Les systèmes particules sous Ogre

Introduction

Les particules sont des éléments générés par un émetteur et représentés par un imposteur.

La gestion des particules dans Ogre se fait à base de scripts dont l’extension est « particle ». Ces scripts sont rédigés en suivant le même concept que les matériaux. Le format de ces scripts est du pseudo-c++. Les sections sont délimitées par des accolades et le commentaire se fait avec un double slache.

La première section est spécifique au type de script. Il s’agit de « particle\_system » pour les particules. Les paramètres les plus fréquents sont :

matrial : utilisé pour indiquer quel matériau est utilisé pour le rendu de chaque particule.

particle\_width et particle\_height : utilisées pour préciser les tailles de chaque entité.

Quota : indique le nombre maximum de particules à un instant donné. Pour éviter une interruption involontaire de l’effet, car il y a trop de particules, il convient d’adapter le quota au nombre de particules émises au taux d’émission et à leur durée de vie.

Pour cette section, il existe d’autres paramètres permettant de configurer l’imposteur, si les particules doivent être ordonnées, masquées les unes par rapport aux autres, etc.

Pour la génération des particules, Ogre met à notre disposition différents types d’émetteur tel que le point, la boite, le cylindre, l’ellipsoïde, l’anneau et l’ellipsoïde creuse. Tout système de particules peut avoir plusieurs émetteurs distincts et indépendants. Pour chacun d’entre eux, il convient d’indiquer le nombre de particules par seconde, leur durée de vie… Pour que le rendu ne soit pas monotone, certains paramètres comme la durée de vie ou la vitesse de déplacement peuvent être mentionnés sous la forme d’une plage.

Ce n’est pas parce que la particule est émise qu’elle n’est plus modifiable. Il existe des sections « affector » de différents types comme : « Rotator » ou « Scaler » qui permettent respectivement de programmer un comportement de rotation et de changement d’échelle dans le temps.

Comme mentionné au début, un système de particules se voit affecté un matériau. Également paramétrable sous la forme de script, il est possible d’appliquer aux matériaux d’autres programmes exécutés cette fois par le moteur de rendus, ce sont les shaders. Parmi les programmes existants, on retrouve le glsl pour l’OpenGL, l’hlsl de DiectX et le CG de Nvidia.

Comme on peut le voir, créer un bel effet de particules nécessite d’ajuster de nombreux paramètres. Comme il est extrêmement agaçant d’exécuter son application pour constater la modification apportée, il existe une application pour visualiser et paramétrer ses effets, il s’agit de Ogre Particle Lab. Cependant attention, cette application utilise OpenGl comme moteur de rendu, de ce fait, certaines particules présentent dans sa bibliothèque ne sont donc pas utilisables partout.

Les effets de particules pour Short-Stroy

Notre application utilise trois systèmes de particules différents, il y a la pluie, le verre brisé et les flammes sous le robot.

La pluie

Notre pluie se veut abondante, dense et elle doit rester à l’extérieur. Cette brève description montre bien les problèmes à régler. Pour avoir une pluie abondante et dense, il faut augmenter le nombre de particules émises en même temps et une vitesse élevée. Le but est que le joueur perçoive la pluie en permanence sans qu’il puisse compter les gouttes.

La gestion du nombre de particules

Mettre un émetteur au-dessus de toute la surface du terrain et suffisamment haut pour dépasser l’immeuble est l’approche naïve. Le nombre de particules à émettre est bien trop important pour que cela soit supportable par l’application, le lag sur les ordi-centre est perceptible. De plus cela pose un autre problème, il pleut dans le bâtiment.

En effet, comme tout objet de la scène, sans précision, les particules traversent tout. Sous Ogre, il n’y a pas de boite d’exclusion et l’idée de faire une détection de collision pour chaque élément est à ranger au côté des contes et légendes. Bref cette solution est mauvaise.

Nous avons donc attaché notre système de particules au nœud de la caméra et du personnage, ainsi l’émetteur peut être beaucoup plus petit et les nombres de particules à générer pour le même résultat sont tout à fait convenables. Il y a tout de même quelques précautions à prendre.

Une fois que les particules sont émises, elles deviennent indépendantes de leur émetteur. Déplacer ce dernier ne change donc pas la position des particules existantes. Pour éviter que le joueur ne fasse la course avec la pluie, c’est à dire qu’il sorte de la zone de retombée des goutes, il faut que l’émetteur soit une boite qui englobe la caméra sur une distance suffisante.

Pour limiter le nombre de particules, on peut encore jouer sur la hauteur de la boite émettrice. Inutile quelle fasse la taille de l’immeuble, mais si elle est trop petite une caméra qui regarde vers le ciel verra les goutes se créer.

Afin d’améliorer l’effet de désastre naturel, on ajoute un angle à l’émetteur pour incliner les gouttes. On déforme l’image pour que celle-ci soit effilée afin que cela corresponde à la vitesse de chute, et on l’oriente avec le même angle que l’émetteur.

À présent nous avons une pluie, bien comme il faut.

La gestion du bâtiment

Nous avons réglé le problème du nombre, reste celui de l’immeuble. À moins qu’il ne soit délabré, il faut éviter qu’il pleuve à l’intérieur. Pour cela rien de plus simple, le lancer de rayon qui gère la hauteur entre le sol et la caméra, sait sur quel objet il est projeté. Si c’est sur l’immeuble, alors on coupe la pluie.

Afin que le joueur constate que la météo ne s’arrange pas quand il regarde par une fenêtre, on affiche une bande de pluie sur les faces de l’immeuble qui possèdent des vitres.

Si cette technique devenait trop gourmande, il est possible d’ajuster la hauteur des émetteurs en fonction de la hauteur de la caméra dans l’immeuble. Mais cela n’a pas été nécessaire.

Le verre brisé,

Une fenêtre qui casse, ça fait des morceaux de taille variable et certains ne tombent pas. Pour avoir un émetteur de meshs, il faut le créer. Ce qui revient à patcher le plugins ParticleFX. Le seul pose/site/wiki encore accessible traitant de ce sujet se trouve sur le forum d’Ogre3d ( <http://www.ogre3d.org/forums/viewtopic.php?f=2&t=19606>) par contre il est écrit pour la version 1.0.6 de OgreSDK, et on utilise la 1.8.1, il y a plus de six ans d’écart et de nombreuses modifications. Nous avons donc changé de principe … Nous allons casser une vitre faite avec du verre sécurité.

Le gros avantage de ce type de verre, c’est qu’il se casse en plein de petits morceaux de taille homogène et qu’ils tombent tous en même temps. L’inconvénient réside dans le nombre de particules à générer subitement.

Comme les morceaux sont d’aspect irrégulier et que leurs côtés reflètent la lumière, il faut un matériau qui utilise une image au contour flou, car avec la vitesse de chute des débits, les facettes n’apparaissent pas distinctement.

Au niveau de l’émetteur, les couleurs de particules doivent changer dans le temps, car les reflets évoluent et l’impact de la lumière aussi. C’est pourquoi les couleurs sont des variations de bleu marine, bleu bris et un peu de violet.

L’émetteur plan est placé au même niveau que la fenêtre, et quand celle-ci est masquée, l’effet de particule est affiché.

Les réacteurs sous les pieds du robot

À l’heure où j’écris ces lignes, à défaut d’avoir un chat joueur, nous avons un robot volant. Cette modification du scénario vient du fait que le chat et ces animations n’existent pas. Mais ce détail n’est pas l’objet de ce chapitre.

Pour qu’un robot vole, il lui faut un jet pack. Et bien à présent, le mesh « robot » qui existe par défaut dans Ogre, en possède un dans chaque pied.

Nous avons donc modifié un système de fumée existant dans Ogre pour que celui-ci émette des particules avec un angle de dispersion moins large, un jaune plus instance à la base et un sens d’émission vers le bas.

Il ne nous reste plus qu’à attacher nos effets (car il y a deux pieds) aux nœuds du robot pour que le déplacement de l’un entraine celui des autres. Afin de faire les choses convenablement et que le déplacement se passe bien, les systèmes sont rattachés aux os du robot. On sait qu’il possède un squelette, car il est animé, mais il reste à découvrir le nom des nœuds qui nous intéresse. Il existe un logiciel nommé ogreMeshy, son rôle est de montrer les animations des meshs, leur orientation, la position du squelette et le nom des nœuds. Grâce à lui, les particules doivent être attachées au nœud « Join8 » et « Join4 ». Cette affectation est faite à l’initialisation de l’animation et elles sont désactivées.