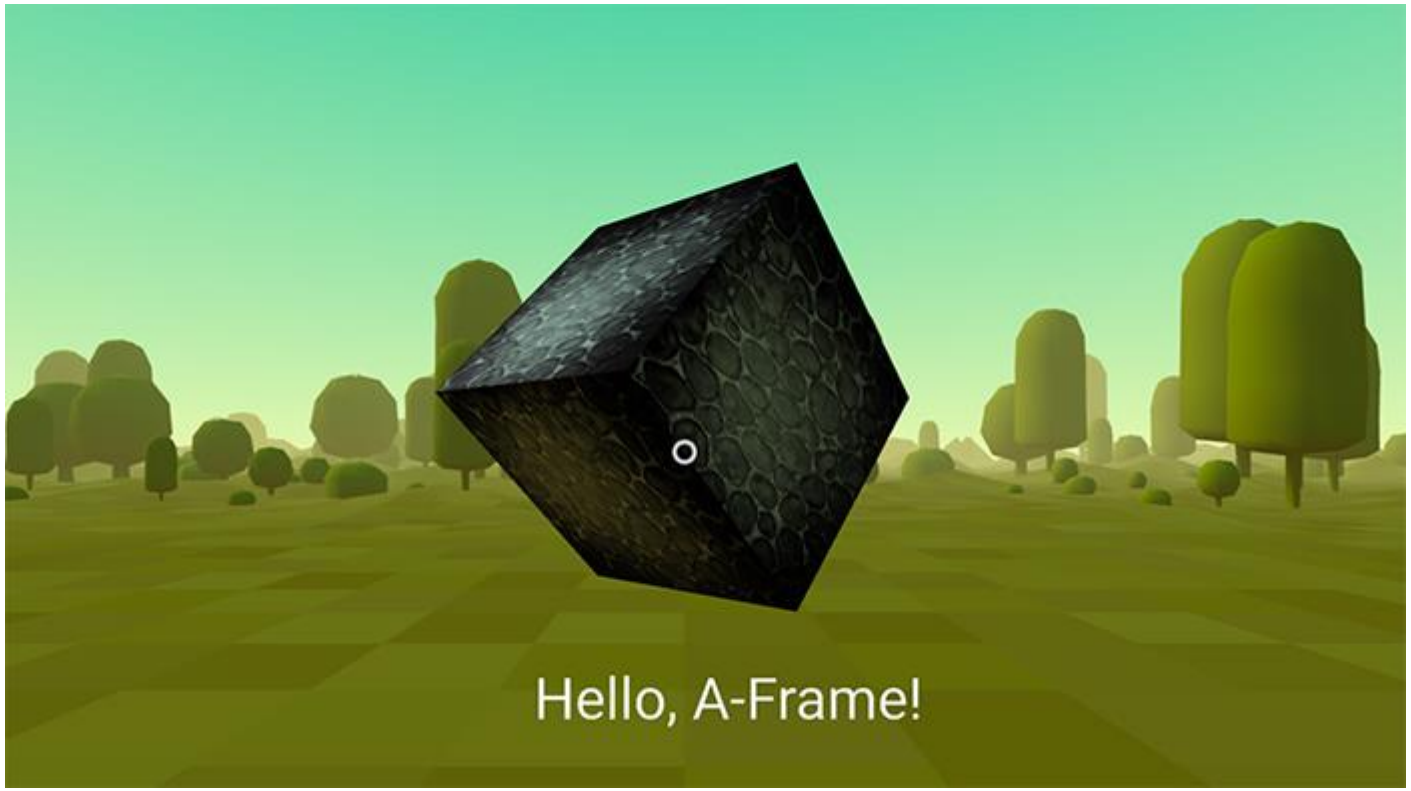
Фреймворк A-Frame



A-Frame не написано з 0 на чистому WebGL, в його основі лежить бібліотека *Three.js*.

A-Frame працює з HTML

Багато базових елементів A-Frame, таких як scene, camera, box, sphere та ін, додаються на сцену через однойменні теги з префіксом **a-** . Кожен подібний елемент зареєстрований як власний. На сьогоднішній день A-Frame (0.8.0) використовує специфікацію v0.

```
<a-scene>
<a-box position="-1 0.5 -3" rotation="0 45 0" color="#4CC3D9"></a-box>
<a-sphere position="0 1.25 -5" radius="1.25" color="#EF2D5E"></a-sphere>
</a-scene>
```

Крім згаданих вище двох елементів, існує ще ряд інших примітивів, які можуть бути додані на сцену таким же чином: *<a-circle>, <a-cone>, <a-cylinder>, <a-dodecahedron>, <a-icosahedron> , <a-octahedron>, <a-plane>, <a-ring>, <a-tetrahedron>, <a-torus-knot>, <a-torus>, <a-triangle>* .

•Також в A-Frame існує низка інших елементів, які виконують певні функції:
•

<a-camera> — створює камеру. На даний момент підтримується лише перспективна камера (PerspectiveCamera)
•**<a-obj-model>, <a-collada-model>, <a-gltf-model>** — всі вони вантажать та відображають моделі відповідного формату.
•**<a-cursor>** — елемент, який дозволяє виконувати різні дії: клік, наведення та ін. Курсор прив'язаний до центру камери, таким чином він завжди буде по центру того, що бачить користувач.
•**<a-image>** — відображення вибраного зображення на площині (<a-plane>).
•**<a-link>** - те ж саме тег, тільки для 3D сцени.
•**<a-sky>** — величезний циліндр навколо сцени, що дозволяє відображати 360 фотографій. Або його просто можна залити якимось кольором.
•**<a-sound>** - створює джерело звуку в заданій позиції.
•**<a-text>** - малює плоский текст.
•**<a-video>** - програє відео на площині.

A-Frame використовує ECS

ECS (Entity Component System) - патерн проектування програм та ігор. Широку поширеність набув якраз у другому сегменті. Як видно з назви, три основні поняття патерну – це Entity (Сутність), Component (Компонент), System (Система). У класичному вигляді вони взаємопов'язані один з одним таким чином: ми маємо певний об'єкт-контейнер (Сутність), до якого можна додавати компоненти. Зазвичай компонент відповідає за окрему частину логіки.

В A-Frame цей патерн реалізований за допомогою атрибутів.

```
<a-entity geometry="primitive: box; width: 1; height: 1; depth: 1"></a-entity>
```

geometry — у разі є компонентом, який було додано до сутності *<a-entity>* . Сам по собі *<a-entity>* не має будь-якої логіки (у глобальному сенсі), а компонент *geometry* — по суті перетворює його на куб чи щось інше. Іншим не менш важливим ніж *geometry* компонентом є **матеріал** .

Будь-який компонент в A-Frame має бути зареєстрований глобально через спеціальну конструкцію:

```
AFRAME.registerComponent('hello-world', {
init: function () {
console.log('Hello, World!');
}
});
```

Потім цей компонент можна буде додати на сцену або будь-який інший елемент.

```
<a-entity hello-world></a-entity>
```

У компонентах A-Frame є інші коллбеки життєвого циклу:

- **update** — викликається як із ініціалізації як init, і при оновленні будь-якої властивості даного компонента.
- **remove** - викликається після видалення компонента або сутності, що його містить. Тобто, якщо ви видалите <a-entity> з DOM, всі його компоненти викликають remove коллбек.
- **tick** - викликається щоразу перед рендерингом сцени. Всередині цикл рендерингу використовує requestAnimaitonFrame
- **tock** - викликається щоразу після рендерингу сцени.
- **play** – викликається кожен при відновленні рендерингу сцени. Насправді після scene.play();
- **pause** - викликається щоразу при зупинці рендерингу сцени. Насправді після scene.pause();
- **updateSchema** - викликається щоразу після оновлення схеми.

Ще одним важливим концептом компонента A-Frame є схема. Схема визначає властивості компонента. Вона визначається так:

```
AFRAME.registerComponent('my-component', {
schema: {
arrayProperty: {type: 'array', default: []},
integerProperty: {type: 'int', default: 5}
}
}
```

В даному випадку наш компонент **my-component** міститиме дві властивості *arrayProperty* і *integerProperty* . Щоб передати їх у компонент, потрібно задати значення відповідного атрибута.

```
<a-entity my-component="arrayProperty: 1,2,3; integerProperty: 7"></a-entity>
```

Отримати ці властивості всередині компонента можна через властивість **data**.

```
AFRAME.registerComponent('my-component', {
schema: {
arrayProperty: {type: 'array', default: []},
integerProperty: {type: 'int', default: 5}
},
init: function () {
console.log(this.data);
}
}
```

Щоб отримати властивості компонента з сутності до якої він доданий, можна скористатися трохи видозміненою функцією *getAttribute* . При зверненні до сутності A-Frame вона поверне не просто рядкове значення атрибуту, а об'єкт *data*

```
console.log(this.el.getAttribute('my-component'));
// {arrayProperty: [1,2,3], integerProperty: 7}
```

Приблизно так само можна змінити властивості компонента:

```
this.el.setAttribute('my-component',{arrayProperty: [], integerProperty: 5})
```

Системи A-Frame реєструється схожим чином як і компоненти:

```
AFRAME.registerSystem('my-system', {
schema: {},
init: function () {
console.log('Hello, System!');
},
});
```

Як і компонент система має схему і коллбеки. Тільки у системи їх всього 5: **init, play, pause, tick, tock** . Систему не потрібно додавати як компонент сутності. Вона буде автоматично додана до сцени.

```
this.el.sceneEl.systems['my-system'];
```

Якщо компонент матиме таке саме ім'я як і система, то система буде доступна за посиланням **this.system**.

```
AFRAME.registerSystem('enemy', {
schema: {},
init: function () {},
});

AFRAME.registerComponent('enemy', {
schema: {},
init: function () {
const enemySystem = this.system;
},
});
```

Комунікація в A-Frame відбувається за допомогою подій браузера

кожен елемент A-Frame пропатчений функцією **emit** , вона приймає три параметри: *перший* - назва події, *другий* - дані, яку потрібно передати, *третій* - чи повинна подія спливати.

```
this.el.emit('start-game', {level: 1}, false);
```

Підписатися на цю подію можна звичним усім нам способом, використовуючи **addEventListener:**

```
const someEntity = document.getElementById('someEntity');
someEntity.addEventListener('start-game', () => {...});
```

Створення базової сцени в A-Frame

- як додавати 3D об'єкти за допомогою примітивів;
- як трансформувати об'єкти у 3-х мірному просторі за допомогою, переміщень, поворотів та масштабування;
- як додати оточення;
- як додати текстури;
- як додати базову інтерактивність за допомогою подій та анімації;
- Як додати текст

Створення базової сцени в A-Frame

```
<html>
  <head>
    <script src="https://aframe.io/releases/0.8.0/aframe.min.js"></script>
  </head>
  <body>
    <a-scene>
    </a-scene>
  </body>
</html>
```

Додаємо об'єкти

Використовуючи <a-scene>, ми додаємо 3D об'єкти за допомогою стандартних примітивів A-Frame, наприклад <a-box>. Ми можемо використовувати <a-box> просто як звичайний HTML елемент, додаючи до нього атрибути кастомізації. Ось ще кілька стандартних примітивів доступних в A-Frame: <a-cylinder>, <a-plane>, <a-sphere>.

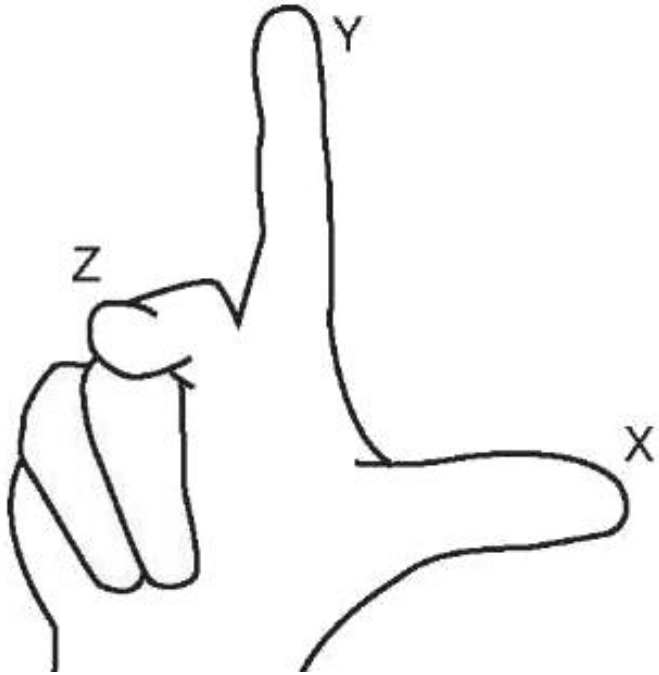
```
<a-entity id="box" geometry="primitive: box" material="color: red"></a-entity>
```

Створення базової сцени в A-Frame

Трансформація об'єкта в 3D

A-Frame вимірює дистанцію в метрах (а не в пікселях як у React360) так як WebVR API повертає дані позиції та пози в метрах. Коли ви проектуєте ВР сцену, дуже важливо розглядати саме реальні розміри об'єктів. Куб з висотою 10 метрів може виглядати нормально на екранах наших комп'ютерів, але він буде занадто масивним при зануренні у віртуальну реальність.

Одиниці обертання в A-Frame - це градуси (не радіани як Three.js), тому вони автоматично будуть перераховані в радіани при попаданні в Three.js. Для визначення позитивного напрямку обертання використовується правило правої руки. Направте палець правої руки вздовж позитивного напрямку будь-якої осі, тоді напрямок у якому наші пальці зігнуті та будуть позитивним напрямком обертання.



обертання та масштабування:

```
<a-scene>
<a-box color="red" rotation="0 45 45" scale="2 2 2"></a-box>
</a-scene>
```

Створення базової сцени в A-Frame

Трансформації предків та нащадків

Граф сцени в A-Frame реалізований за допомогою HTML. Кожен елемент такої структури може мати кілька нащадків і лише предка. Будь-який нащадок у такій структурі успадковує властивості трансформації (position, rotation, scale) свого предка.

Наприклад, ми можемо додати сферу як нащадка куба:

```
<a-scene>
<a-box position="0 2 0" rotation="0 45 45" scale="2 4 2">
<a-sphere position="1 0 3"></a-sphere>
</a-box>
</a-scene>
```

Позиція сфери на сцені буде *123* , а не *103* , оскільки за умовчанням сфера знаходиться в координатах свого предка, тобто куба - *020* . Ця точка у разі буде точкою відліку для сфери, що має координати *1 0 3* . Те саме стосується обертання і масштабу. Якщо будь-який із атрибутів куба буде змінено, ця зміна автоматично торкнеться всіх нащадків (у нашому випадку сферу).

Створення базової сцени в A-Frame

Розміщуємо наш куб попереду від камери

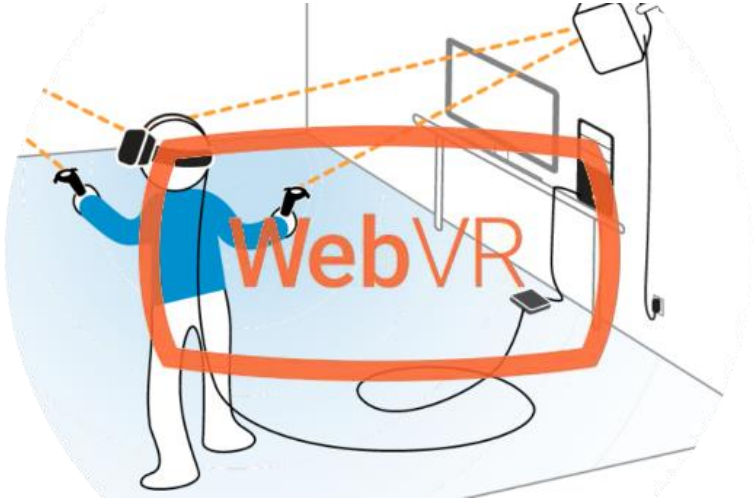
Тепер зробимо наш куб видимим для нашої камери. Ми можемо пересунути наш куб на 5 метрів по осі Z у негативному напрямку (тобто від нас). Давайте також піднімемо його на два метри вгору по осі Y. Все це можна зробити, змінивши атрибут position:

```
<a-scene>
<a-box color="red" position="0 2 -5" rotation="0 45 45" scale="2 2 2"></a-box>
</a-scene>
```

Створення базової сцени в A-Frame

Управління

Для плоских дисплеїв (ноутбуки, комп'ютери) керування камерою за замовчуванням здійснюється через перетягування мишкою та WASD або стрілки на клавіатурі. У телефонах відповідає акселерометр. Незважаючи на те, що A-Frame є фреймворком для WebVR, він підтримує дані схеми управління для того, щоб сцену можна було переглянути і без шолома віртуальної реальності.



Створення базової сцени в A-Frame

Додаємо оточення

A-Frame дозволяє розробникам створювати та ділитися компонентами, які потім легко використовувати у своїх проектах.

Компонент оточення це чудовий та простий спосіб створити візуальну платформу для наших ВР додатків. Він дозволяє створювати десятки оточень, що настроюються численною кількістю параметрів.

Для початку потрібно підключити скрипт до розділу після A-Frame:

```
<head>
<script src="https://aframe.io/releases/0.8.0/aframe.min.js"></script>
<script src="https://unpkg.com/aframe-environment-component/dist/aframe-environment-component.min.js"></script>
</head>
```

Після цього, вже в сцені, потрібно додати тег <a-entity> з атрибутом environment та заданими налаштуваннями, нехай це буде forest (ліс) з 200 деревами:

```
<a-scene>
<a-box color="red" position="0 2 -5" rotation="0 45 45" scale="2 2 2"></a-box>
<a-entity environment="preset: forest; dressingAmount: 200"></a-entity>
</a-scene>
```

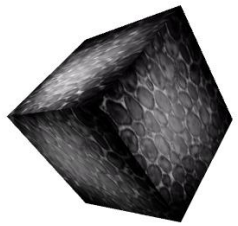
Створення базової сцени в A-Frame

Застосовуємо текстури

Ми можемо застосувати текстуру до нашого куба, використовуючи зображення, відео або canvas за допомогою атрибута src, також як ми робимо це зі звичайним тегом. Також нам необхідно видалити color=«red» оскільки колір все одно не відображатиметься під час використання текстури.

```
<a-scene>
<a-box src="https://i.imgur.com/mYmmbrp.jpg" position="0 2 -5" rotation="0 45 45"
scale="2 2 2"></a-box>
<a-sky color="#222"></a-sky>
</a-scene>
```

Тепер ми маємо побачити наш куб уже з текстурою, підвантаженою он-лайн.



Створення базової сцени в A-Frame

Використовуємо систему завантаження файлів

Якщо ми напишемо тег всередині <a-assets>, A-Frame не відображатиме його як у звичному HTML, він обробить джерело і відправить його до Three.js. Важливо відзначити, що завантаживши зображення один раз ми можемо використовувати його будь-де, наприклад як текстуру. Також A-Frame подбає про крос-доменності та інші можливі проблеми, пов'язані з передачею файлів по мережі.

Щоб використовувати систему керування завантаженням файлів для додавання текстури потрібно:

Додати тег **<a-assets>** всередину тега <a-scene> (на перший рівень)
•Додати тег **<img>** , атрибут src повинен містити посилання на файл зображення (так само як і в звичному нам HTML)
•Потрібно встановити атрибут **id** для тега img. В ідеалі він повинен описувати саму текстуру (наприклад, id=”boxTexture”)
•Додати атрибут **src** для об'єкта, який повинен її містити. Наприклад, для <a-box src="#boxTexture" >

```
<a-scene>
<a-assets>
<img id="boxTexture" src="https://i.imgur.com/mYmmbrp.jpg">
</a-assets>
<a-box src="#boxTexture" position="0 2 -5" rotation="0 45 45" scale="2 2 2"></a-box>
<a-sky color="#222"></a-sky>
</a-scene>
```

Створення базової сцени в A-Frame

Створюємо довільне оточення

Ми вже говорили про компонент оточення вище. Воно генерується автоматично. А якщо нам потрібно зробити своє власне оточення: додати хмари, землю, інші об'єкти? Пропоную поговорити про це.

Додаємо фонове зображення для сцени

Для додавання фонового зображення для сцени в A-Frame є спеціальний елемент <a-sky>. <a-sky> дозволяє додавати як чистий колір, так і 360 зображень або відео. Наприклад, щоб додати темно-сірий фон потрібно написати наступний код:

```
<a-scene>
<a-box color="red" position="0 2 -5" rotation="0 45 45" scale="2 2 2"></a-box>
<a-sky color="#222"></a-sky>
</a-scene>
```

Або ми можемо використовувати 360 фотографій:

```
<a-scene>
<a-assets>
<img id="boxTexture" src="https://i.imgur.com/mYmmbrp.jpg">
<img id="skyTexture"
src="https://cdn.aframe.io/360-image-gallery-boilerplate/img/sechelt.jpg">
</a-assets>
<a-box src="#boxTexture" position="0 2 -5" rotation="0 45 45" scale="2 2 2"></a-box>
<a-sky src="#skyTexture"></a-sky>
</a-scene>
```

Створення базової сцени в A-Frame

Додаємо землю

Щоб додати землю, ми будемо використовувати елемент <a-plane>. За замовчуванням площини A-Frame орієнтовані паралельно осі XY. Щоб зробити площину паралельної землі, нам потрібно повернути її так, щоб вона була паралельна осі XZ. Це можна зробити, повернувши площину на -90° по осі X.

```
<a-plane src="#groundTexture" rotation="-90 0 0" width="30" height="30"
repeat="10 10"></a-plane>
```

Працюємо над світлом

Щоб змінити освітлення нашої сцени, ми можемо додати або переналаштувати елемент <a-light> . За замовчуванням, у нас не було жодних джерел світла, але A-Frame сам додає розсіяне світло (ambient light) і направлене світло (directional light). Якби A-Frame не додав ці джерела світла, сцена була б повністю чорною. Якщо ми додамо власні джерела світла, то джерела, які додає A-Frame за замовчуванням, будуть видалені.

```
<a-scene>
<a-assets>
<img id="boxTexture" src="https://i.imgur.com/mYmmbrp.jpg">
<img id="skyTexture"
src="https://cdn.aframe.io/360-image-gallery-boilerplate/img/sechelt.jpg">
<img id="groundTexture" src="https://cdn.aframe.io/a-painter/images/floor.jpg">
</a-assets>
<a-box src="#boxTexture" position="0 2 -5" rotation="0 45 45" scale="2 2 2"></a-box>
<a-sky src="#skyTexture"></a-sky>
<a-light type="ambient" color="#445451"></a-light>
<a-light type="point" intensity="2" position="2 4 4"></a-light>
</a-scene>
```

Створення базової сцени в A-Frame

Додаємо анімацію

У нас є можливість додати анімацію, використовуючи вбудовану систему анімації A-Frame. Анімація змінює деяке значення (властивість компонента) з часом. Нам потрібно лише додати елемент <a-animation> як нащадок деякого об'єкта, наприклад <a-box>. Спробуємо анімувати куб таким чином, щоб він рухався вгору-вниз.

```
<a-scene>
<a-assets>
<img id="boxTexture" src="https://i.imgur.com/mYmmbrp.jpg">
</a-assets>
<a-box src="#boxTexture" position="0 2 -5" rotation="0 45 45" scale="2 2 2">
<a-animation attribute="position" to="0 2.2 -5" direction="alternate" dur="2000"
repeat="indefinite"></a-animation>
</a-box>
</a-scene>
```

Ми говоримо <a-animation>:

- Що потрібно анімувати атрибут position.
- Анімувати до *0 2.2 -5* що на 20 сантиметрів вище за першочергову позицію.
- Змінювати напрямок анімації на кожному циклі.
- Цикл анімації займатиме 2 секунди (2000 мілісекунд).
- Повторювати анімацію нескінченно.

Створення базової сцени в A-Frame

Додаткові деталі

<a-animation> використовує цикл рендерингу A-Frame, тому кожна зміна властивостей об'єкта відбувається один раз за кадр. Якщо вам потрібно більше контролю і ви хочете змінювати значення вручну, ви можете написати компонент A-Frame з колбеком tick та бібліотекою такою як Tween.js (яка, до речі, доступна за посиланням AFRAME.TWEEN). Для кращої продуктивності, покадрові операції повинні бути виконані на рівні A-Frame, не потрібно створити власну функцію *requestAnimationFrame* коли A-Frame **вже має її**.

Додаємо інтерактивність

Давайте додамо можливість взаємодіяти з нашим кубом: коли ми дивимося на нього, ми збільшимо його розміри, а при натисканні він прокрутиться навколо своєї осі.

Щоб прив'язати курсор до камери, нам потрібно додати його як нащадок до елемента камери (<a-camera>).

Оскільки ми ще не визначили камеру, A-Frame зробив це автоматично. але оскільки нам потрібно додати курсор до камери, ми визначимо <a-camera> вручну і додамо туди <a-cursor>:

Створення базової сцени в A-Frame