVÍK, Pavel. Průhledový displej: z letadel do aut. *AutoRevue.cz* [online]. 5. 11 2011. [cit. 17.08.2022] https://www.autorevue.cz/pruhledovy-displej-z-letadel-do-aut.

[4] Head-up displej: Pilotem stíhačky snadno a rychle. *Autíčkář* [online]. 29. 6 2017. [cit. 17.08.2022] https://www.autickar.cz/clanek/head-up-displej-pilotem-stihacky-snadno-a-rychle/.

[5] PLURALSIGHT. *Designing a HUD That Works for Your Game* [online]. 5. 3 2014. [cit. 17.08.2022] https://www.pluralsight.com/blog/film-games/designing-a-hud-that-works-for-your-game.

[6] HUD Aiming Crosshair - The Last of Us Part II. *Interface in Game* [online]. 19. 6 2020. [cit. 17.08.2022] https://interfaceingame.com/screenshots/the-last-of-us-part-ii-hud-aiming-crosshair/.

[7] LENK, Štěpán. Apex Legends (RECENZE) – Nový princ battle royale žánru. *Alza.cz* [online]. 18. 2 2019. [cit. 17.08.2022] https://www.alza.cz/apex-legends-recenze.

[8] MICROSOFT. HUD Components. *GameUX MasterGuide* [online]. © 2021. [cit. 17.08.2022] https://gameuxmasterguide.com/2019-05-09-HUDComponents/.

[9] BYCER, Josh. Implementing Context Sensitive Commands in UI Design. *Game Developer* [online]. 28. 4 2016. [cit. 17.08.2022] https://www.gamedeveloper.com/design/implementing-context-sensitive-commands-in-ui-design.

[10] STARGAME, Alex. Designing a menu in computer games. *Medium* [online]. 14. 11 2018. [cit. 17.08.2022] https://medium.com/@alexstargame/designing-a-menu-in-computer-games-c8fbaf7a3d9a.

[11] DIRKSEN, Jos. *Three.js Essentials* Birmingham: Packt Publishing Ltd., 2014. 182 s. ISBN 978-1-78398-086-4.

[12] Foundamentals. *three.js manual* [online]. © 2010-2022. [cit. 17.08.2022] https://threejs.org/manual/#en/fundamentals.

[13] DIRKSEN, Jos. *Three.js Cookbook.* Birmingham: Publishing Ltd., 2015. 283 s.ISBN 978-1-78398-118-2.

[14] Setup. *three.js manual* [online]. © 2010-2022. [cit. 17.08.2022] https://threejs.org/manual/#en/setup.

[15] How to Run a Basic Web Server With Google Chrome. *Simple Help* [online]. 9. 7 2021. [cit. 17.08.2022] https://www.simplehelp.net/2021/07/09/how-to-run-a-basic-web-server-with-google-chrome/.

[16] EYGI, Cem. VS Code Live Server – How to Auto-Refresh Your Browser with this Simple Extension. *freeCodeCamp* [online]. 14. 10 2020. [cit. 17.08.2022] https://www.freecodecamp.org/news/vscode-live-server-auto-refresh-browser/.

[17] Creating a scene. *three.js docs* [online]. © 2010-2022. [cit. 17.08.2022] https://threejs.org/docs/#manual/en/introduction/Creating-a-scene.

[18] DIRKSEN, Jos. *Learning Three.js: The JavaScript 3D Library for WebGL.* Birmingham: Packt Publishing Ltd., 2013. 383 s. ISBN 978-1-78216-628-3.

[19] SERGIUSZ a DORA. Introduction to 3D: Three.js basics. *IntexSoft* [online]. 9. 4 2020. [cit. 17.08.2022] https://intexsoft.com/blog/introduction-to-3d-three-js-basics/.

[20] LEWY, Blue. Transformations, Coordinate Systems, and the Scene Graph. *Discover three.js* [online] 2020. [cit. 17.08.2022] https://discoverthreejs.com/book/first-steps/transformations/.

[21] Color. *three.js docs* [online]. © 2010-2022. [cit. 17.08.2022] https://threejs.org/docs/?q=CanvasRenderer#api/en/math/Color.

[22] Materials. *three.js manual* [online]. © 2010-2022. [cit. 17.08.2022] https://threejs.org/manual/#en/materials.

[23] Lights. *three.js manual* [online]. © 2010-2022. [cit. 17.08.2022] https://threejs.org/manual/#en/lights.

[24] LEWY, Blue. The Document Object Model and DOM API. *Discover three.js* [online]. 2020. [cit. 17.08.2022] https://discoverthreejs.com/book/appendix/dom-api-reference/.

[25] W3.CSS Templates. *W3Schools* [online]. © 1999-2022. [cit. 17.08.2022] https://www.w3schools.com/w3css/w3css_templates.asp.

[26] WILKINS, Jessica. How to Use CodePen – A Beginner's Guide. *freeCodeCamp* [online]. 1. 11 2021. [cit. 17.08.2022] https://www.freecodecamp.org/news/how-to-use-codepen/.

PŘÍLOHY

PŘÍLOHA A – CD

Tato příloha obsahuje webovou prezentaci a zdrojové kódy ukázek HUD prvků v jazycích HTML, CSS a JavaScript. Adresářová struktura je následující:

prezentace – Obsahuje HTML stránky s webovou prezentací a složky s obrazovou přílohou.

textova – Obsahuje textovou část práce ve formátu DOCX a PDF.

ukazky – Obsahuje podsložky s vytvořenými ukázkami, které jsou pojmenovány ve formátu cisloUkazky_nazevUkazky. Tyto složky obsahují soubory main.js, index.html a případné obrazové přílohy.