ChatGPT

Vývoj A-Frame projektu s Vite: Návod krok za krokem

Úvod

A-Frame je webový framework vyvinutý nad knihovnou Three.js, který usnadňuje tvorbu 3D a VR aplikací pomocí jednoduchého HTML zápisu. V kombinaci s moderním nástrojem **Vite** můžeme rychle vytvořit a spustit interaktivní 3D scénu přímo v prohlížeči. V tomto návodu si ukážeme, jak vytvořit projekt s A-Frame a Vite, a zaměříme se na následující oblasti: nastavení projektu, kamera z pohledu první osoby (POV kamera), načítání 3D modelů, tvorba vlastních komponent, raycasting (výběr objektů pomocí paprsku) a možnosti debugování.

Cílem je vybudovat základní VR/3D aplikaci krok za krokem. Po dokončení tohoto návodu budete mít funkční prostředí s **POV kamerou**, budete umět do scény přidat vlastní modely, rozšířit funkcionalitu pomocí komponent, implementovat interakce pomocí raycastingu a využít nástroje pro ladění scény.

Nastavení projektu s Vite

Nejprve si připravíme projektovou strukturu pomocí nástroje Vite a nainstalujeme A-Frame: 1. **Inicializace Vite projektu:** Otevřete terminál, přejděte do adresáře, kam chcete projekt umístit, a spusťte inicializační skript pro Vite. Zvolíme šablonu pro jednoduchý JavaScript projekt:

```
npm create vite@latest my-aframe-project -- --template vanilla
```

Tento příkaz vytvoří nový adresář `my-aframe-project` s základní strukturou aplikace. 2. **Instalace závislostí:** Přejděte do složky projektu a nainstalujte potřebné balíčky:

```
cd my-aframe-project
npm install
npm install aframe
```

Tím přidáme A-Frame jako závislost projektu (z registru npm). 3. **Úprava vstupních souborů:** Otevřete soubor `index.html` ve složce projektu a upravte jej tak, aby obsahoval A-Frame scénu. Například přidejte do těla dokumentu značku `<a-scene>`:

```
<body>
  <a-scene>
    <!-- Zde může být obsah scény -->
    <a-sky color="#ECECEC"></a-sky>
```

```
    <a-box position="0 1 -3" rotation="0 45 0" color="#4CC3D9"></a-box>
  </a-scene>
  <script type="module" src="/src/main.js"></script>
</body>
```

V tomto příkladu jsme do scény vložili oblohu (`<a-sky>`) s neutrální barvou pozadí a jednoduchou krychli (`<a-box>`) pro ověření, že scéna funguje. 4. **Import A-Frame v JavaScriptu:** Aby se A-Frame korektně načetl, otevřete soubor `src/main.js` (nebo ekvivalentní hlavní skript) a importujte modul A-Frame:

```
import 'aframe';
```

Tento import zajistí, že se knihovna A-Frame zavede (jako globální objekt `AFRAME`). Nyní můžete používat značky A-Frame v HTML. 5. **Spuštění vývojového serveru:** V terminálu spusťte Vite vývojový server:

```
npm run dev
```

Otevřete v prohlížeči URL, kterou Vite vypíše (typicky `http://localhost:5173`). Měli byste vidět základní A-Frame scénu s oblohou a objektem.

> **Poznámka:** Některé funkce WebXR a přístupu ke kameře zařízení vyžadují, aby aplikace běžela na zabezpečeném původu (HTTPS), i během vývoje. Vite ve výchozím nastavení spouští server přes HTTP. Pokud plánujete testovat VR režim nebo např. pohyb zařízení, zvažte použití pluginu *vite-plugin-basic-ssl*, který zajistí generování lokálního SSL certifikátu pro vývojový server (díky tomu bude adresa `https://localhost` funkční).

Kamera z pohledu první osoby (POV kamera)

Pro interaktivní prohlížení scény z pohledu uživatele nastavíme kameru tak, aby simulovala perspektivu první osoby. A-Frame umožňuje do scény vložit speciální entitu kamery. Pokud žádnou kameru nedefinujeme, vloží A-Frame automaticky výchozí kameru; my si však přidáme vlastní, abychom mohli nastavit ovládání.

Do HTML scény (uvnitř `<a-scene>`) přidejte entitu kamery s komponentami pro ovládání pohybu a pohledu:

```
<a-entity id="cameraRig" position="0 0 0">
  <a-entity
    camera
    wasd-controls
    look-controls
    position="0 1.6 0">
  </a-entity>
</a-entity>
```

Zde: - Vnější `<a-entity id="cameraRig">` slouží jako tzv. *rig* (podstavec) kamery. Můžeme s ním pohybovat jako celkem. - Vnořená entita s komponentou `camera` představuje samotnou kameru v oční výšce (~1.6 metru nad zemí). Komponenta `look-controls` zajišťuje možnost rozhlížet se (otáčet kamerou pomocí myši nebo gyroskopu zařízení) a komponenta `wasd-controls` povoluje pohyb pomocí klávesnice.

> **Tip:** Místo výše uvedené dvojice entit lze pro jednoduchost použít také zkrácenou formu `<a-camera>`. Například:
>
> ```
> <a-camera position="0 1.6 0" wasd-controls look-controls></a-camera>
> ```
>
> má podobný účinek – představuje kameru s nastaveným ovládáním. Uvnitř se automaticky spravuje entita kamery a její ovládání.

Jakmile kameru přidáte a obnovíte stránku, budete moci scénou volně procházet. **WASD ovládání** znamená, že stiskem kláves **W, A, S, D** (případně šipek) se budete pohybovat dopředu, doleva, dozadu a doprava. Pohyb myší (při aktivním okně scény) mění směr pohledu:

- **W / šipka nahoru:** pohyb postavy dopředu
- **S / šipka dolů:** pohyb dozadu
- **A / šipka doleva:** pohyb doleva (strafe)
- **D / šipka doprava:** pohyb doprava (strafe)

> **Poznámka:** Pokud kliknete do okna scény, A-Frame aktivuje tzv. *pointer lock* režim – skryje kurzor a umožní plynulé otáčení pohledu pomocí myši (podobně jako ve FPS hrách). Klávesou <kbd>Esc</kbd> lze myš opět uvolnit.

Výchozí rychlost chůze lze upravit parametrem `acceleration` u komponenty `wasd-controls` (v metrech za sekundu na druhou). Pokud se vám pohyb zdá příliš pomalý či rychlý, můžete například nastavit `wasd-controls="acceleration: 100"`. Také je možné povolit vertikální pohyb kamery: nastavením `wasd-controls="fly: true"` docílíte toho, že klávesy **Q** a **E** budou posouvat kameru dolů a nahoru (mód “létání”).

Načítání 3D modelů

Aby scéna nebyla tvořena jen primitivními tvary (krychle, koule apod.), načteme do ní vlastní 3D model. A-Frame podporuje formát glTF (ideálně jeho binární variantu *.glb*), který je pro webové 3D ideální.

Příprava modelu: Umístěte soubor modelu (např. `model.glb`) do projektu. Používáte-li Vite, je vhodné dát model do adresáře `public` (např. `public/models/model.glb`), aby byl při spuštění serveru volně dostupný na URL. Alternativně lze využít import modulů, ale přímé umístění do `public` je jednodušší.

Vložení modelu do scény: K načtení modelu slouží komponenta `gltf-model`. Do scény přidáme novou entitu s touto komponentou. Doporučuje se využít `<a-assets>` pro přednačtení modelu:

```
<a-scene>
  <a-assets>
    <a-asset-item id="myModel" src="models/model.glb"></a-asset-item>
  </a-assets>

  <a-entity gltf-model="#myModel" position="0 0 0" rotation="0 45 0" scale="1 1
1"></a-entity>
</a-scene>
```

V části `<a-assets>` definujeme prostředek (asset) typu `asset-item` s unikátním id (`myModel`) a cestou k souboru modelu. Díky tomu se model začne načítat, jakmile je scéna inicializována, a počká se s vykreslením scény, dokud se model nenačte. Následně vytvoříme `<a-entity>` s atributem `gltf-model="#myModel"`, čímž se do ní načte náš 3D model.

Nezapomeňte modelu nastavit **polohu** a případně **měřítko**: - `position` určuje souřadnice umístění modelu ve scéně (X, Y, Z). V našem příkladu (0, 0, 0) je model v počátku souřadnic (střed scény, na zemi). - `rotation` může být potřeba upravit, pokud model není natočen správně (údaje jsou ve stupních pro osy X, Y, Z). - `scale` pomůže přizpůsobit velikost modelu, pokud je příliš velký nebo malý.

> **Poznámka:** Pokud se model ve scéně nezobrazuje, zkontrolujte konzoli prohlížeče (panel **Network** a **Console**) – může jít o problém s cestou k souboru nebo nepodporovaným formátem. Ujistěte se, že cesta v atributu `src` nebo `gltf-model` směřuje na správný soubor. Pro glTF modely se také ujistěte, že všechny související soubory (textury, .bin soubory) jsou na správném místě.

Vlastní komponenty v A-Frame

Síla A-Frame spočívá v architektuře **Entity-Component-System**. Můžeme definovat vlastní komponenty, kterými rozšíříme chování entit ve scéně. Komponenta je v podstatě objekt obsahující logiku, který lze přiřadit jedné či více entitám.

Vlastní komponentu vytvoříme v JavaScriptu registrací přes `AFRAME.registerComponent`. Například vytvoříme komponentu, která bude nepřetržitě otáčet daným objektem (*spin* efektem):

```
AFRAME.registerComponent('spin', {
  schema: {
    speed: { type: 'number', default: 1 }
  },
  tick: function (time, timeDelta) {
    // Otočení entity kolem Y osy
    this.el.object3D.rotation.y += this.data.speed * (timeDelta / 1000);
  }
});
```

Tato komponenta s názvem `spin` má jednu vlastnost `speed` (rychlost otáčení, výchozí hodnota 1). Metoda `tick` se volá na každém snímku (zhruba 60× za sekundu) a postupně zvyšuje rotaci objektu kolem osy Y. Pomocí `timeDelta` (čas od posledního snímku v ms) zajišťujeme plynulost nezávisle na snímkové frekvenci.

Komponentu poté můžeme použít v HTML při definici entity – stačí uvést její název jako atribut:

```
<a-box position="0 0.5 -3" rotation="0 0 0" color="#4CC3D9" spin="speed: 2"></a-box>
```

Tímto se daná krychle bude otáčet rychlostí 2 (dvojnásobek výchozí rychlosti).

Komponenty mohou obsahovat více logiky a životních cyklů. Přehled hlavních částí, které lze v komponentě definovat: - **schema:** Objekt popisující vstupní vlastnosti komponenty (jejich názvy, datové typy a defaultní hodnoty). Tyto hodnoty jsou pak dostupné přes `this.data`. - **init():** Volá se při inicializaci komponenty na entitě (např. zde můžeme nastavit výchozí stav, přidat posluchače událostí apod.). - **update(oldData):** Volá se při každé změně dat (atributů) komponenty. `oldData` obsahuje předchozí hodnoty. Hodí se, pokud potřebujeme reagovat na změny parametrů. - **tick(time, timeDelta):** Volá se každé políčko (frame), pokud je entita zapnutá. Používá se pro průběžné aktualizace, animace nebo kontroly kolizí atd. (v našem příkladě zde provádíme rotaci). - **remove():** Volá se při odstranění komponenty z entity (např. při rušení entit). Můžeme sem dát úklid (odstranění event listenerů apod.). - **pause() a play():** Volá se při dočasném vypnutí/zapnutí entity či scény (např. po opuštění stránky, minimalizaci okna). Obvykle se nevyužívá tak často, ale je k dispozici.

Pomocí vlastních komponent můžeme scénu obohatit o libovolné chování. Dalším příkladem by mohla být komponenta, která reaguje na uživatelské akce, např. při kliknutí na objekt změní jeho barvu nebo vypíše zprávu do konzole. Takovou komponentu můžeme snadno vytvořit pomocí naslouchání na událost `click` v metodě `init()`:

```
AFRAME.registerComponent('log-on-click', {
  init: function() {
    this.el.addEventListener('click', (evt) => {
      console.log('Kliknuto na entitu:', this.el);
    });
  }
});
```

Kdybychom tuto komponentu přidali na entitu (např. `<a-box log-on-click></a-box>`), po kliknutí by se do konzole prohlížeče vypsala zpráva. Tím se dostáváme k problematice interakce a tzv. raycastingu.

Raycasting a interakce s objekty

Raycasting je technika, která se v A-Frame využívá pro detekci, na který objekt uživatel právě "míří" nebo kliká. Jednoduše řečeno, z kamery (nebo jiné entity) se vysílá neviditelný paprsek a zjišťuje se, který objekt ve scéně protíná.

V A-Frame je raycasting zprostředkován komponentou **cursor** (kurzor) společně s komponentou **raycaster**. Pro jednoduché interakce stačí do kamery přidat *kurzor*, který bude vysílat paprsek směrem, kam se díváme:

```
<a-entity camera wasd-controls look-controls position="0 1.6 0">
  <a-cursor></a-cursor>
</a-entity>
```

Vložením `<a-cursor>` jako potomka kamery získáme ve scéně virtuální kurzor. Ve výchozím stavu se jedná o malý bod (nebo kruh) uprostřed pohledu. Když tento bod namíří na nějaký objekt a uživatel stiskne tlačítko myši (nebo potvrzovací tlačítko v brýlích), A-Frame vyhodnotí událost **click** na daném objektu.

Abychom omezili, na které objekty kurzor reaguje, můžeme použít atribut `objects` komponenty raycaster. Například nastavíme-li `objects: .clickable`, bude paprsek "vidět" jen entity s CSS třídou `clickable`. Doporučujeme tedy označit interaktivní prvky ve scéně touto třídou:

```
<a-box class="clickable" position="0 0 -3" color="#4CC3D9"></a-box>
...
<a-entity camera look-controls wasd-controls position="0 1.6 0">
  <a-cursor objects=".clickable"></a-cursor>
</a-entity>
```

V tomto příkladu bude kurzor (raycaster) ignorovat vše kromě entit, které mají `class="clickable"`. Pokud namíříte středem obrazovky na danou krychli a kliknete, krychle obdrží událost `click`. Díky tomu může fungovat i jednoduchý HTML kód nebo komponenta reagující na kliknutí (např. naše `log-on-click` komponenta zmíněná výše, nebo použití standardního posluchače události přes JavaScript).

> **Tip:** Komponenta `cursor` generuje také události jako `mouseenter` a `mouseleave` na cílových entitách. To znamená, že můžete detekovat, kdy uživatel "najel" pohledem na objekt a kdy z něj pohled odvrátil (podobně jako hover efekt myši). Těchto událostí lze využít k např. zvýraznění objektu pod kurzorem apod.

A-Frame umožňuje mít i pokročilejší nastavení raycastů, například ručně vypouštět paprsky nebo zpracovávat detailní data o průsečíku (vzdálenost, souřadnice zásahu apod.), ale pro většinu interakcí si vystačíte s výše popsaným kurzorem a událostmi.

Debugging a nástroje pro ladění

Při vývoji 3D aplikací se neobejdeme bez ladicích nástrojů. A-Frame nabízí několik užitečných funkcí, které nám pomohou analyzovat a vylepšovat scénu:

- **A-Frame Inspector (vizuální inspektor):** Vestavěný nástroj, který zobrazíte stisknutím kombinace kláves <kbd>Ctrl</kbd> + <kbd>Alt</kbd> + <kbd>I</kbd> (na většině klávesnic; na Macu může být potřeba Fn+Ctrl+Alt+I). Otevře se přehledné rozhraní s 3D náhledem scény a panelem s hierarchií entit. Inspector umožňuje:
- Volně se rozhlížet a pohybovat v scéně nezávisle na herní kameře (inspektor má vlastní kameru pro náhled).
- Klikat na objekty a zobrazit či měnit jejich komponenty a vlastnosti v reálném čase. Můžete tak interaktivně ladit pozice, rotace, barvy, atd.
- V konzoli inspektoru vidíte případné chyby nebo varování (shodné s konzolí prohlížeče).
- Po dokončení můžete inspektor zavřít a vrátit se do aplikace; všechny provedené změny se dají exportovat jako HTML pro případné uložení (v inspektoru je funkce **Export to HTML**).
- **Konzole prohlížeče:** Klasický nástroj vývojáře v prohlížeči (Chrome DevTools, Firefox Developer Edition apod.). Zde se vypisují logy z `console.log` (například z našich komponent), chyby v kódu či upozornění A-Frame. V panelu **Network** pak uvidíte, zda se správně načetly modely a assety.
- **Statistiky výkonu:** A-Frame má vestavěný ukazatel FPS a dalších statistik. Aktivujete jej snadno přidáním atributu `stats` na `<a-scene>`:

  ```
  <a-scene stats>
  ```

  V rohu aplikace se poté zobrazí overlay s aktuální snímkovou frekvencí, počtem vykreslených entit, trojúhelníků atd. To je užitečné pro sledování optimalizací.
- **Další debug komponenty:** A-Frame nabízí např. komponentu `inspector` (pro případ, že byste chtěli inspektor otevřít programově), `stats-in-vr` (zobrazení statistik ve VR režimu) a další. Pro pokročilé ladění lze také dočasně používat nekomprimovanou verzi A-Frame knihovny, aby byly chybové hlášky čitelnější.

  Tip: Vite jako vývojový server podporuje *Hot Module Replacement*, takže můžete upravovat komponenty a kód "za běhu". Při editaci JavaScriptu Vite automaticky zrefrešuje modul a A-Frame může obnovit scénu (pozn.: komplexnější stavy aplikace je někdy lepší po změně plně obnovit stránku).

Závěr

Tento návod pokrývá základy vytvoření interaktivního 3D projektu s pomocí A-Frame a Vite. Probrali jsme nastavení prostředí, ovládání kamery z pohledu uživatele, načítání vlastních 3D modelů, tvorbu komponent rozšiřujících funkcionalitu, použití raycastingu pro interakce a nástroje pro ladění aplikace.

Dále doporučujeme prozkoumat oficiální dokumentaci A-Frame (např. sekce *Writing a Component*, *Interactions & Controllers* apod.), zejména pokud plánujete rozšířit aplikaci např. o podporu VR ovladačů či pokročilejší uživatelské rozhraní. S tímto základem můžete začít tvořit vlastní virtuální světy – přejeme hodně štěstí při vývoji!