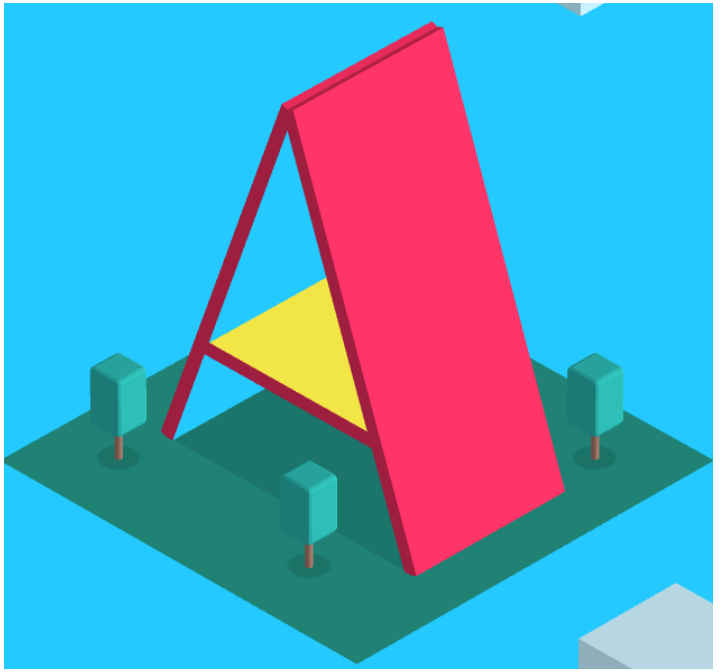


A-Frame & AR.js

Meetup CreativeCodeParis, 21 juin 2018

Grégory Jarrige - gregory.jrrg[at]gmail.com

Avertissement de l’auteur

- Tout est bon dans A-Frame
- Ce framework est tellement riche qu’un bouquin de 300 pages ne suffirait probablement pas à présenter toutes ses possibilités, aussi cette présentation est forcément partielle
- Tout est bon aussi dans AR.js, cette présentation n’est qu’une très (trop) courte à ce framework.

A-Frame, c’est quoi ?

- Un framework web pour construire des expériences de réalité virtuelle (en anglais « virtual reality » ou VR) dans un navigateur (d’où le terme de « WebVR »)
- A-Frame enrichit le vocabulaire HTML des navigateurs avec des balises HTML dédiées à la production d’univers 3D, balises qui peuvent être manipulées avec Javascript
- A-Frame est compatible avec de nombreux périphériques orientés VR tels que Vive, Rift, Daydream, GearVR, desktop (navigateur), etc.

Terminologie

- La réalité virtuelle (VR) immerge le spectateur dans un univers virtuel. Souvent mise en œuvre au travers de lunettes englobant complètement le champ de vision, la VR provoque chez certaines personnes une fatigue visuelle excessive, voire des maux de tête.
- La réalité augmentée (AR), c’est la possibilité d’intégrer dans l’univers réel (reconstitué sur écran) des artefacts numériques (pas de fatigue visuelle dans ce cas)

Sous le capot d’A-Frame

- Tween.js pour l’animation
  - https://github.com/tweenjs/tween.js
- Three.js pour la visualisation 3D (qui s’appuie sur l’API WebGL)
  - https://threejs.org
- + quelques composants complémentaires, dont certains proviennent de l’écosystème Node.js/NPM, et d’autre sont spécifiques à A-Frame

Liens

- https://aframe.io/
- https://aframe.io/docs/0.8.0
- https://aframe.io/docs/0.8.0/introduction/faq.html
- https://aframe.io/community/
- https://aframe.io/blog/
- https://aframe.io/examples/

Outils d’édition

- Pour travailler, n’importe quel éditeur HTML peut faire l’affaire :
  - https://www.sublimetext.com/
  - https://atom.io/
  - http://brackets.io/
  - https://code.visualstudio.com/
- Pas besoin de serveur web pour débuter, même si cela peut être utile sur des projets plus importants

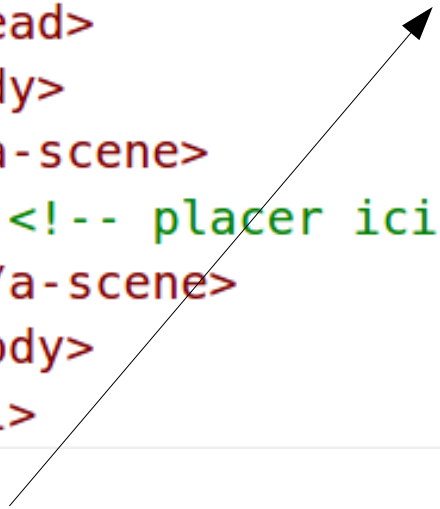
Archive de la prez

- Tous les exemples utilisés dans cette présentation, ainsi que le support de cette présentation, sont téléchargeables librement dans le dépôt Github suivant :
- TODO : lien à ajouter

Exemples

- Pour créer une scène VR avec A-Frame, on a juste besoin d'un squelette HTML :

```
1   <!DOCTYPE html>
2   <html>
3     <head>
4       <meta charset="utf-8">
5       <title>A-Frame Skeleton</title>
6       <script src="../js/aframe.js"></script>
7     </head>
8     <body>
9       <a-scene>
10        <!-- placer ici les primitives de A-Frame -->
11      </a-scene>
12    </body>
13  </html>
```

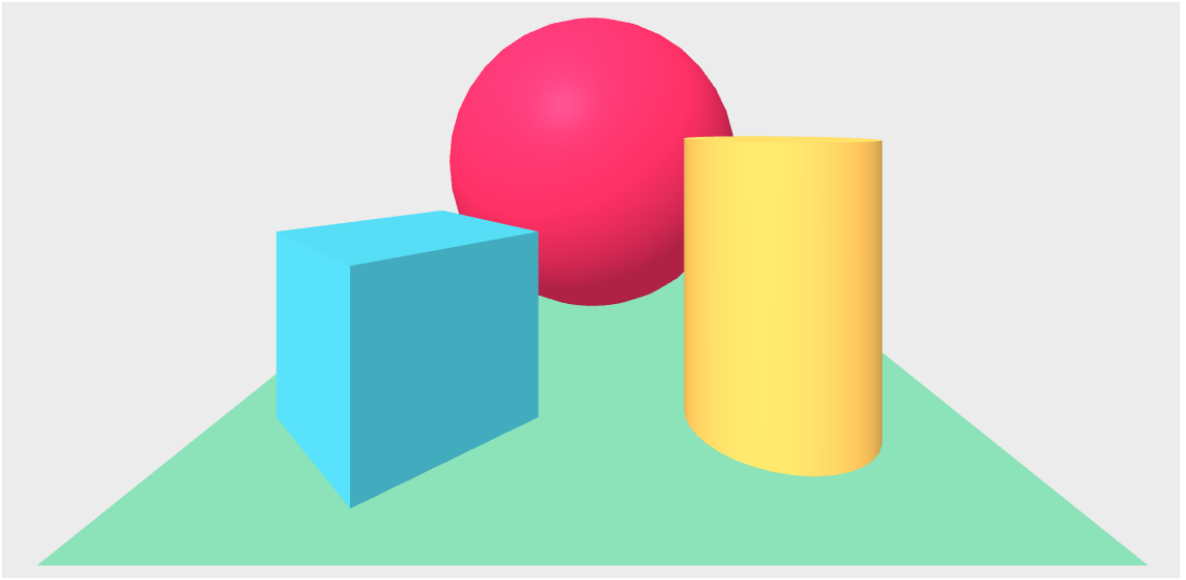


Chemin à adapter en fonction de l'endroit où vous avez installé A-Frame (localement)

Premiers pas

- Premier exemple proposé par le site officiel :

```
<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-cylinder position="1 0.75 -3" radius="0.5" height="1.5"
    color="#FFC65D"></a-cylinder>
  <a-plane position="0 0 -4" rotation="-90 0 0" width="4" height="4"
    color="#7BC8A4"></a-plane>
  <a-sky color="#ECECEC"></a-sky>
</a-scene>
```

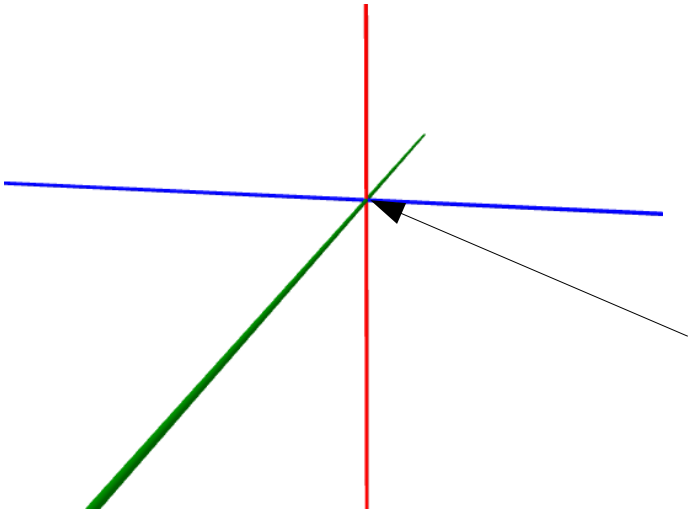


5 primitives

Premiers pas

- Pour se situer plus facilement dans l'espace 3D, je recommande de créer 3 axes temporaires avec la primitive « cylindre » :

```
<a-scene>
  <a-cylinder position="0 0 0" radius="0.02" height="10" color="red"
    ></a-cylinder>
  <a-cylinder position="0 0 0" radius="0.02" height="10" color="green"
    rotation="90 0 0" ></a-cylinder>
  <a-cylinder position="0 0 0" radius="0.02" height="10" color="blue"
    rotation="0 0 90" ></a-cylinder>
</a-scene>
```



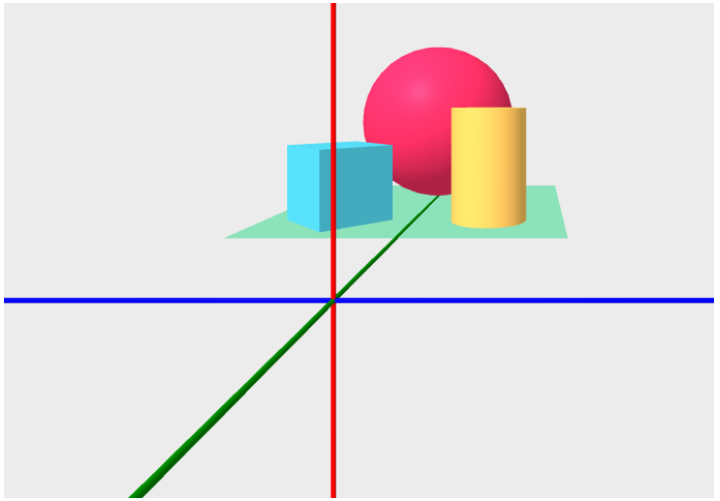
Point 0, 0, 0

Les cylindres sont verticaux par défaut, et centrés par défaut sur le point 0,0,0. On joue sur l'attribut « rotation » pour faire pivoter 2 cylindres sur 3.

Premiers pas

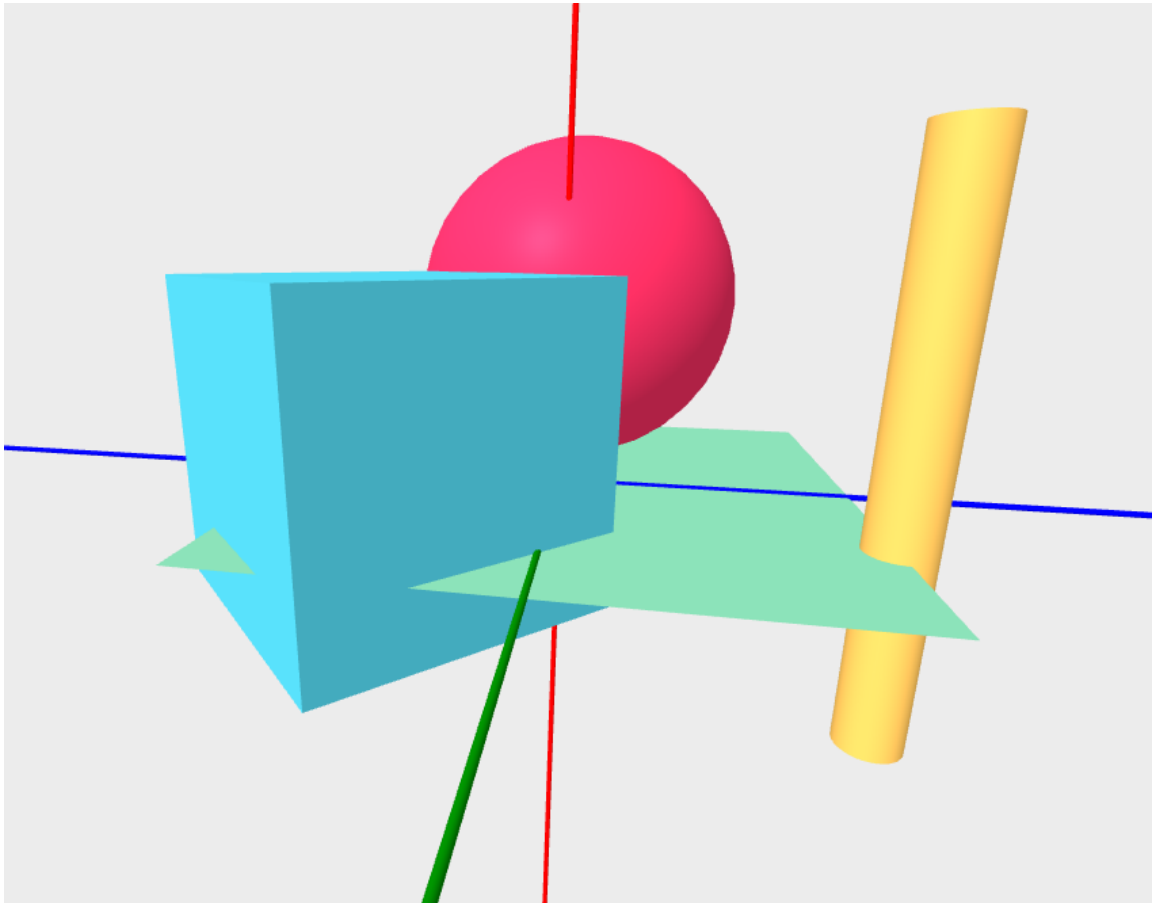
- Exemple initial avec ajout des axes X, Y et Z :

```
<a-scene>
  <a-cylinder position="0 0 0" radius="0.02" height="10" color="red"></a-cylinder>
  <a-cylinder position="0 0 0" radius="0.02" height="10" color="green"
    rotation="90 0 0" ></a-cylinder>
  <a-cylinder position="0 0 0" radius="0.02" height="10" color="blue"
    rotation="0 0 90" ></a-cylinder>
  <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-cylinder position="1 0.75 -3" radius="0.5" height="1.5" color="#FFC65D"
    ></a-cylinder>
  <a-plane position="0 0 -4" rotation="-90 0 0" width="4" height="4"
    color="#7BC8A4"></a-plane>
  <a-sky color="#ECECEC"></a-sky>
</a-scene>
```



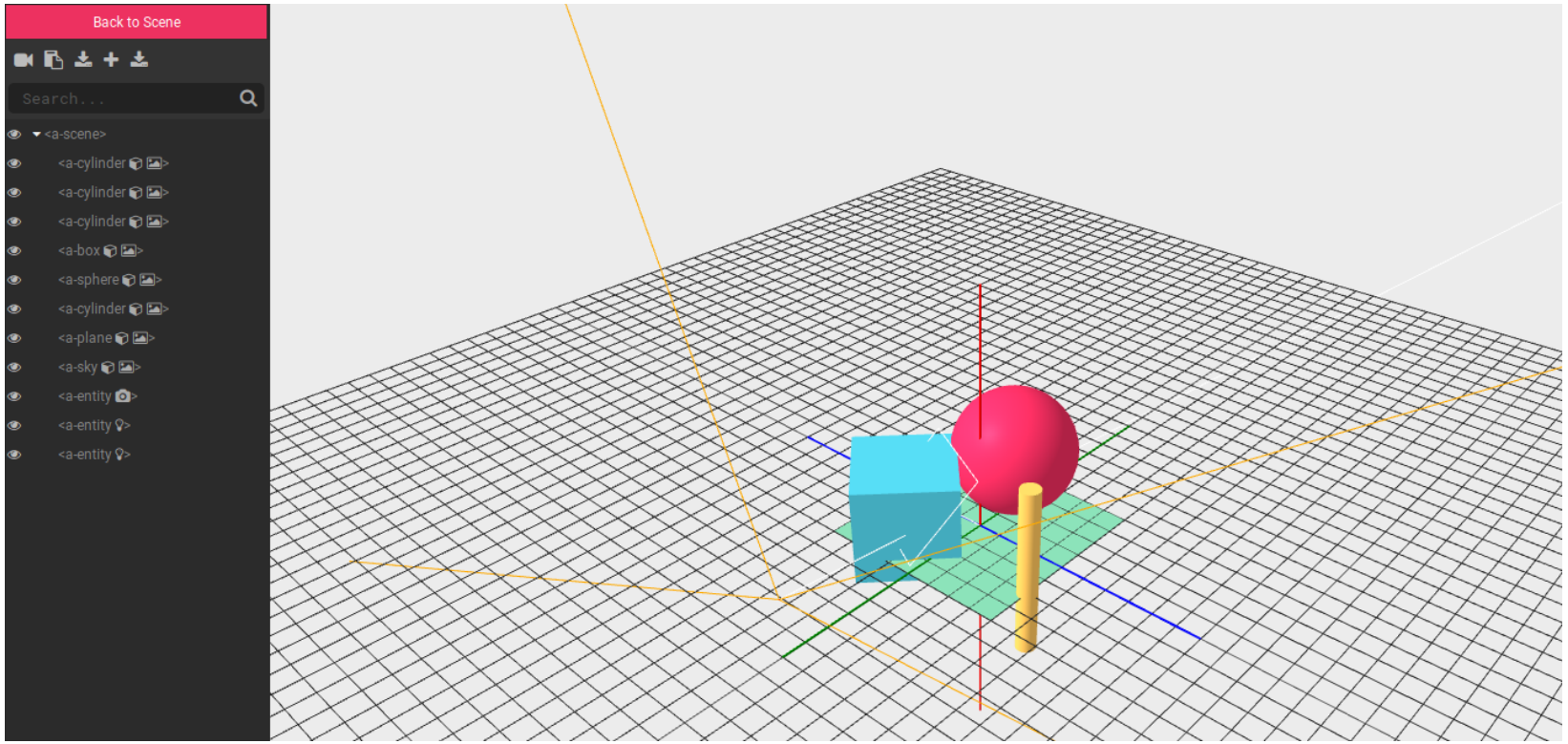
Premiers pas

- On peut « croiser » librement les primitives



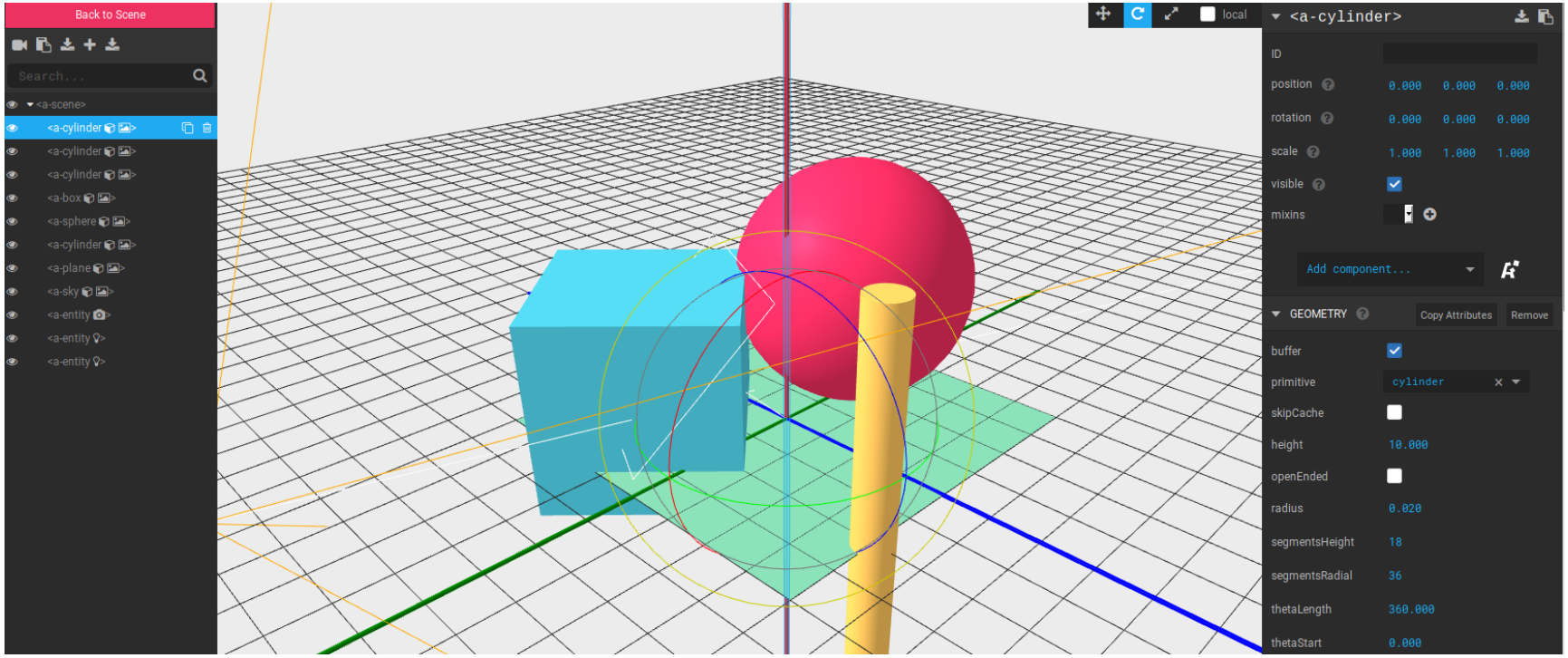
Editeur 3D

- La combinaison des touches Ctrl-Alt-i permet d’activer un éditeur 3D avancé :



Editeur 3D

- On peut effectuer certaines manipulations à la souris, et modifier les paramètres de chaque composant, et exporter le résultat final :



 → Fonction d'export

Placement de la caméra

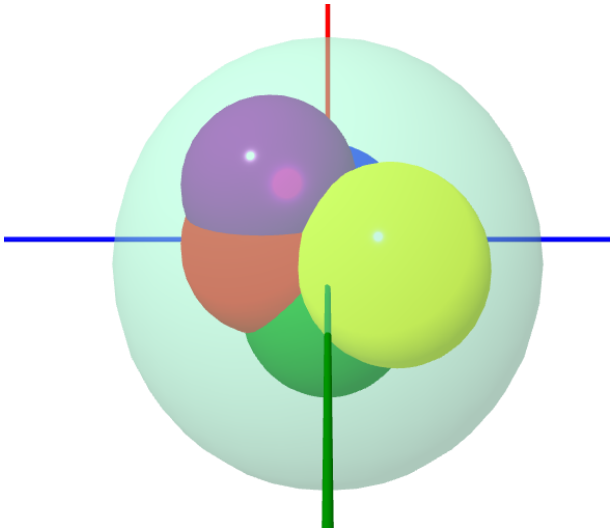
- Par défaut, la caméra se positionne sur le point 0,0,0, ce qui peut être embêtant
- Il est préférable de placer soi-même sa caméra en fonction des objets de la scène
- Essayez dans un premier temps ces paramètres, ensuite vous ajusterez selon vos besoins

```
<a-camera position="0 1.8 6"></a-camera>
```

Composants imbriqués

- On peut imbriquer des éléments entre eux pour former un nouveau composant, comme ici une sphère semi-transparente contenant d'autres sphères :

```
<a-sphere position="0 0 0"  radius="1.75" color="#7BC8A4"
  roughness="0.2" opacity=".5" >
  <a-sphere position="-.5 0 0" radius="0.75" color="red" roughness="0.2"></a-sphere>
  <a-sphere position="0 -.5 0" radius="0.75" color="green" roughness="0.2"></a-sphere>
  <a-sphere position="0 0 -.5" radius="0.75" color="blue" roughness="0.2"></a-sphere>
  <a-sphere position="-.5 .5 0" radius="0.75" color="purple" roughness="0.2"></a-sphere>
  <a-sphere position=".5 0 .5" radius="0.75" color="yellow" roughness="0.2"></a-sphere>
</a-sphere>
```



Composants imbriqués

- Il est possible d'appliquer les mêmes effets, notamment de rotation, à tous les enfants de la sphère « mère », en ajoutant une balise « animation » à cette dernière :

```
<a-sphere position="0 0 0"  radius="1.75" color="#7BC8A4"
   roughness="0.2" opacity=".5" >
  <a-animation attribute="rotation" from="0 0 0" to="0 360 360"
    repeat="indefinite" easing="linear"></a-animation>
  <a-sphere position="-.5 0 0"  radius="0.75" color="red" roughness="0.2"></a-sphere>
  <a-sphere position="0 -.5 0"  radius="0.75" color="green" roughness="0.2"></a-sphere>
  <a-sphere position="0 0 -.5"  radius="0.75" color="blue" roughness="0.2"></a-sphere>
  <a-sphere position="-.5 .5 0"  radius="0.75" color="purple" roughness="0.2"></a-sphere>
  <a-sphere position=".5 0 .5"  radius="0.75" color="yellow" roughness="0.2"></a-sphere>
</a-sphere>
```

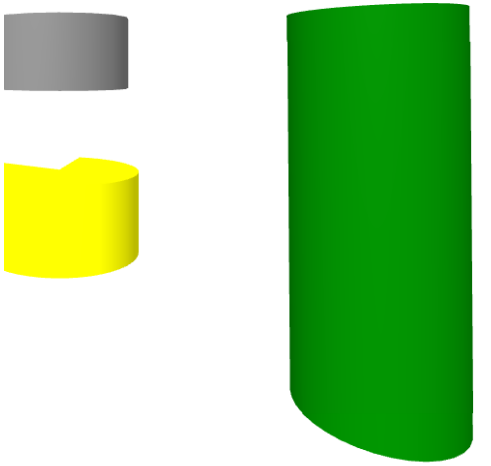
- On peut aussi placer des balises « animation » sur chacun des enfants, pour aboutir à des effets bizarres... ou sympas

Les effets

- On dispose d’effets multiples, combinables entre eux :
  - translation, rotation, apparition/disparition, changement d’échelle, changement de couleur et d’opacité, etc.
  - Chaque catégorie d’effets a ses paramètres spécifiques,
  - Pour passer d’un état à un autre, on dispose d’effets de transition :
    - linear, ease-in, ease-out, etc.
  - Lecture vivement recommandée :
    - https://aframe.io/docs/0.8.0/core/animations.html

Exemples d’effets

```
<a-scene>
    <a-camera position="0 1.8 4"></a-camera>
    <!-- Cylindre gris -->
    <a-cylinder position="-4 0 -3" rotation="90 -45 20" radius="1"
        segments-radial="16" color="grey">
        <a-animation attribute="rotation" from="0 0 0" to="0 180 180" repeat="3"
        easing="ease"></a-animation>
        <a-animation attribute="position" from="-2 0 0" to="0 2 -4" repeat="3"
        easing="ease-in"></a-animation>
    </a-cylinder>

    <!-- Pacman -->
    <a-cylinder position="0 0 -3" rotation="65 45 0" radius="1" height="1"
        theta-start="57" theta-length="286" side="double" color="yellow">
        <a-animation attribute="rotation" from="0 0 0" to="0 180 360" repeat="6"
            easing="linear"></a-animation>
    </a-cylinder>

    <!-- Cylindre pourpre -->
    <a-cylinder position="4 0 -3" rotation="-80 15 -20" height="5"
        open-ended="true" color="purple">
        <a-animation attribute="rotation" from="0 0 0" to="0 180 180"
            repeat="indefinite"
            easing="ease-out"></a-animation>
        <a-animation attribute="material.color" from="purple" to="green"
            dur="1000" repeat="indefinite" easing="ease-in-out"></a-animation>
    </a-cylinder>
</a-scene>
```



Astuce sur les couleurs

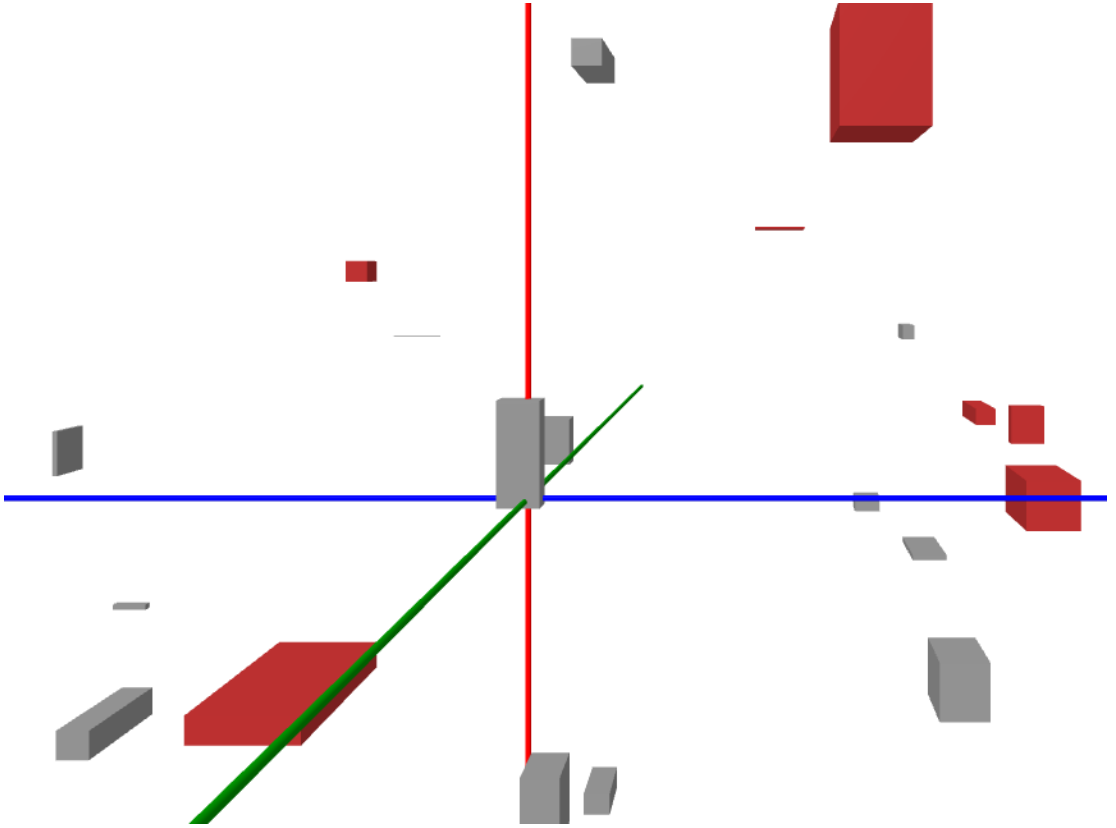
- Il est possible de personnaliser les couleurs de chaque composant avec l'attribut color :

```
<a-scene>
  <a-cylinder position="0 0 0" radius="0.02" height="10" color="red"></a-cylinder>
  <a-cylinder position="2 0.5 1" radius="0.2" height="3.5" color="#FFC65D"></a-cylinder
  <a-sky color="#ECECEC"></a-sky>
</a-scene>
```

- On a parfois besoin de jongler avec des codifications couleurs alternatives (RGB, HSL, CMYK, ...). Pour vous aider, W3Schools propose un jeu de fonctions Javascript téléchargeables sur cette page :
  - https://www.w3schools.com/colors/colors_converter.asp

Ajout de composants en JS

- On peut créer des composants A-Frame à la volée, avec Javascript. Exemple avec la création de 50 boîtes dispatchées au hasard dans l’espace (code source sur diapo suivante) :



Ajout de composants JS

```
<script>
  AFRAME.registerComponent('multiboxes', {
    init: function () {
      var scene = document.querySelector('a-scene');

      for (var i = 0; i < 50; i++) {
        var box = document.createElement('a-box');
        if (i%2) {
          box.setAttribute('material', {color: 'brown'});
        } else {
          box.setAttribute('material', {color: 'grey'});
        }
        box.setAttribute('position', {x: Math.random() * 20 - 10,
                                      y: Math.random() * 20 - 10,
                                      z: Math.random() * 20 - 10});
        box.setAttribute('scale', {x: Math.random() ,
                                   y: Math.random() ,
                                   z: Math.random() });
        scene.appendChild(box);
      }
    }
  });
</script>
<a-scene multiboxes>
  <a-cylinder position="0 0 0" radius="0.02" height="10" color="red"></a-cylinder>
  <a-cylinder position="0 0 0" radius="0.02" height="10" color="green"
    rotation="90 0 0" ></a-cylinder>
  <a-cylinder position="0 0 0" radius="0.02" height="10" color="blue"
    rotation="0 0 90" ></a-cylinder>
</a-scene>
```

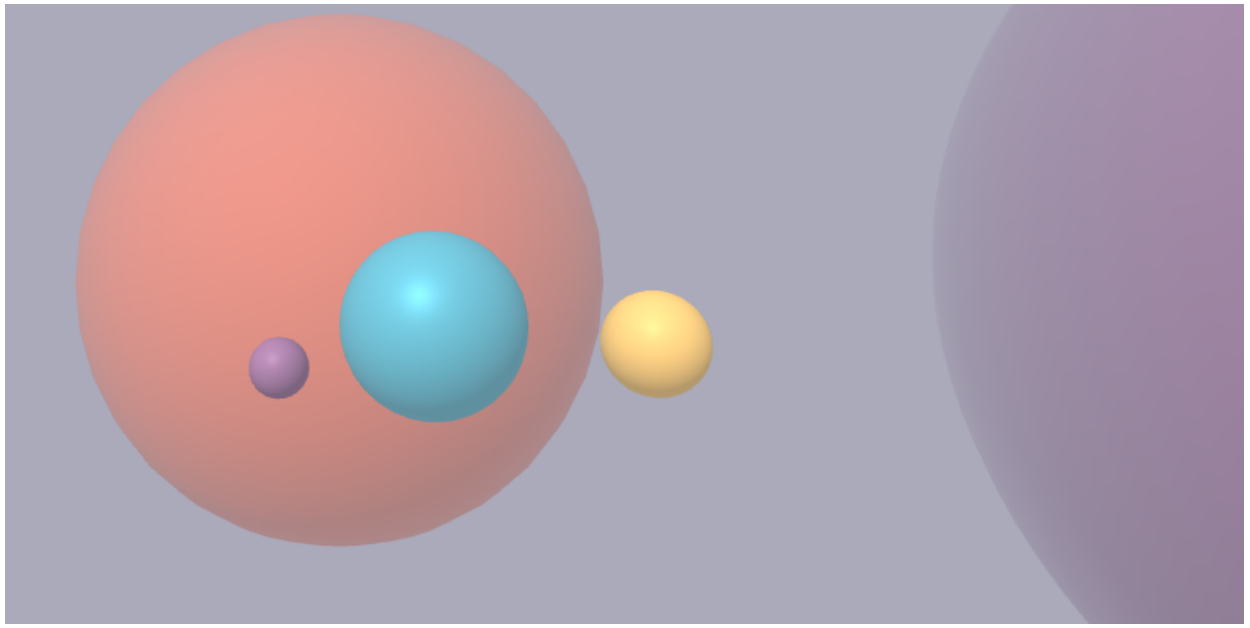
Composant « multiboxes »

Positions et échelles randomisées

Effet de brouillard

- On peut obtenir un effet de brouillard plus ou moins prononcé avec l’attribut « fog » :

```
<a-scene fog="type: linear; color: #AAB; far: 20; near: 0">
```



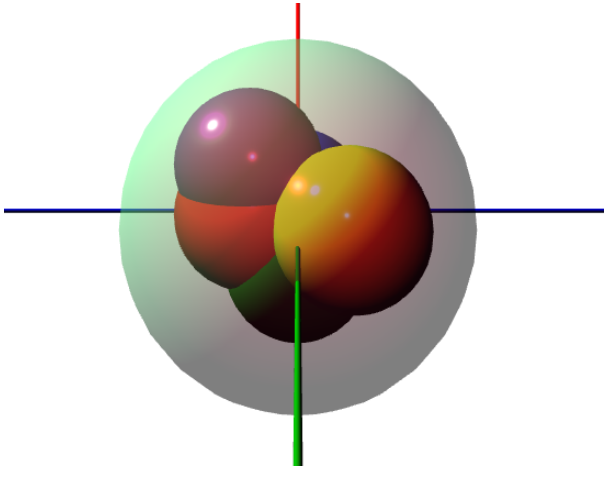
Ambiance, ambiance...

- A-Frame définit un éclairage par défaut qui est le suivant : :

```
<a-entity light="type: ambient; color: #BBB"></a-entity>
<a-entity light="type: directional; color: #FFF; intensity: 0.6" position="-0.5 1 1">
</a-entity>
```

- On peut complètement personnaliser l’ambiance d’une scène, en combinant éventuellement plusieurs sources de lumière :

```
<a-entity light="type: point; color: crimson" position="0 1.5 3"></a-entity>
<a-entity light="color: #AFA; intensity: 1.5" position="-1 1 0"></a-entity>
```



Texte en 3D

- A-Frame propose différentes manières de positionner du texte :
  - Soit très simplement en utilisant la balise a-text :

```
<a-text value="Il était une fois, dans une galaxie très lointaine..."
  position="-2 6 0" color="red" scale="1 1 1"></a-text>
```

  - Soit de manière plus avancée au moyen de mixins (voir exemple)

Mapping de texture

- Pour appliquer une texture sur une forme géométrique, on peut importer une image dans les « assets » :

```
<a-assets>
    <img id="texture" src="mandelbrot-zoom-10303.png">
</a-assets>
```

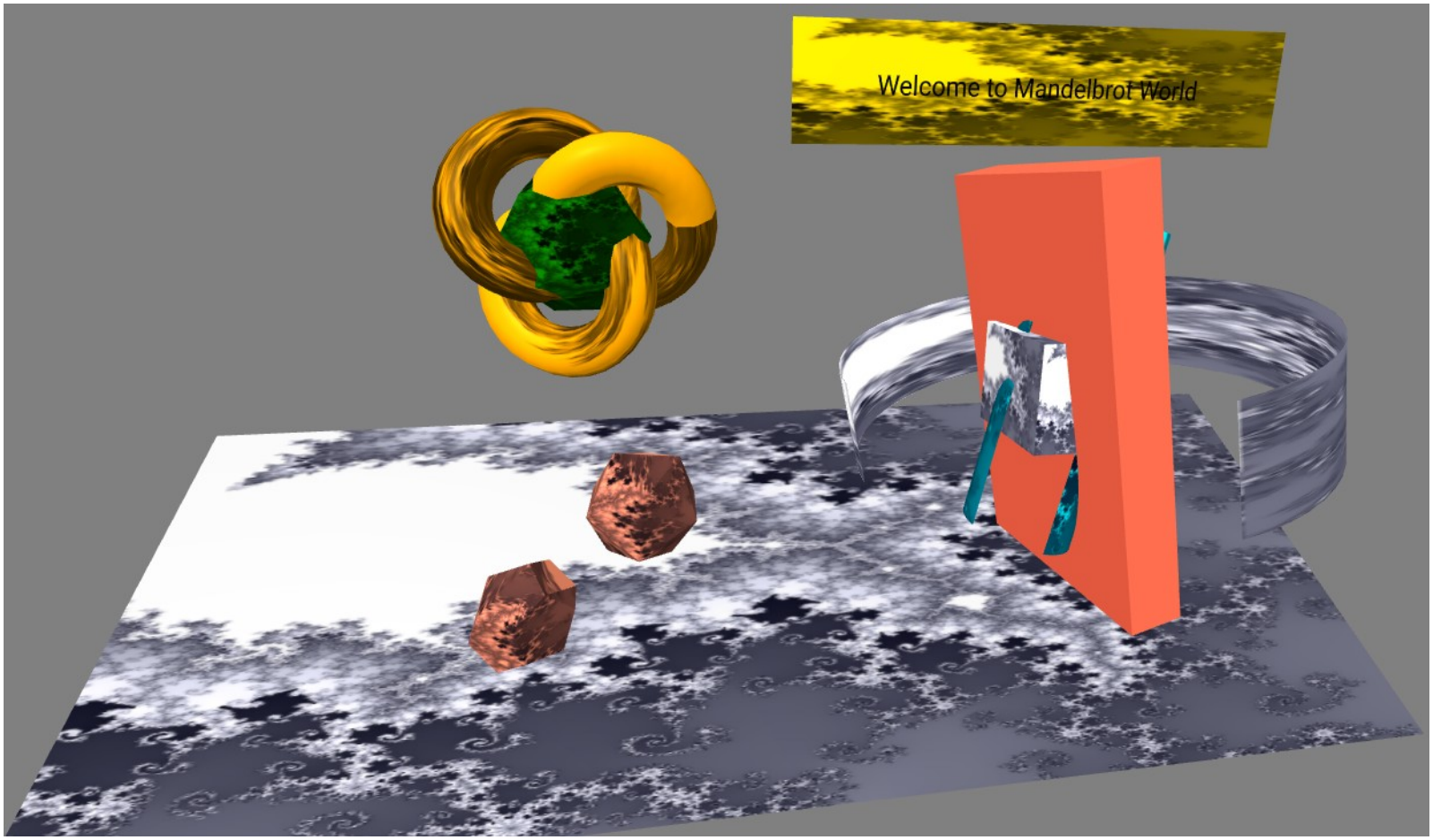
- Puis appliquer cet « asset » sur un ou plusieurs éléments de la scène :

```
<a-box src="#texture"></a-box>
```

- Cela permet de charger une image une seule fois et de la réutiliser plusieurs fois (cf. diapo suivante)

Mapping de texture

- La même image de fractale est ici plaquée sur différentes primitives



Import d’objets 3D

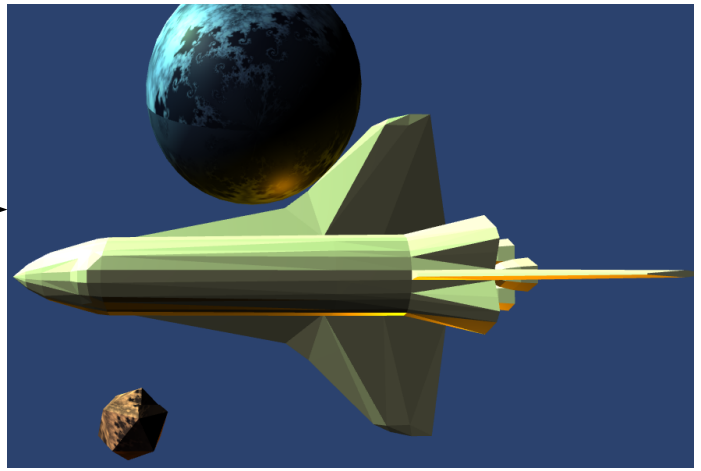
- On peut importer dans une scène des objets créés sur un modeleur 3D tel que Blender, Sketchup ou Tinkercad
- Formats d’import supportés : OBJ (Wavefront), MTL, DAE (Collada), glTF
- Fichiers OBJ prêts à l’emploi sur :
  - http://people.sc.fsu.edu/~jburkardt/data/obj/obj.html
- Nombreux formats disponibles dans différents formats sur :
  - https://clara.io/

Import d’objets 3D

- Création d’un asset d’objet 3D de navette spatiale :

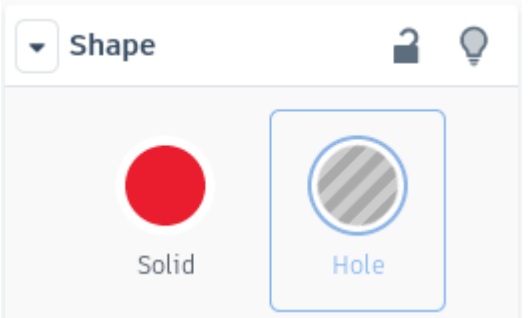
```
<a-assets>
    <!-- http://people.sc.fsu.edu/~jburkardt/data/obj/obj.html -->
    <a-asset-item id="shuttle-obj" src="assets/shuttle.obj"></a-asset-item>
</a-assets>
<a-obj-model src="#shuttle-obj" scale="0.5 0.5 0.5" color="#d0bf67"
                animation="startEvents: foo"
            light="type: directional; color: #FFF; intensity: 0.6"
            position="-1.3 1 1">
    <a-animation attribute="rotation" dur="20000" fill="backwards"
      to="-45 -270 180" id="anime-shuttle" end="foo" repeat="3">
    </a-animation>
</a-obj-model>
```

Rendu final avec quelques éléments complémentaires →



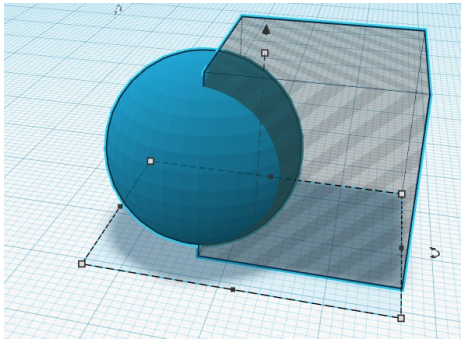
Modéliser avec Tinkercad

- Les débutants en 3D peuvent facilement modéliser leurs propres objets avec Tinkercad
- Tinkercad propose de nombreuses primitives et une fonction d’export (format OBJ et STL).
- On peut utiliser des primitives comme des ciseaux pour découper d’autres primitives, et ainsi façonner ses propres objets 3D
- L’astuce consiste à définir un objet comme « hole » au lieu de « solid »

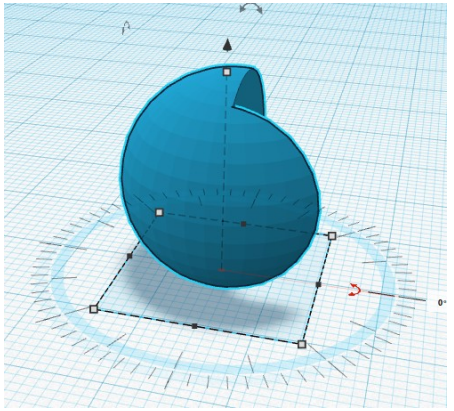


Modéliser avec Tinkercad

- Après avoir combiné un objet « solid » et un objet « hole », on les sélectionne tous les deux en les encerclant avec la souris

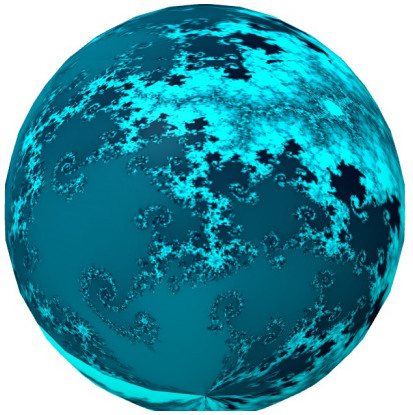


- Puis on utilise la combinaison de touches Ctrl+G, et voilà !



Gestion des événements

- On peut déclencher un effet (par exemple une rotation) par rapport à un événement tel qu'un clic de souris. Exemple avec un cylindre qui tourne quand on clique dessus :



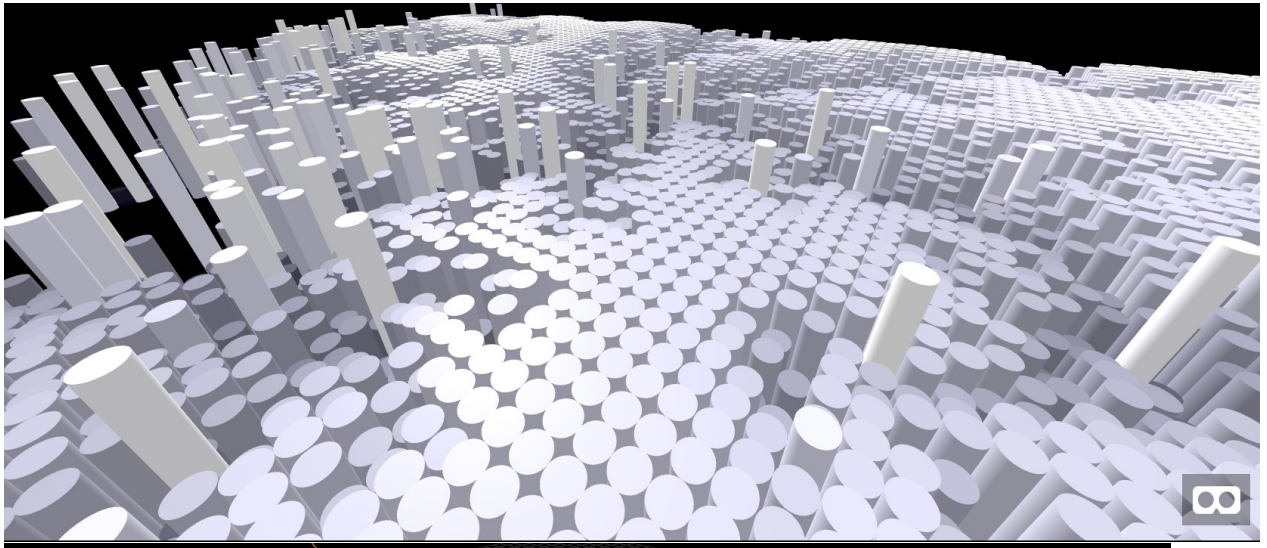
```
<script>
    AFRAME.registerComponent("sphere-anime", {
      init: function() {
        var self = this.el;
        // recherche de l'élément animation à l'intérieur de l'élément courant
        let anim = self.querySelector("a-animation");

        self.addEventListener("click", (e) => {
          console.log("cylinder-anime started");
          anim.emit("fallclick");
        }, false);

      }
    });
</script>
<a-scene cursor="rayOrigin:mouse">
  <a-assets>
    <img id="texture" src="mandelbrot-zoom-10303.png">
  </a-assets>
  <a-camera position="0 0.8 2"></a-camera>
  <a-sphere position="0 0 -2" color="aqua" radius="2"
    src="#texture" sphere-anime >
    <a-animation attribute="rotation" to="360 0 0" delay="500" dur="10000"
      easing="ease-in-out" begin="fallclick"></a-animation>
  </a-sphere>
</a-scene>
```

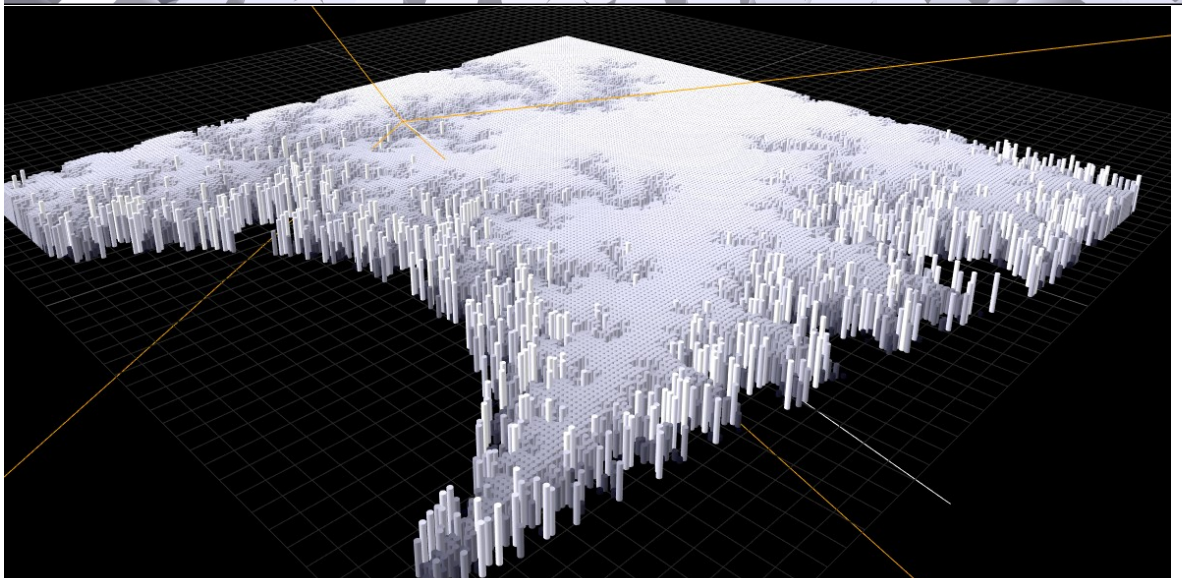
Double déclaration du composant

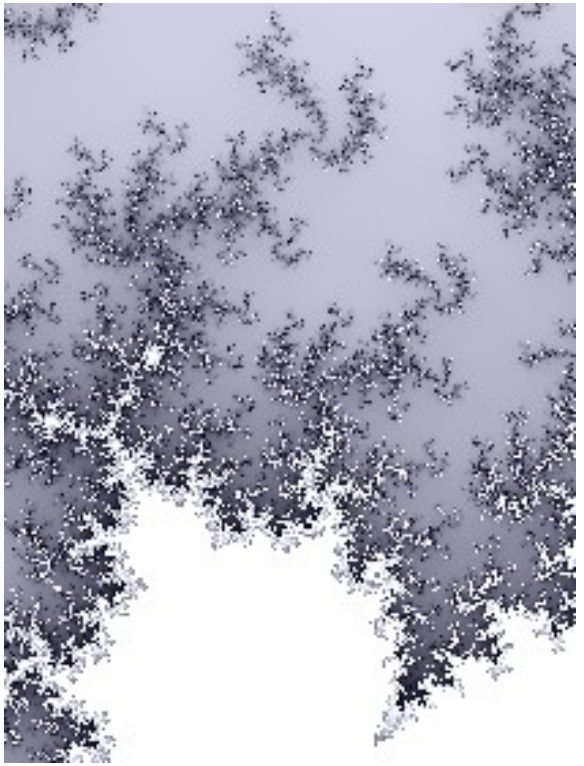
Indispensable pour activer le contrôle de la souris sur la scène courante

Fractales en 3D

- Avec A-Frame, il est facile de représenter une fractale dans des projections 3D originales :







Fractales en 3D

- Sur la diapo précédente, chaque pixel de la fractale d'origine (à droite) est représenté sous forme d'un cylindre. Chacun des cylindres a une hauteur proportionnelle à la couleur du pixel d'origine.

```
var sceneEl = document.querySelector('a-scene');
//console.log('datas => '+e.data.length);
for (var i=0, imax=e.data.length; i < imax ; i++) {
    var item = e.data[i];

    var tmp_c = item.c + 20;
    var tmp_col = "rgb("+item.c+","+item.c+","+tmp_c+")";

    var col = w3color(tmp_col, false);
    col.desaturate(.5);

    var hauteur = .01+item.c/250;

    var cyl = document.createElement('a-cylinder');
    cyl.setAttribute('material', {color: col.toHexString()});
    cyl.setAttribute('position', {x: .1+item.x/10-6, y: hauteur/2, z:.1+item.y/10-6});
    cyl.setAttribute('height', hauteur);
    cyl.setAttribute('radius', ".05");
    cyl.setAttribute('rotation', {x: 0, y: 180, z:0});
    sceneEl.appendChild(cyl);
}
```

Ça fait beaucoup de cylindres à créer prévoir l'utilisation de l'API WebWorker pour soulager le thread principal du navigateur, lors du calcul des datas (l'API WebWorker nécessite l'usage d'un serveur)

AR.js

- AR.js est une librairie développée par Jérôme Etienne, qui enrichit A-Frame en lui apportant le support de la réalité augmentée
- Site officiel :
  - https://github.com/jeromeetienne/AR.js
- Tutos et vidéos de présentation (par Alexandra Etienne) :
  - https://bit.ly/2ocIKrv
  - https://bit.ly/2yvEePe

AR.js

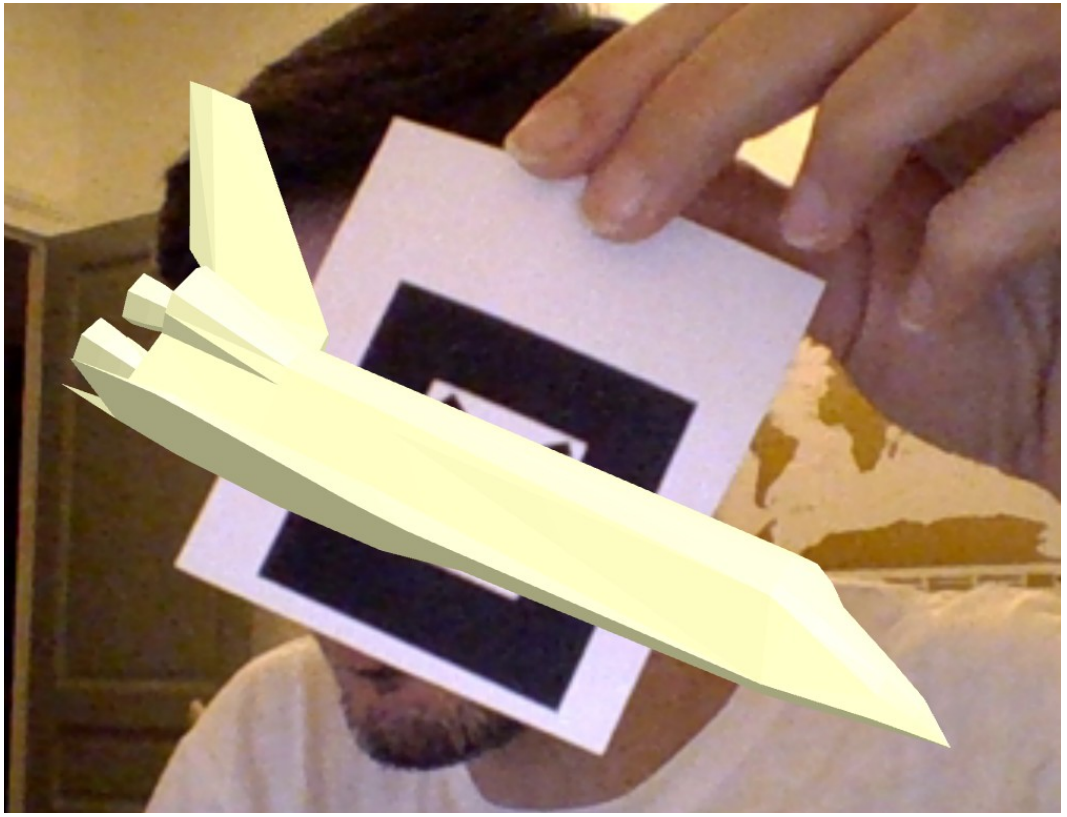
- Voici un exemple :

```
1  <!DOCTYPE html>
2  <html>
3    <head>
4      <meta charset="utf-8">
5      <title>AR with Mandelbrot Planet</title>
6      <meta name="description" content="test">
7      <script src="../js/AR.js-master/aframe/examples/vendor/aframe/build/aframe.min.js"></script>
8      <script src="../js/AR.js-master/aframe/build/aframe-ar.js"></script>
9    </head>
10   <body>
11     <a-scene embedded arjs="trackingMethod: best;">
12         <a-assets>
13           <img id="texture" src="mandelbrot-zoom-10303.png">
14         </a-assets>
15         <a-sphere radius=".8" position="0 0 0"
16                   color="silver" src="#texture">
17           <a-animation attribute="rotation" to="360 0 0" delay="500" dur="10000"
18                repeat="indefinite" easing="ease-in-out" ></a-animation>
19         </a-sphere>
20         <a-marker-camera preset="hiro"></a-marker-camera>
21         <a-sky color="grey"></a-sky>
22         <a-camera-static></a-camera-static>
23     </a-scene>
24   </body>
25 </html>
```

Marqueur Hiro détecté par AR.js

AR.js

- Activez la webcam, présentez le marqueur devant la caméra, et voilà !



AR.js

- Le marqueur Hiro est fourni avec le code source de AR.js, il est facile à trouver
- Il est possible d'afficher le marqueur sur un smartphone ce qui rend l'expérience encore plus amusante
- Il est possible d'utiliser le marqueur « kanji » à la place de « hiro », dans le code source. Ce marqueur n'est pas fourni avec le code source, mais on le trouve ici :
  - https://bit.ly/2M8d4zJ

    Pour des infos supplémentaires sur AR.js :
  - https://webxr.io/

Sujets en suspens

- Parmi les points que je n’ai pas eu le temps d’explorer et de développer ici :
  - Comment détecter les collisions entre objets ?
  - Comment gérer des dispositifs de pointage autres que la souris (leap motion, autres…)
  - L’amélioration des perfs sur des scènes avec beaucoup d’objets (comme la fractale en 3D)
  - Etc…

Conclusion

- A-Frame est un framework génial, nous n’avons fait qu’effleurer ses possibilités
- Le fait qu’il s’appuie sur ThreeJS est une chance car on peut enrichir sa VR avec toute la puissance de ThreeJS
- AR.js est amusant à utiliser et surtout très prometteur.
- A-Frame et AR.js mettent la 3D, la VR et l’AR à la portée des non spécialistes de ces techniques (développeurs amateurs, artistes, enseignants, étudiants, etc...)

MERCI BEAUCOUP