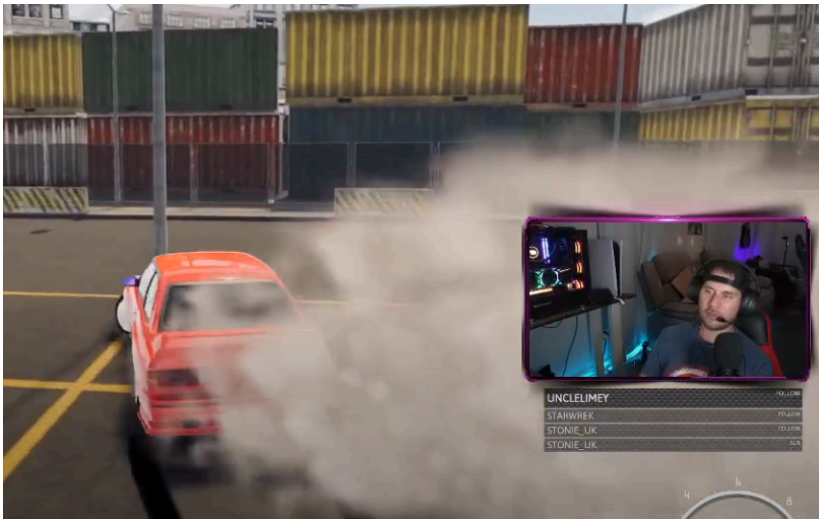
R5.DWeb-DI.07 - TP3

Le but du TP est de voir comment créer un effet de fumée qui se base sur un “système de particules”. Ce principe est très utilisé dans les jeux vidéos, par exemple pour animer la fumée dégagée par la friction d’un pneu sur la route : https://youtu.be/fVUyWyDscAY?t=189 (voir à partir de 3:05)



Ici on ne se soucie pas vraiment de la physique, l’idée principale est d’utiliser beaucoup d’éléments très simples mais qui bougent très vite, avec un peu d’aléatoire et de transparence pour empêcher de percevoir autre chose qu’un mouvement global.

Exercice 1

Commencez par découvrir le principe en 2D avec p5js :
https://www.youtube.com/watch?v=UcdigVaIYAk
Testez le code en local pour créer des particules qui montent à une vitesse aléatoire à partir d’un point de départ que nous appellerons par la suite “émetteur”. Améliorez l’exemple en faisant en sorte que l’émetteur suive la position de la souris et en ajoutant un petit effet de vent vers la droite.

Exercice 2

Transposez maintenant cet exemple en 3D avec ThreeJS en plaçant l’émetteur au centre d’un plan, et en utilisant de simples sphères qui montent en diminuant leur transparence. Même si votre code devrait n’utiliser qu’une seule instance de **SphereGeometry**, quand il faut afficher beaucoup de particules vous verrez rapidement la démo ralentir.

Exercice 3

Une solution pour améliorer les performances consiste à utiliser le type Points de ThreeJS, qui comme son nom l’indique sert à afficher des points :

- https://threejs.org/docs/index.html#api/en/objects/Points

- https://threejs.org/examples/#webgl_points_billboards

Les positions sont stockées dans une **BufferGeometry** comme celles manipulées au TP précédent. Une petite image peut être affichée à chaque position, ce qui correspond au terme de “billboard” (on parle parfois aussi d’”imposteur” ou de “sprite” comme dans les jeux vidéo) : quelle est l’image utilisée dans l’exemple ? Quelle est la caractéristique de ce type d’affichage ?

A vous de jouer pour faire monter les particules vers le haut, sachant que le tableau de positions complet doit être créé une bonne fois pour toutes au début du code (*i.e.* on ne peut pas créer de nouvelles particules à la volée comme dans les exercices précédents). Pour l’image à utiliser vous pouvez reprendre **spark1.png** disponible dans le même dossier que la démo, ce qui permettra d’obtenir un effet de “mélange” : quand deux points sont proches, leurs images respectives se superposent et créent une zone plus foncée et donc plus facile à percevoir, un peu comme des particules de fumée. Essayez d’obtenir le résultat ci-dessous en jouant sur les paramètres de transparence et de mélange du matériau : **sizeAttenuation**, **alphaTest**, **transparent**, **opacity** et **depthTest**.

Résultat à obtenir : https://mediaserver.unilim.fr/videos/08082024-122054/

Exercice 4

Dans l’exercice précédent les paramètres de taille ou de transparence sont gérés de façon globale, alors que les positions sont gérées de façon indépendante. Il est possible de rajouter un contrôle indépendant des propriétés du matériau mais c’est un peu compliqué, une autre solution est d’utiliser le type Sprite qui permet de gérer des objets indépendants (comme dans l’exercice 2) et peut lui aussi afficher une petite image au lieu d’un objet 3D (comme dans l’exercice 3) : https://threejs.org/docs/index.html#api/en/objects/Sprite

Résultat à obtenir : https://mediaserver.unilim.fr/videos/08082024-123501/

Exercice 5

Une autre amélioration consiste à utiliser comme sprite une série d’images que vous pouvez faire défiler de façon infinie. Voici un exemple Scratch avec un seul sprite : https://scratch.mit.edu/projects/12277522/

On peut adapter cette idée à notre simulation de fumée, en utilisant cet ensemble d’images : https://drive.google.com/drive/folders/1mWetMRulTqHTlkeGz2jfqodA_KzSkAXZ?usp=sharing

L’idée est que chaque sprite commence avec une image prise au hasard, puis “avance” indéfiniment dans la série (en repartant du début le cas échéant). Au final ceci permet d’avoir à manipuler beaucoup moins de sprites que dans les codes précédents, pour un résultat visuel aussi convaincant. Pour la version Threejs vous devez donc gérer un tableau de maps :

```
const loader = new THREE.TextureLoader();
for (let i = 0; i < N_images; i++) {
```

```
    maps.push(loader.load('./smoke/'
                          + String(i).padStart(4, '0')
                          + '.png'));
}
```

Vous pourrez ensuite modifier dans la boucle d'animation la map associée à un sprite :

```
sprite.material.map = maps[ … ];
sprite.material.mapNeedsUpdate = true;
```

Résultat à obtenir : https://mediaserver.unilim.fr/videos/08082024-124146/

Comme vous pouvez le voir il y a également ici un mécanisme qui place automatiquement l'émetteur au point détecté comme une intersection entre la souris et le plan horizontal, grâce au système de Raycast : https://threejs.org/docs/index.html#api/en/core/Raycaster

Exercice 6

Ajoutez un effet de fumée lors du tir au pistolet par le robot :
https://mediaserver.unilim.fr/videos/08082024-124502/