Eidos project

Abstract. Computing rotations is a problem for animation in DCC. In his paper about my process for responding in my problem: recreate cinema vfx effect in to video game.

Key words. Animation, Quaternions, SLERP, LERP, Retro-Engineering

1 Introduction

With the aim of creating animations that are fluid and versatile in their employability, I have divided my process into several sections:

In section 1. an activation module switchable between activation by distance from third-party object(s) or activation by frame range integrated into the HDA parameters.

Section 2. A set of parameters for transforming the given object for use in the LERP.

In section 3. The central node that computes SLERP and LERP in parallel, with distinct and efficient options for ease of use.

In section 4. A node called TimeStamp, which allows you to copy the animation with a delta in time and a defined number of iterations, while having the complete geometry from the start to avoid creating geometries.

In section 5. Shading management by comparing orient's and their delta, to avoid overlapping Prim's by injecting the displayed geometry into vertex colors.

Eidos project

Abstraction. Le calcul des rotations est un problème pour l'animation en DCC. Dans cette article, je décrit mon processus pour répondre à une problèmatique donné : recréer un effet vfx de cinéma dans un jeu vidéo.

Mots-Clées. Animation, Quaternions, SLERP, LERP, Retro-Engineering

1 Introduction

Dans le but de créer des animations fluides et versatiles dans leur employabilité, j'ai réfléchi mon process en plusieurs sections :

Dans la section 1. Un module d'activation switchable entre une activation par distance d'objet(s) tierce(s) ou une activation par frame range intégré dans les paramètres du HDA.

Dans la section 2. Une série de paramètres de transformation de l'objet donné pour servir au LERP.

Dans la section 3. Le node central qui compute le SLERP et le LERP en parallèle avec des options distinctes et efficaces pour une utilisation simple.

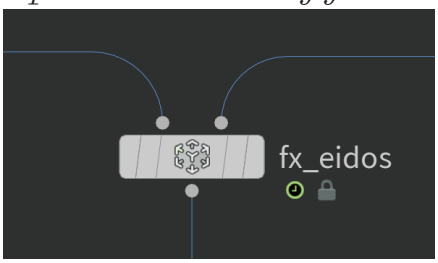
Dans la section 4. Un nœud nommé TimeStamp qui permet de copier l'animation avec un delta dans le temps et un nombre d'itérations défini, tout en ayant la géométrie complète dès le début pour éviter la création de géométries.

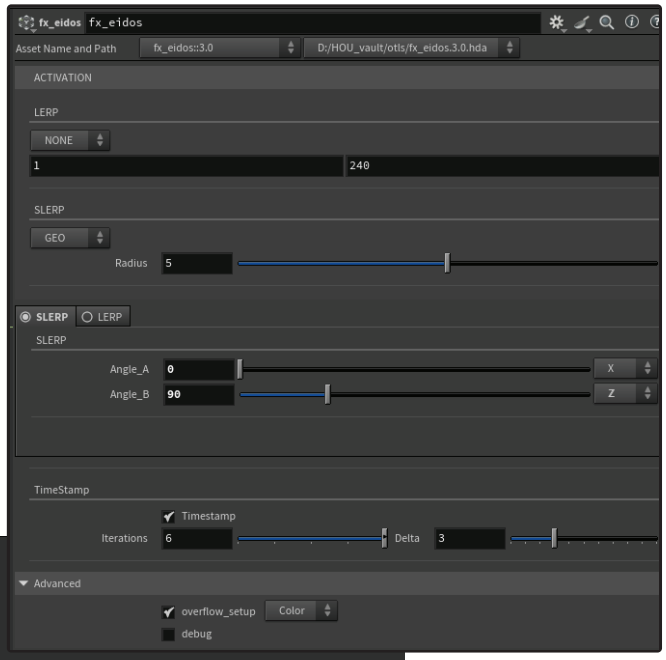
Dans la section 5. La gestion du shading grâce à la comparaison des orient's et de leur delta qui permet, entre autre, de ne pas avoir d'overlapping des Prim's en injectant dans les vertex color la géométrie affichée.

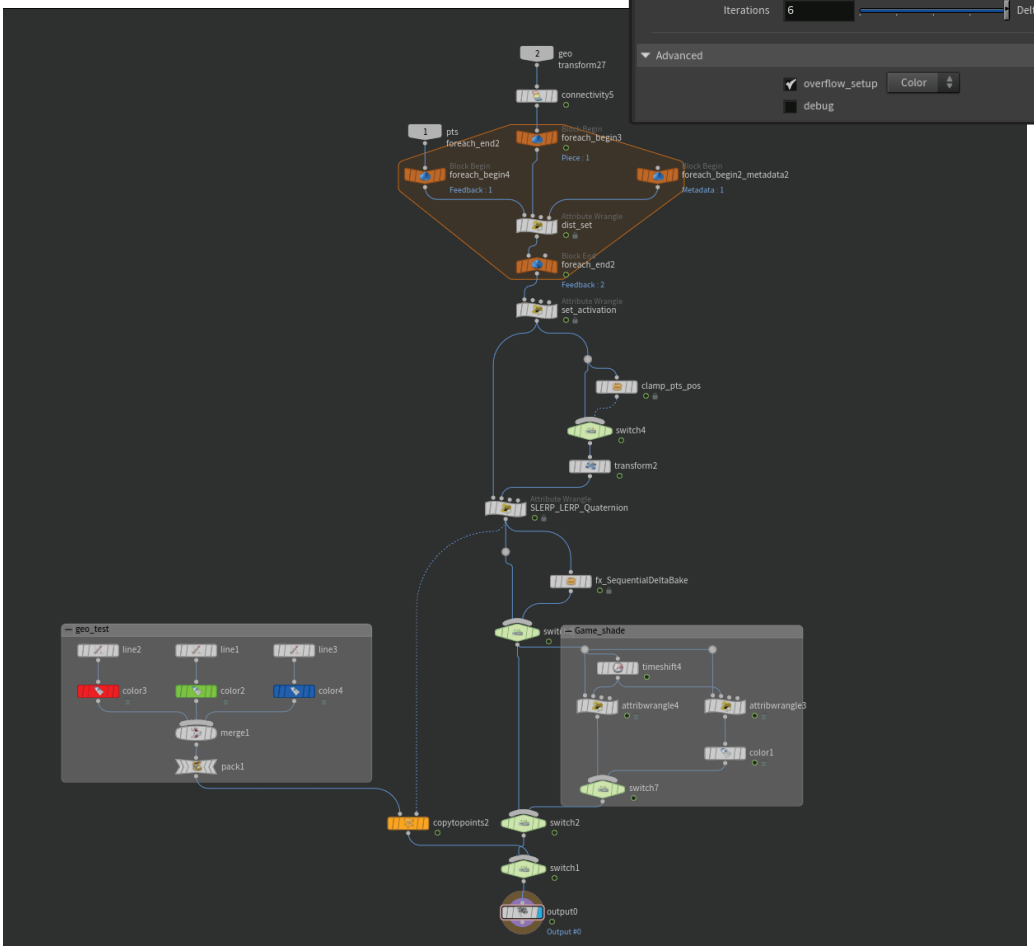
0 · Houdini Digital Assets

Voici les paramètres du HDA. je l'ai pensé dans le but d'avoir une simplicité d'utilisation et pour pouvoir essayer rapidement différentes configurations et également pour pouvoir l'incorporer dans un workflow PDG

Input 0 : pts target
Input 1 : Activation by geo







1 · Activation

Dans la section 1. Un module d'activation switchable entre une activation par distance d'objet(s) tierce(s) ou une activation par frame range intégré dans les paramètres du HDA.

Calculate distance between in to centroid geo and pts :

a = n°pts de l'input 0
b = n°pts centroid des geo externe
len = a-b
$F(len) = \sqrt{(x\_a*x\_a + y\_a*y\_a + z\_a*z\_a)}$

Dans un second temp nous géront les parametres d'activation:

```
float frame_a = fit(@Frame,ch('../time_ax') ,ch('../time_ay'),0,1);
float frame_b = fit(@Frame,ch('../time_bx') ,ch('../time_by'),0,1);
```

Puis gestion des toggle pour le HDA

```
int toggle_a = chi('dist_or_frame');
int toggle_b = chi('dist_or_frame_b');

if(toggle_a == 1){
   f@mask_lerp =  fit(@__dist,0,ch('../rad_a'),1,0) ;
}
if(toggle_a == 2){
   f@mask_lerp = frame_a;
}

if(toggle_b == 1){
   f@mask = fit(@__dist,0,ch('../rad_b'),1,0);
}
if(toggle_b==2){
   f@mask = frame_b;
}
```

2 · Transform

Principale function:

vector4 slerp(vector4 q1, vector4 q2, float bias)
def : Blends between quaternions ‹q1› and ‹q2› based on the ‹bias›.

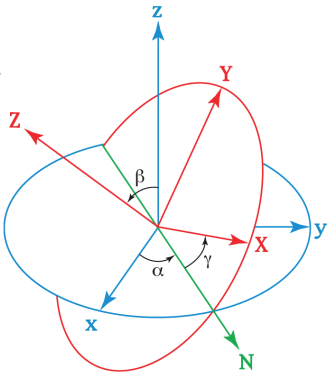
3 · SLERP - LERP

Dans la section 3. L'utilisation combinée du SLERP (Spherical Linear Interpolation) et du LERP (Linear Interpolation) dans une animation permet de gérer simultanément les transitions entre rotations et positions avec fluidité et précision. Voici pourquoi et comment ces deux techniques sont employées en parallèle

Principale function:

vector4 slerp(vector4 q1, vector4 q2, float bias)
def : Blends between quaternions ‹q1› and ‹q2› based on the ‹bias›.

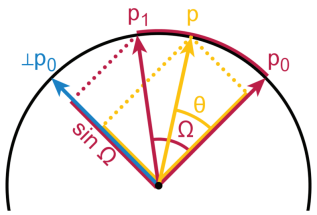
float lerp(float value1, float value2, float amount)
def : Performs linear interpolation between the values.




```
float LERP = f@mask_lerp;
@P = lerp(@P,point(1,"P",@ptnum),LERP);
float angle_degrees_a = chf('degrees_a');
int axis_a =  chi('axis_a');
float angle_radians_a = radians(angle_degrees_a);
vector4 a = set(0,0,0,0);
vector4 b = set(0,0,0,0);

if(axis_a==0){
   p@a = quaternion(angle_radians_a, set(1,0,0));
}else if(axis_a==1){
   p@a = quaternion(angle_radians_a, set(0,1,0));
}else if(axis_a==2){
   p@a = quaternion(angle_radians_a, set(0,0,1));
}
float angle_degrees_b = chf('degrees_b');
int axis_b =  chi('axis_b');
float angle_radians_b = radians(angle_degrees_b);

if(axis_b==0){
   p@b = quaternion(angle_radians_b, set(1,0,0));
}else if(axis_b==1){
   p@b = quaternion(angle_radians_b, set(0,1,0));
}else if(axis_b==2){
   p@b = quaternion(angle_radians_b, set(0,0,1));
}
float SLERP = @mask_slerp;
p@orient = slerp(@a,@b, SLERP);
```



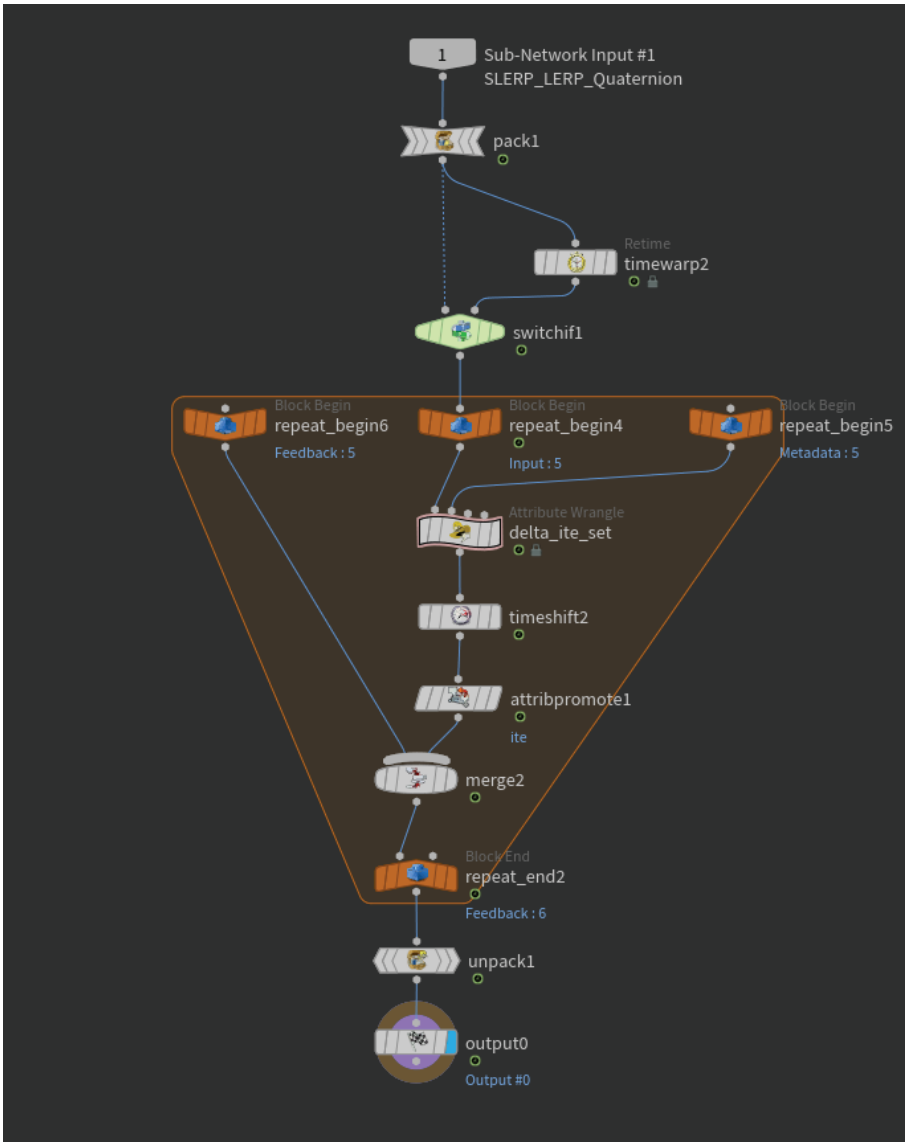
réferences : https://citeseerx.ist.psu.edu/document

4 · Sequential Delta Bake

Dans la section 4. L'objectif principal est de compiler plusieurs itérations d'une animation source en les disposant de manière ordonnée sur une timeline unique. Chaque itération est espacée des autres par un delta de frame défini.

n = *Nombre d'itération*
Δt = *Delta de frame*
T = *Timeline globale*
i = *itération*

$$t\ start, i = i \cdot (\Delta t + T),$$



5 · Shading

Dans la section 5. Le module prévient les chevauchements de géométries en calculant les deltas entre leurs matrices de transformation , détectant les overlaps via une distance minimale, et appliquant un attribut color dans les vertex colors.

```
vector prev_orient = point(1, "orient", @ptnum);
    @Cd.x = 1;
    @Cd.y = 1;
    @Cd.z = 1;
    @group_fix =0;
if (distance2(@rient, prev_orient) < 0.000001) {
    if(@ite != 0){
        setpointgroup(0, "fix", @ptnum, 1);
    }
}
```