Gestion dynamique des particules

On peut créer une particule à partir d'un système de base, **(Menu Hierarchy/Create/Effects/ParticleSystem)**

Voir les vidéos sur le site du cours pour comprendre comment modifier une particule pour créer un effet.

Accès aux modules d'un système de particules par programmation

Pour modifier les propriétés d'un système de particule d'un objet, il faut :

1- obtenir le composant *ParticleSystem* et le sous-module :

GetComponent<ParticleSystem>( ).main

GetComponent<ParticleSystem>( ).emission

GetComponent<ParticleSystem>( ).shape

etc.

2- **Il faut mémoriser dans une variable le nom du module et ensuite accéder aux propriétés.**

Exemple: pour modifier la durée des particules émises alors il faut :

```
var Mesparticules = GetComponent<ParticleSystem>().main;
Mesparticules.duration = 2.0f  ;
```

Remarque: il faut faire les instructions en 2 lignes et en utilisant une variable, sinon il y aura un message d'erreur.

Pour le module *main* (GetComponent<ParticleSystem>( ).main) , les propriétés à modifier sont:

startSpeed

duration

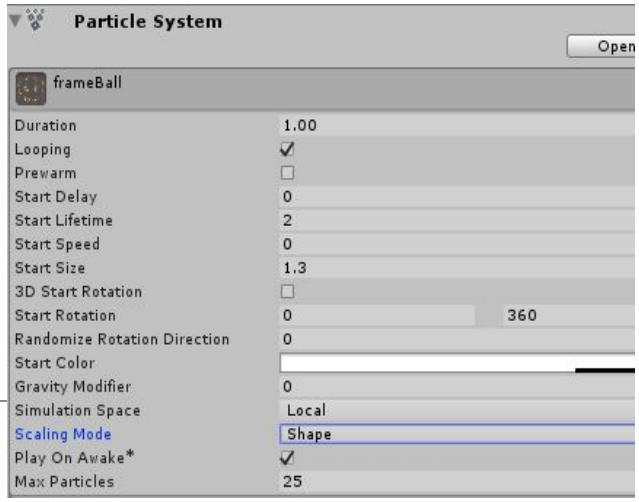
loop

startDelay

startLifetime

startSpeed

startSize

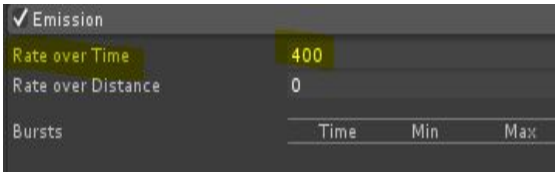


startColor

startRotation

gravityModifier et maxParticles

Pour les autres sous modules (emission, shape etc.), il faut accéder aux propriétés de la même façon.



Exemple

```
//Augmente la vitesse, la taille et la quantité des particules de neige quand la touche "p" est appuyée
public GameObject particuleNeige ;
public float augmentation = 1;   // le facteur d'augmentation de la particule neige,

void Update ()
{
   if(Input.GetKeyDown("p"))
  {
     augmentation ++;
      var neigeMain = particuleNeige.GetComponent<ParticleSystem>( ).main;

    neigeMain.startSpeed = -0.1f *  augmentation ;
    neigeMain.startSize = 0.2f * augmentation ;
    //pour modifier la quantité de particules émises alors il faut :

    var neige = particuleNeige.GetComponent<ParticleSystem>( ).emission;
    neige.rateOverTime = 200 * augmentation ;
  }
} ///Attention: les opérations de multiplication directe ( exemple: neigeMain.startSize *2) ne fonctionne pas.
```

Exemple: OnTriggerEnter(Collider info) : lorsque l'objet entre dans la zone trigger

```
// détecte les collisions du personnage,  s'il entre dans la zone de l'objet "Bombe" alors Explosion s'active
  public GameObject particuleExplosion;

   void OnTriggerEnter(Collider infoObjet)   // le type de la variable est Collider
   {
      if(infoObjet.gameObject.tag == "Bombe")
      {
         particuleExplosion.SetActive(true);
      }
   }
```

OnTriggerStay(Collider info) : tanque l'objet se trouve dans la zone trigger
OnTriggerExit(Collider info) : lorsque l'objet quitte la zone trigger

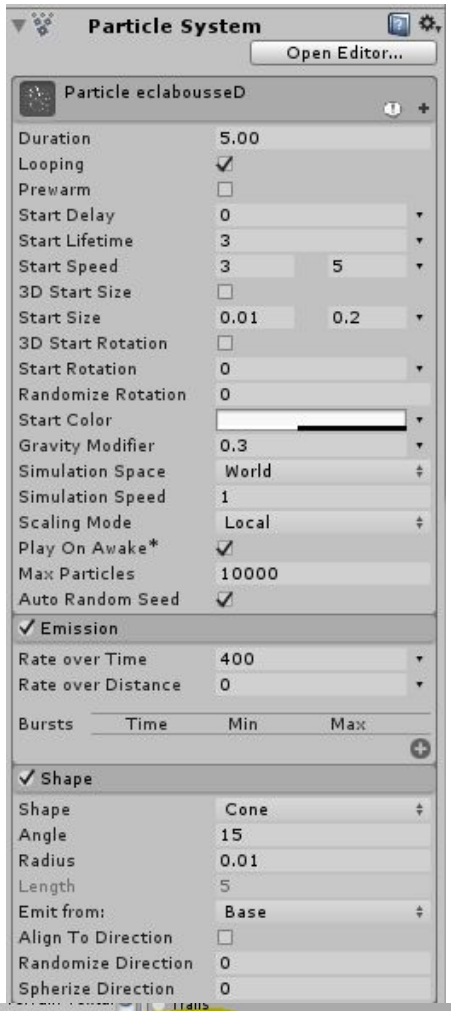
Consignes pour l'exercice

Créez deux effets de particule pour l'hélico, l'éclaboussure et le brouillard.

1- Éclaboussures: lorsque l'hélico entre en contact avec la surface de l'eau et qu'il est en mouvement, un effet d'éclaboussures d'eau est émis par ses pattes (comme dans le démo).

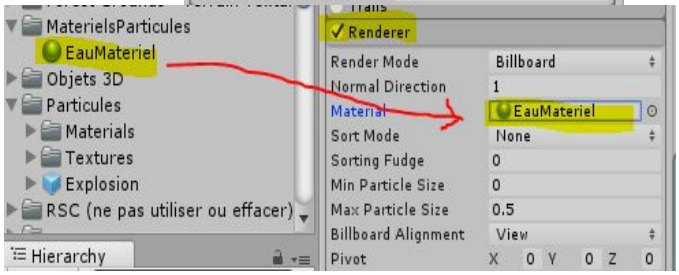
- Créez une particule à partir d'un système de base, **(Create/Effects/ParticleSystem)**
- Placez-la en avant d'une patte de l'hélico,

- Ajustez l'orientation de l'objet dans le monde et les propriétés du système dans l'inspecteur en vous inspirant de ces graphiques.



20

- Importez les textures pour l'eau, et associez le matériel "EauMateriel" au composant Renderer.



- Dupliquez la particule et placez la copie sur l'autre patte de l'hélico.

- Placez les particules dans l'objet hélico comme enfant, et désactivez-les.

- Pour détecter le début , le milieu et la fin de contact de l'hélico avec l'eau, placez un Cube invisible très grand avec un *BoxClollider* de type *Trigger* sur la surface de l'eau. (Cochez la case isTrigger du collider).

Remarque: il peut se produire des erreurs lorsque le *collider* est placé directement sur l'objet Eau.

- Dans le script de l'hélico, déclarez les variables et les fonctions pour accéder aux objets particules que vous avez créés.

- **OnTriggerEnter**: active les particules
- **OnTriggerStay** : permet de déterminer la quantité de particules qui sont émises. Elle dépend de la vitesse de l'hélico. Si la vitesse est 0 alors rateOverTime est égale à 0, sinon rateOverTime est proportionnel à la vitesse de l'hélico (+ ou - ou * UneValeur).
- **OnTriggerExit** : désactive les particules

2- Effet de brouillard:

lorsque l'hélicoptère est près de l'eau, un effet de brouillard apparaît. La densité de brouillard dépend de la distance entre l'hélico et l'eau. La distance peut être obtenue par la fonction :

Physics.Raycast(position , direction, out infoCollision, longueur)

Cette fonction trace un rayon invisible, si un objet ayant un COLLIDER est touché par le rayon alors la fonction retourne la valeur "vraie" et elle retourne les infos sur la collision dans une variable (ici infoCollision).

Quatre paramètres sont nécessaires :

1- La position d'origine du rayon (un Vector3, qui peut être le transform.position de l'objet du script)

2- La direction du rayon (Vector3.up ou Vector3(0,1,0) vers le haut, -Vector3.up vers le bas, transform.forward, vers l'avant selon l'axe z de l'objet, transform.right vers le côté selon l'axe x de l'objet) .

3- La variable infoCollision contiendra les informations si le rayon touche un objet, (**variable de type RaycastHit**)

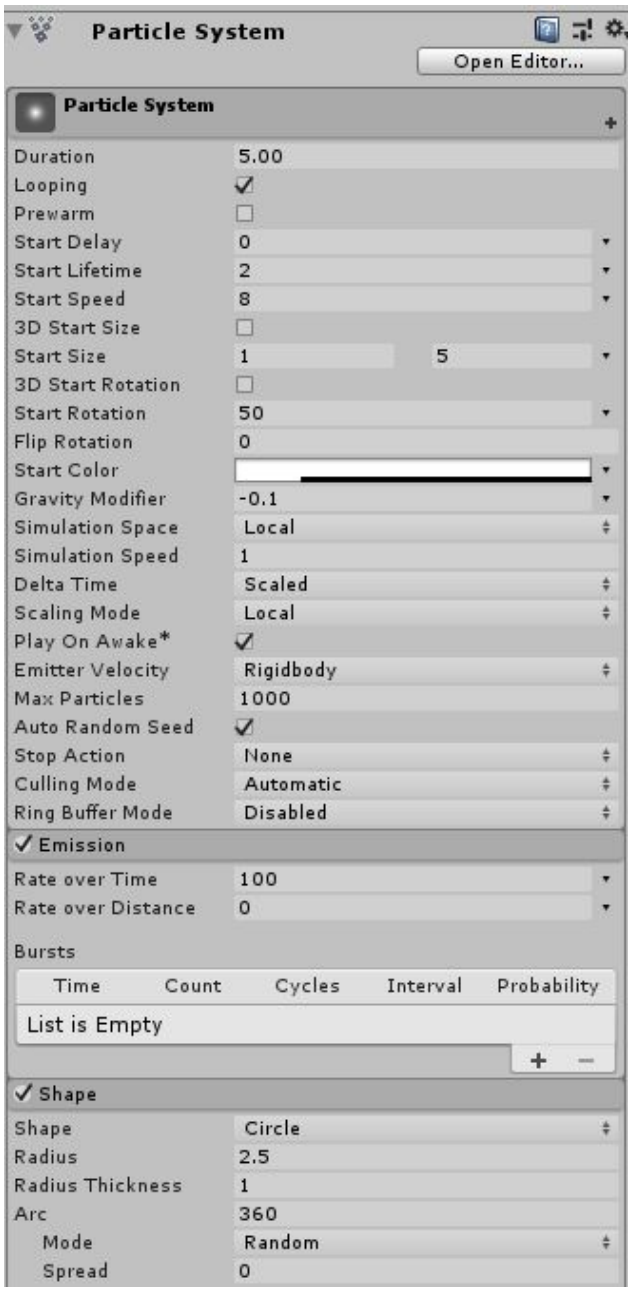
- infoCollision.collider.name : le nom de l'objet touché (l'objet doit avoir un Collider)
- infoCollision.collider.gameObject : l'objet touché (l'objet doit avoir un Collider)
- infoCollision.distance : la distance à laquelle un objet a été touché, (en mètre)
- infoCollision.point : la position de contact entre le rayon et l'objet touché, (en Vector3)
- infoCollision.point.y : la hauteur de contact entre le rayon et l'objet touché,

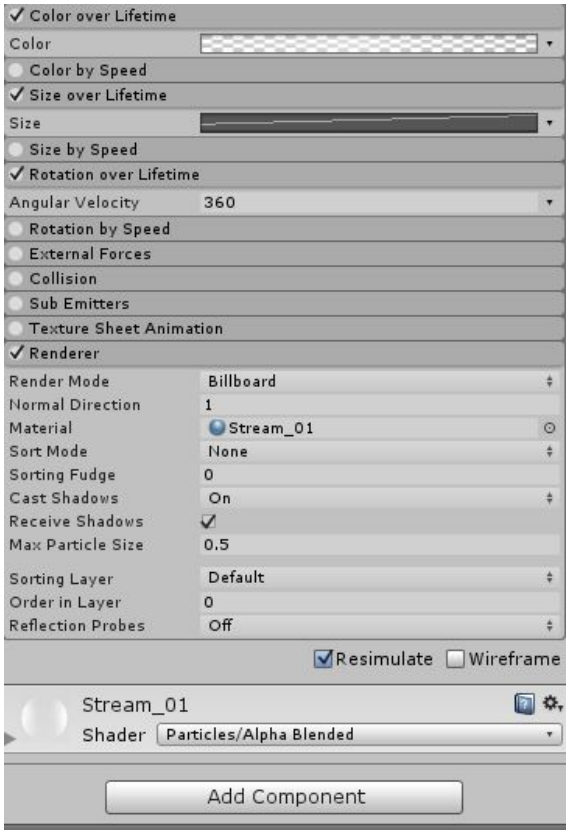
4- La longueur du rayon en mètre.

Exemple

```
void Update ()
{
        RaycastHit infoCollision ;
        // si un objet est touché la fonction retourne "vraie"et les infos dans infoCollision ("out"  devant les valeurs de retour)
        // la position de départ du rayon est celle de l'objet du script
         if (Physics.Raycast(transform.position, new Vector3(0,-1,0), out infoCollision, 20) )
            {
               print(infoCollision.collider.name);
               if (infoCollision.collider.name == "Eau")
               {
                  print(infoCollision.distance);
                  print(infoCollision.point);
               }
            }
            else
           {
              print("aucun objet touché sur la distance de 20 mètres");
           }
           // on peut dessiner la ligne de détection pour débogage
          //Debug.DrawLine(  positionDepart    ,                 positionFinale                        , couleur de ligne
            Debug.DrawLine(transform.position, transform.position + new Vector3(0,-20f,0), Color.red);
}
```

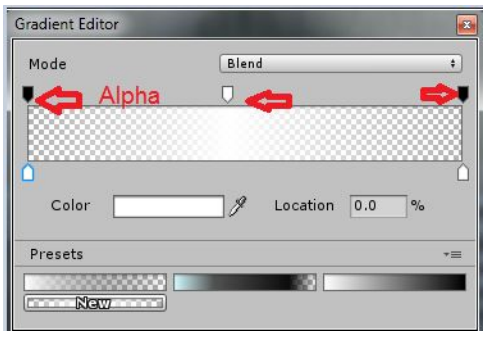
- Créez un système de particule (*BrouillardParticule*) et placez-le sous l'hélicoptère. Cet objet doit aussi être l'enfant de l'hélicoptère. (voir à la page suivante les différentes propriétés)





- La propriété ColorOverLifeTime peut être ajustée pour permettre aux particules de devenir visible graduellement et devenir invisible graduellement (alpha 0) vers la fin de leur durée.

  (alpha 0 au début, 150 au milieu, 0 à la fin)



- Modifiez le matériel de la particule dans le module Renderer. Utilisez le matériel *Steam* que vous avez importé.

- Créez une fonction dans le script de l'Hélico (EmetBrouillard()), cette fonction est appelée continuellement dans le Update().

- Cette fonction trace un rayon d'environ 20 mètres à l'aide de la fonction Raycast, vers le bas.

- Si l'eau est touchée par le rayon, ajustez la quantité de particules émises et leur vitesse en fonction de la distance qui sépare l'hélicoptère et l'eau. Modifiez les propriétés:
  - *rateOverTime* de la particule qui doit émettre peu de particules si la distance de contact est grande et plus de particules si la distance est petite. (ex: 100 - (distance de contacte * ?))
  - *startSpeed* qui doit être petite si la distance de contact est grande et grande si la distance est petite.

  (vitesse max de particule) - (distance de contacte *??)). //quand la distance est 20, le résultat doit être 0

- Pour permettre que l'émission du brouillard soit sur la surface de l'eau, ajustez la position de *l'objet particule* de brouillard (objetBrouillard.transform....) en utilisant la position de contact obtenue par le RayCast.

- Si le rayon ne touche pas à l'eau (la commande Raycast est alors « false »), la propriété *rateOverTime* doit être mise à zéro.

Remise de l'exercice (vendredi le 28 avant minuit)

- Créez un dossier à votre nom (Nom_Prénom). Ce dossier devra inclure :
  - Un **exécutable** de votre exercice (pas de WebGl)
  - Un dossier "scripts" contenant une copie de tous les scripts de votre projet. **Le code doit être indenté et commenté.**
  - Remettez ce dossier → Remise/tim/Vahik/447 Assemblage jeu/Exr2/

Critères de corrections :

- les fonctionnalités exigées dans les exercices de collisions, de l'interface UI, d'animations et des particules. (9 points)
  - collisions (2 points)
  - interface UI (2 points)
  - animations (1.5 points)
  - particules (1.5 points)
- Indentation et commentaire du code (1 point)
  - une description globale du code au début de chaque script avec date et nom;
  - description des variables dans chaque script, lors de la déclaration;
  - description au début de chaque fonction ;
  - description pour chaque condition ou boucle ;