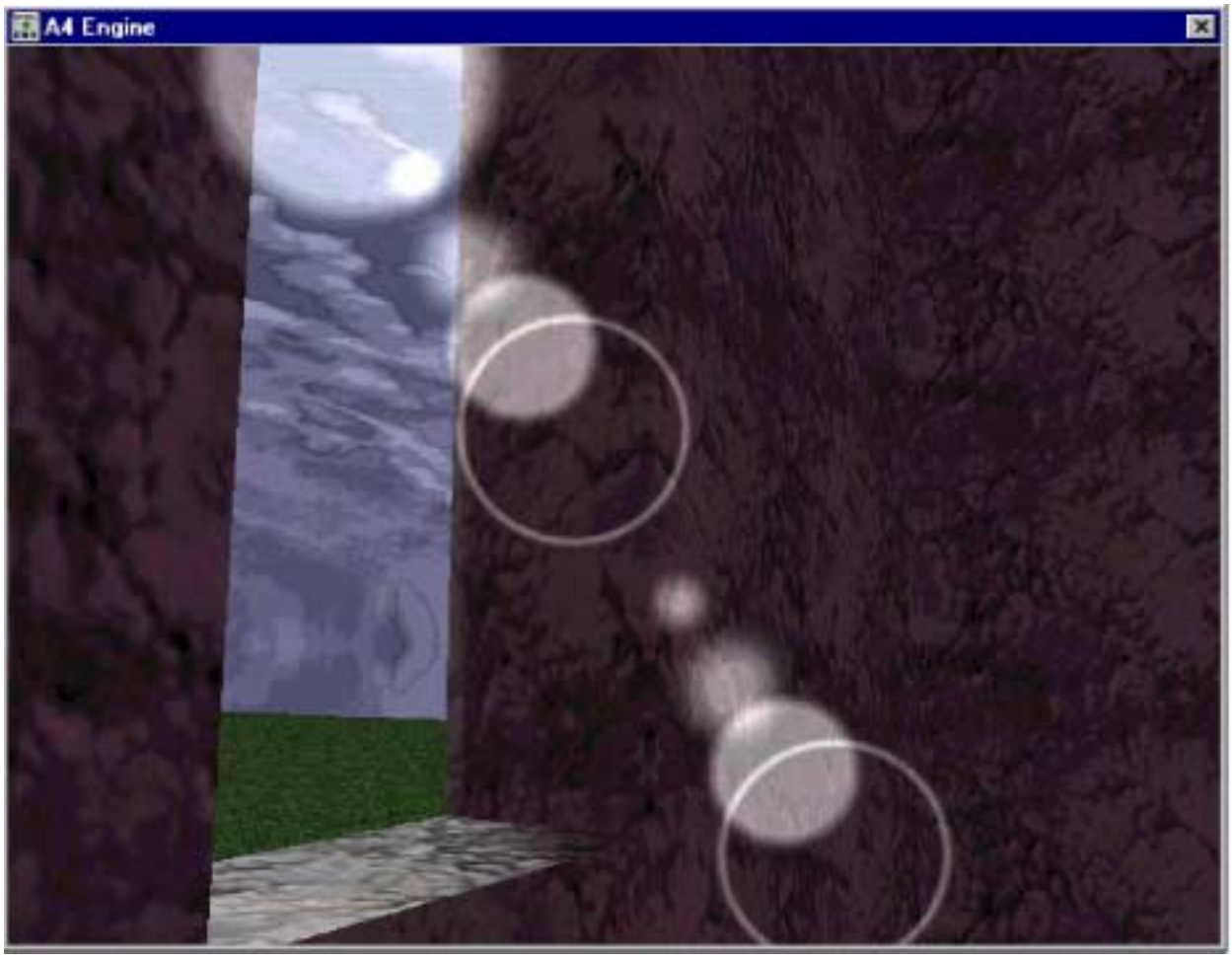
# Les mini-ateliers de 3D Gamestudio

## Les éclats de lentilles




par Doug POSTON Mars 2001

Les dernières nouvelles, les démonstrations, les mises à jour et les outils, aussi bien que le Magazine des Utilisateurs, le Forum des Utilisateurs et le concours annuel sont disponibles à la page principale GameStudio http://www.3dgamestudio.com/

## Avant propos à propos de la traduction française de ce manuel

La traduction est un art difficile. D'autant plus qu'il m'a fallu apprendre le moteur au fur et à mesure de la traduction. Malgré tous mes efforts il est clair que tout n'est pas parfait. Je vous invite donc lorsque vous avez le moindre doute, un problème de compréhension voire un mauvais fonctionnement par rapport à ce que vous avez lu, à consulter le manuel en Anglais ou en Allemand, tous deux étant téléchargeables sur le site de Conitec et à me faire part de vos remarques. Et je réclame par avance votre indulgence.

Je vous souhaite le plus grand plaisir à lire ces lignes et à mettre en pratique ce que vous apprendrez.

Alain Brégeon
mailto:alainbregeon@hotmail.com

## Avant propos de la part de l'auteur

Cher Lecteur,

Bienvenue sur le premier "Mini Atelier". Tandis que les ateliers "normaux" couvrent de grands sujets (c'est-à-dire l'essentiel des jeux d'aventures ou de rôles ou des simulateurs de vol) ces ateliers sont conçus pour couvrir des sujets plus petits que vous pouvez ajouter à un projet existant. Notre premier sujet sera "Les éclats de lentilles".

Cet atelier, comme les ateliers normaux, est écrit pour des utilisateurs ayant déjà un peu d'expérience sur 3DGameStudio. Je suppose que vous avez travaillé les différents tutoriaux et que vous comprenez au moins l'essentiel sur la façon d'employer GameStudio et WDL.

Ce texte vient en complément du reste de la documentation 3DGameStudio, et ne la remplace pas. Si quelque chose dans cet atelier est peu clair, parcourez s'il vous plaît les manuels qui sont fournis avec 3DGameStudio. Je fais d'avance des excuses de n'importe quelle formulation peu claire, code défectueux, erreurs, ou omissions.

J'espère que vous trouvez ces nouveaux mini ateliers informatifs et agréables.

- Doug..

# Table des matières

# Les premières choses à faire

Avant que vous ne commenciez, il est très important de s'assurer que vous avez la dernière version de 3D GameStudio (moteur, éditeurs et scénarios de Template). J'essaye de tirer profit de la dernière version ainsi certaines commandes / dispositifs que j'utiliserai sont seulement disponibles avec les dernières mises à jour.

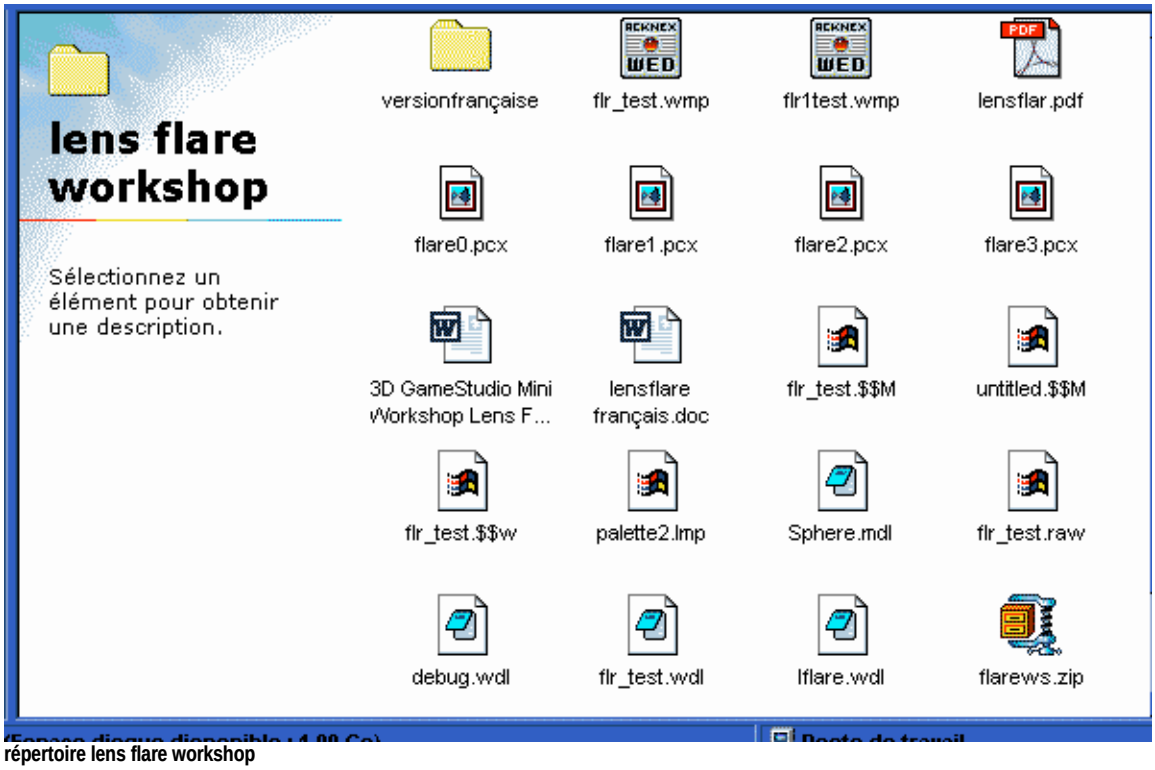
J'essayerai également de tirer profit des dispositifs A5 aussi souvent que possible. Quand j'utilise un dispositif spécifique A5 je le noterai et suggérerai un " travail autour de " (s'il est disponible). Cet atelier a été fait avec la version A5 5.05 ; si vous l'essayez avec des moteurs de version inférieure, c'est à vos propres risques.

## Préparez votre zone de travail

Crée un dossier appelé "lens flare workshop" dans votre dossier GSTUDIO. C'est le dossier où tout vos éléments de jeu seront stockés. Décompressez le contenu de flarews.zip dans ce dossier.

Votre dossier doit maintenant contenir les fichiers suivants :

Lensflaref.pdf (ce document)
flare0.pcx
flare1.pcx
flare2.pcx
flare3.pcx
flr_test.wmp
flr_test.wdl l
flare.wdl



répertoire lens flare workshop

le fichier **lflare.wdl** contient la source entière pour le code d'éclats de lentille que nous allons couvrir dans cet atelier. Si vous voulez voir ce que donne ce code, là, maintenant, tout de suite, alors compilez et exécutez le niveau **flr_test** de WED. Déplacer vous dans le niveau et observez lorsque l'éclat de lentille est visible et comment il se déplace.

## Création du niveau

Parce que c'est un mini atelier je ne vais pas vous faire perdre votre temps en vous donnant les étapes de construction d'un niveau simple. Le code que nous allons créer doit travailler avec n'importe quels niveaux extérieurs. Si vous avez besoin d'un niveau d'essai pour commencer (ou pour la mise au point) j'en ai inclus deux avec cet atelier (**flr_tst.wmp** et **flr1test.wmp**).

Notez que vous pouvez mettre la position du soleil dans le niveau (par les propriétés de carte - **Map properties** -, pour le rendu de la lumière du soleil et des ombres), aussi bien que dans WDL (**sun_angle**). Cependant ces deux positions ne sont pas nécessairement les mêmes! Pour que la lumière du soleil, les ombres et les éclats de lentille donnent parfaitement dans votre niveau, faîtes attention à mettre votre **sun_angle.pan** et **sun_angle.tilt** au même azimut du soleil et angles de déviation que ceux que vous avez entrés dans WED.

Le monde de test

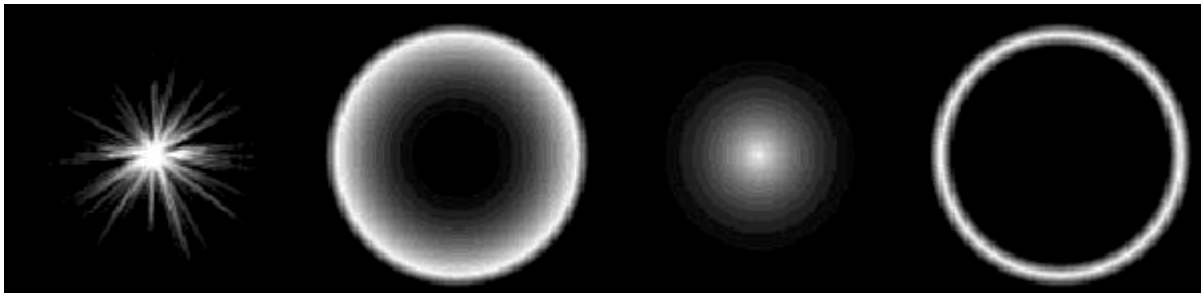
La chose importante à noter en employant votre propre niveau est qu'il doit y avoir au moins un endroit où vous pouvez tracer une ligne de la caméra à la valeur de position stockée dans **flare_sun_pos** (nous parlerons de cette valeur plus tard) ou l'effet de lentille ne se montrera jamais.

# Qu’est-ce qu’un effet de lentille ?

Le terme d’effet de lentille de a été employé par différentes personnes pour dire beaucoup de choses différentes. Une bonne définition vient du livre "The Art and Science of Digital Compositing" (Ron Brinkmann, ISBN : 0121339602)

*Un artefact d'une lumière brillante brillant directement dans l'assemblage de lentille d'une caméra.*

Ceux-ci peuvent prendre la forme de halos, de pointes, ou de taches brillantes ou des halos, qui la semble flotter dans l'espace. L'intensité et la position de ces artefacts dépendent de l'assemblage des lentilles et de la position et de l'intensité de la lumière dans cette relation.



Quelques éclats de lentille

À moins que vous ne portiez des verres vraiment épais ou dépensiez beaucoup de temps avec une caméra ou d'autres dispositifs optiques, vous ne normalement verrez pas d’effet de lentille dans votre vie quotidienne. Ils apparaissent de temps en temps dans des films et la télévision ("X Files" est une grande source pour les éclats de lentille). Parfois ils apparaissent par accident mais souvent ils sont employés pour des buts spectaculaires (quoique de plus en plus des éclats de lentille spectaculaires sont ajoutés en post-production par des ordinateurs).

Aussi quelle idée de vouloir employer des éclats de lentille dans votre jeu ? Peut-être pour essayer de donner une sensation de film à votre jeu, donner le sentiment que le joueur est derrière une feuille de verre, ou vous pensez juste d’avoir des effets 'propres'. Employé correctement un effet de lentille peut ajouter un plus à votre niveau. Imaginez le joueur débusquant un tireur isolé à cause d'un éclat de lentilles sur sa lunette de visée.

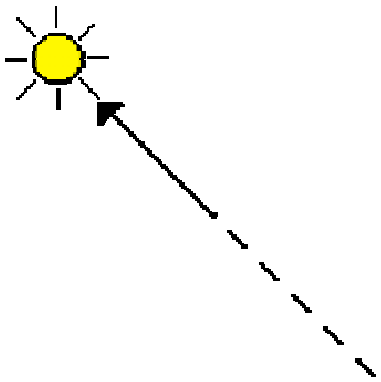
Aussi propre que l'effet puisse être prenez garde de ne pas abuser des éclats de lentille! Ce n'est pas parce que les éclats de lentille sont faciles à ajouter qu'ils faut en abuser.

## Comment nos éclats de lentille fonctionnent

Comme vous pouvez le voir il y a de nombreux types différents d'éclats de lentille. Le type le plus spectaculaire d'éclat de lentille et celui qui est le plus employé dans les films, la télévision et des jeux est l'effet de tache/halo multiple. Il est obtenu en tirant directement dans une source lumineuse brillante comme le soleil et est le résultat de réflexions des surfaces des lentilles et du corps de la caméra.

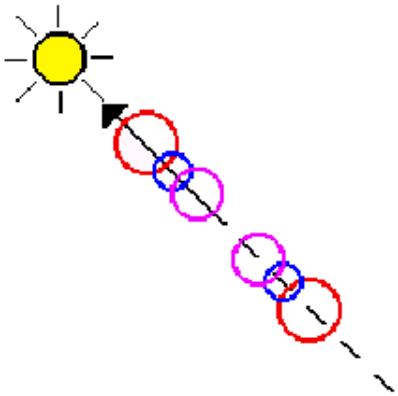
Nous pourrions essayer avec un vrai modèle la lentille et une lumière pour arriver à un quasi effet réaliste mais cela prendrait beaucoup de temps et nous ne pourrions pas le reproduire en temps réel. L'approche que nous allons employer est beaucoup plus simple et se comporte presque de même.

Il y a plusieurs façons différentes de truquer un éclat mais la méthode que je vais employer est l'approche "du bâton et du sprite". Calculez un vecteur (le bâton) avec son origine au centre de la caméra et se dirigeant vers le soleil.



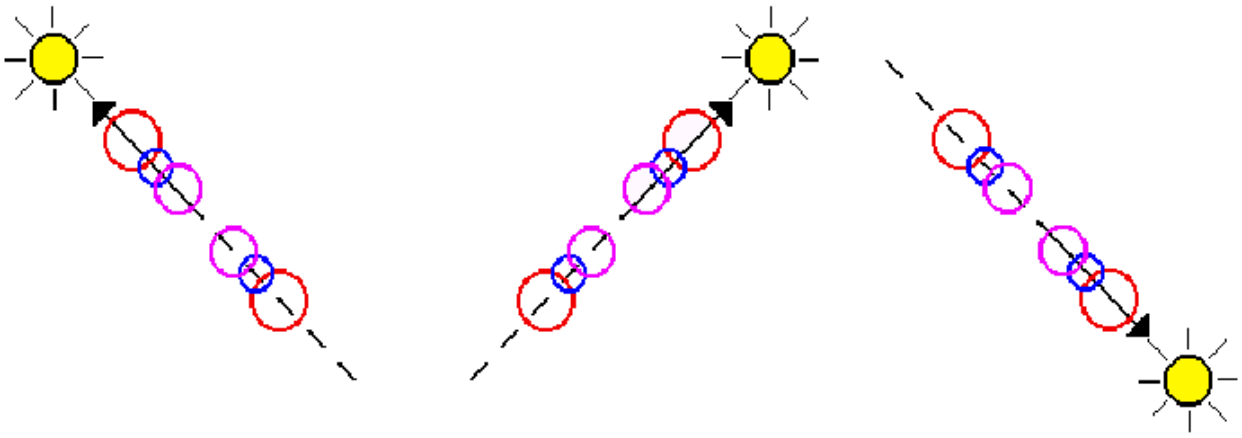
**Bâton pointant vers le soleil**

Prenez une série d'images d'éclat (les sprites) et arrangez les sur ce vecteur pour que la moitié d'entre elles soit vers l'avant de l'origine (placée le long du côté positif du vecteur) et l'autre moitié soit symétriquement reflétée vers l'arrière (la partie négative du vecteur).



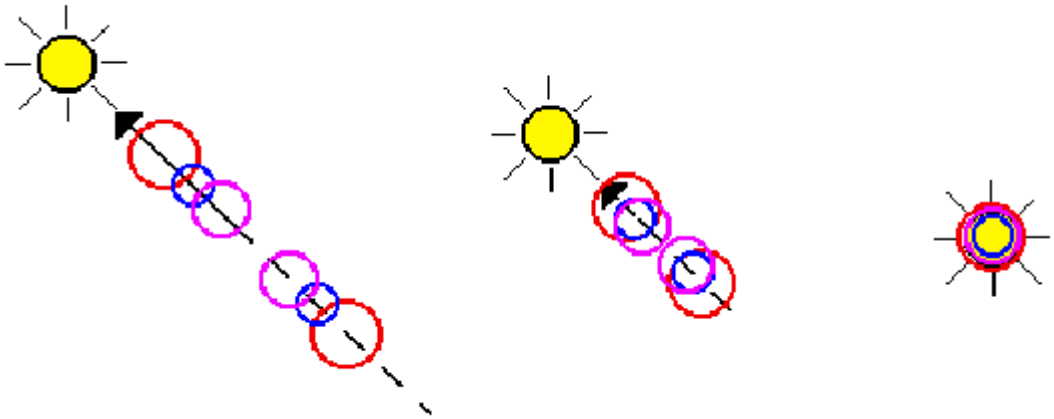
**Les sprites sur le Bâton**

Comme la caméra se déplace par rapport au soleil le vecteur relatif au soleil tournera autour du centre de l'écran comme une main sur une horloge.



**Suivez le Soleil**

Lorsque le soleil s'approche du milieu de l'écran, le vecteur sera plus court et les images d'éclat se resserreront. Quand la caméra sera directement sur le soleil les images seront dessinées les unes sur les autres.



**Les éclats se resserrent**

le résultat est un effet d'éclat de lentille assez convaincant avec très peu d'effort. En ajustant les images du sprite d'éclat et leur espacement nous pouvons produire plusieurs effets uniques en employant le même code.

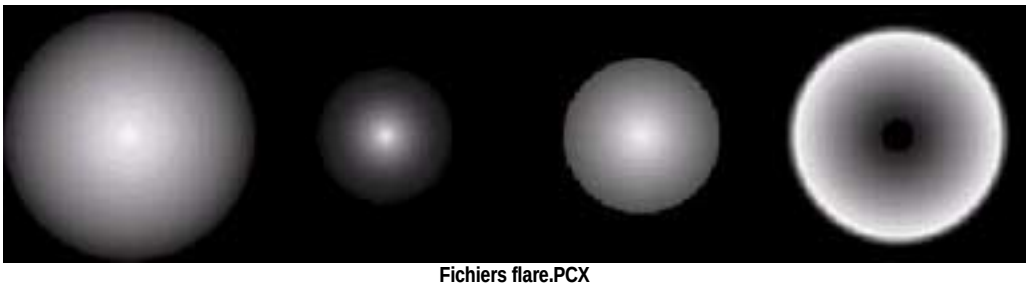
# Scénario des éclats de lentille

Et si nous commencions? Demandons-nous d'abord comment nous allons montrer les images d'éclat projetées. Vous pouvez le faire de plusieurs façons. Ma première version de ce code a employé les panneaux 2D qui ont bien travaillé mais j'ai constaté que l'utilisation du scénario des entités définies m'a donné plus de flexibilité.

## Les entités d'éclat

Le Scénario défini des entités qui se comportent un peu différemment des entités créées dans WED ou par la commande **create()** par laquelle ils existent "à l'extérieur" du niveau. Cela vous donne une flexibilité supplémentaire pour mettre la couche et la caméra dans laquelle ils apparaîtront.

Vous devez avoir quatre fichiers flare.pcx inclus avec ce dossier d'atelier. Vous pouvez employer n'importe quelle collection d'entités pour représenter vos éclats, mais les spots et les halos semble le mieux pour travailler (note : quelques éclats d'objectif d'une caméra réelles sont de forme hexagonale parce qu'ils sont causés par le reflet de lumière sur les lames d'ouverture). Voici les images que nous employons :



**Fichiers flare.PCX**

## Création du scénario

Nous voulons concevoir ce code de telle façon que nous puissions l'ajouter à plusieurs projets différents avec une quantité minimale de modification. Dans ce but nous allons créer toutes nos fonctions, vars et entités dans un fichier séparé qui pourra être inclus facilement dans n'importe lequel de nos projets.

Nous commencerons en créant un nouveau fichier texte appelé lflare.wdl. Vous pouvez créer ce fichier dans votre dossier de projet ou vous pouvez créer un nouveau dossier que vous emploierez pour stocker vos propres fichiers de calibre définis pour vous.

La première chose que nous allons ajouter est un bloc de commentaire au début de notre code afin que si nous regardons ce code dans quelques mois nous saurons ce que font les fonctions dans ce fichier.

```
////////////////////////////////////////////////////////////////
// Fichier: lflare.wdl
// Code WDL pour les éclats de lentilles et autres effets lumineux
////////////////////////////////////////////////////////////////
```

Ensuite nous allons définir un point pour stocker la position de la lumière causant l'effet d'éclat. Nous appellerons cette position **flare_sun_pos**. Ce point détermine quand et où les effets d'éclat apparaissent. Nous mettrons cette valeur dans la fonction **lensflare_create(**).

```
var flare_sun_pos[3];   // position du soleil dans le ciel
```

Le var suivant sera employé pour mettre le **trace_mode** quand nous traçons de la CAMÉRA au soleil.

```
var flare_trace_mode; // utilisé pour tracer vers le soleil
```

Chaque entité sprite d'éclat doit contenir une valeur qui détermine où elle sera placée le long du vecteur (le bâton). Nous redéfinirons la variable de peau pour les entités sprite d'éclat afin qu'elle serve à stocker la position de l'image (en pourcentage de la distance qu'il y a entre le centre de l'écran et le soleil). (nous mettrons les valeurs individuelles quand nous définirons les sprites plus tard).

```
 // utilise le paramètre  peau pour stocker la position en % du point de pivot d'un sprite entité
define pivot_dist,skin;
```

Ensuite nous définirons une valeur de statut qui sera employée par nos fonctions d'éclat de lentille pour allumer et éteindre l'effet et voir dans quel état il est.

```
var qlensflare = -1;    // -1 == n'est pas créé
                        // 0 == éteint
                        // 1 == allumé
                        // autrement == on éteint
```

Maintenant créons les entités sprite qui composeront l'effet d'éclat. Il y a huit d'entre elles (**flare0_ent** à **flare7_ent**) plus un éclat spécial qui sera employé pour le soleil lui-même.

```
// C'est le soleil lui-même
entity flaresun_ent
{
	type = <flare2.pcx>; // effet soleil
	view = camera; // comme les paramètres de la caméra de la vue par défaut
	layer = -6; // affiche en dessous des autres couches d'entités
	pivot_dist = 1; // pourcentage du point de pivot au soleil (1 == sur le soleil)
	scale_x = 2; // le soleil est 2 fois plus grand que les éclats
	scale_y = 2;
}
```

Notez que la valeur de **pivot_dist** est 1 (100 %) cela le placera directement sur la position du soleil. Pour les autres sprites nous emploierons des valeurs entre 0 et 1 pour 'le côté avant' des éclats de lentilles et entre 0 et-1 pour 'le côté arrière'.

```
// les 8 entités d'éclats de lentille
entity flare0_ent
{
	type = <flare0.pcx>;
	view = camera;
	layer = -6;
	pivot_dist = 0.75; // à la distance 0 se trouve le point de pivot –le centre de l'écran
}

entity flare1_ent
{	// 7 lens reflections
	type = <flare1.pcx>;
	view = camera;
	layer = -6;
	pivot_dist = 0.55;
}

entity flare2_ent
{
	type = <flare2.pcx>;
	view = camera;
	layer = -6;
	pivot_dist = 0.35;
}

entity flare3_ent
```

```
{
	type = <flare3.pcx>;
	layer = -6;
	view = camera;
	pivot_dist = 0.15;
}

entity flare4_ent
{
	type = <flare0.pcx>;
	layer = -6;
	view = camera;
	pivot_dist = -0.25;
}

entity flare5_ent
{
	type = <flare1.pcx>;
	layer = -6;
	view = camera;
	pivot_dist = -0.45;
}

entity flare6_ent
{
	type = <flare2.pcx>;
	layer = -6;
	view = camera;
	pivot_dist = -0.65;
}

entity flare7_ent
{
	type = <flare3.pcx>;
	layer = -6;
	view = camera;
	pivot_dist = -0.85;
}
```

Les valeurs définissent ici l'effet d'un type d'éclat, un effet où les images sont répétées pour moitié devant et pour moitié derrière le centre de l'écran (0,1,2,3*0,1,2,3). En changeant les images, les échelles et les valeurs de **pivot_dist** de chacune des entités vous pouvez faire votre propre effets d'éclat.

Maintenant nous allons écrire les fonctions qui contrôlent et animent ces sprites d'éclats de lentille. Elles seront connues comme "fonctions d'interface" puisqu'elles sont appelées à l'extérieur de notre scénario (c'est-à-dire du scénario principal). Les trois fonctions d'interface dont nous avons besoin sont **create** pour créer l'effet d'éclat de lentille, **start** pour commencer à montrer l'effet et **stop** pour arrêter l'effet. Chacune de ces fonctions commencera par le préfixe de **lensflare_** donc nos fonctions seront appelées **lensflare_create()**, **lensflare_start()** et **lensflare_stop()**.

Nous écrirons aussi des fonctions d'aide qui seront appelées de nos fonctions d'interface. Ces fonctions seront employées pour initialiser, montrer/cacher et placer nos sprites d'éclat. Nous donnerons à chacune de ces fonctions d'aide un préfixe de **flare_**.

Notre première fonction d'aide **flare_init** initialise les valeurs de chacune de nos entités d'éclat, mettant leur canal alpha si nous sommes dans le mode D3D (transparence si nous sommes dans le mode logiciel). Cette fonction est appelée de **lensflare_create()**.

```
// Desc: cette fonction prend une entité comme paramètre.
function flare_init(flare_ent)
{
	my = flare_ent; // nécessaire parce que les paramètres de fonction n'ont pas de type
	my.visible = off; // démarre avec les effets à 0 (off)
	if (video_depth > 8) // D3D mode?
	{
		ent_alphaset(0,10);//crée le canal alpha (ne travaillera pas avec l'édition standard)
		my.bright = on;
```

```
		my.flare = on;
	}
	else
	{
		my.transparent = on; // affichage infect en 8 bit, quoique
	}
}
```

Cette fonction prend une entité comme paramètre (par exemple **flare_init(flare0_ent)**;) . Si vous n'avez pas employé souvent de paramètres de fonction vous pourriez trouver cela déroutant. Le manuel WDL dit que nous pouvons seulement passer de simple nombre comme paramètre, mais dans ce cas auquel nous vous montrons que nous passons une entité entière. Le Manuel WDL déclare aussi que "le paramètre original dans la fonction appelante reste inchangé" mais nous employons cette fonction pour changer des valeurs. La raison pour laquelle cela fonctionne quand même est que chaque entité est identifiée dans le moteur par un nombre simple (le numéro d'identification de l'entité). On peut passer ce nombre comme un paramètre de fonction au même titre qu'un autre nombre. Pour retourner ce numéro dans une entité nous l'assignons au synonyme My. Maintenant nous pouvons employer My pour initialiser toutes nos valeurs.

La fonction d'aide suivante prend aussi une entité d'éclat comme paramètre. Cette fois nous rendons l'éclat visible et le plaçons le long du vecteur du centre d'écran au soleil en employant la valeur en % stockée dans la variable **pivot_dist**. La position écran du soleil est stockée dans **temp.x** et **temp.y**, qui est calculée dans la fonction **lensflare_start()**. Parce que le sprite est projeté de l'écran aux coordonnées du monde en employant la commande **rel_for_screen()** la distance d'où apparaissent les éclats de lentille de l'écran (**my.z**) donnera la taille de l'éclat. C'est une autre valeur que vous pouvez adapter pour changer l'effet.

```
// Desc: place un éclat à la déviation du centre de l'écran temp.x/temp.y
function flare_place(flare_ent)
{
	my = flare_ent;
	my.visible = on;
	// multiplie le pixel de la déviation par le facteur du pivot,
	// et ajoute le centre de l'écran
	my.x = temp.x*my.pivot_dist + 0.5*screen_size.x;
	my.y = temp.y*my.pivot_dist + 0.5*screen_size.y;
	my.z = 750; // distance de l'écran, determine la taille de l'éclat
	rel_for_screen(my.x,camera);
}
```

Notre dernière fonction d'aide est simplement employée pour allumer ou éteindre (rendre visibles ou cachés) tous nos sprite d'éclat. Cette fonction est aussi appelée de **lensflare_start()** et prend un paramètre simple de 1 (on) ou 0 (off).

```
// Desc: cette fonction mets tous les flareN_ent et flareSun_ent on ou
// off dépendant de la valeur passée dans  'on_off'.
function flare_visible(on_off)
{
	flaresun_ent.visible = on_off;
	flare0_ent.visible = on_off;
	flare1_ent.visible = on_off;
	flare2_ent.visible = on_off;
	flare3_ent.visible = on_off;
	flare4_ent.visible = on_off;
	flare5_ent.visible = on_off;
	flare6_ent.visible = on_off;
	flare7_ent.visible = on_off;
}
```

Notre première fonction d'interface, **lensflare_create** crée la position de la source lumineuse et les valeurs d'alpha du sprite d'éclat de lentille.

```
// Desc: setup the lens flare effect
function lensflare_create()
{
```

La première chose que nous ferons est d'utiliser notre fonction d'aide **flare_init()** pour initialiser tous nos sprite d'éclat (incluant le sprite du soleil).

```
// met la valeur alpha pour chaque entité
flare_init(flare0_ent);
flare_init(flare1_ent);
flare_init(flare2_ent);
flare_init(flare3_ent);
flare_init(flare4_ent);
flare_init(flare5_ent);
flare_init(flare6_ent);
flare_init(flare7_ent);

flare_init(flaresun_ent); // l'éclat pour le soleil
```

Ensuite nous emploierons les valeurs du moteur pour **sun_angle** pour mettre le vecteur **flare_sun_pos**. De cette façon les valeurs de l'Azimut du Soleil et des valeurs d'élévation mises dans WED seront employées pour placer notre source lumineuse d'éclats. Vous pouvez remplacer cette section par n'importe quelle valeur qui signifie quelque chose pour votre niveau. Par exemple : un simulateur spatial pourrait avoir le soleil à l'origine du niveau (**0,0,0**). Indépendamment de la valeur que vous donnez à **flare_sun_pos** il est important que vous puissiez tracer jusqu'à ce point de n'importe quelle section du niveau ou alors l'éclat de lentille n'apparaîtra jamais.

```
// met le point du soleil (pour tracer vers ce point et voir si le soleil est visible)
// x = (h*cos(tilt))*sin(pan);
flare_sun_pos.x = (500000 * cos(sun_angle.tilt)) * sin(sun_angle.pan);
// y = (h*cos(tilt))*cos(pan);
flare_sun_pos.y = (500000 * cos(sun_angle.tilt)) * cos(sun_angle.pan);
// z = h*sin(tilt)
flare_sun_pos.z = 500000 * sin(sun_angle.tilt);
```

Ensuite nous mettrons le **flare_trace_mode**. Dans notre démonstration simple vous devez seulement ignorer des blocs passables parce que **flare_sun_pos** se trouve à l'extérieur de la boîte de ciel passable. J'ai aussi ajouté **ignore_models** parce que lorsque vous utilisez un jeu de type première personne, la caméra se trouve à l'intérieur du modèle et ne sera donc jamais capable de tracer vers le soleil.

```
// met le mode trace pour qu'il puisse tracer vers le soleil
//IGNORE_PASSABLE est nécessaire pour que la trace traverse la boîte de ciel
flare_trace_mode = ignore_passable + ignore_models;
```

Nous commencerons par l'effet d'éclat de lentille éteint.

```
	qlensflare = 0; // démarre 'éteint'
}
```

La fonction suivante est le cœur réel du code d'éclat de lentille. Si vous tapez vous même cette fonction avec vos petits doigts, assurez vous d'ajouter cette fonction **après lensflare_create()**.

```
// Desc: démarre et anime un effet de lentille aussi longtemps que qLensFlare == 1
function lensflare_start()
{
```

La première chose que nous vérifions c'est de voir si l'utilisateur a déjà créé les éclats de lentille en appelant **lensflare_create()** directement ou en appelant cette fonction précédemment. Si ce n'est pas fait nous appelons le code **lensflare_create()** maintenant.

```
	if(qlensflare == -1) // crée les effets de lentilles
	{
		lensflare_create();
	}
```

Ensuite nous vérifions pour voir si l'éclat de lentille est déjà actif. Avec ce code on fait en sorte qu'il ne puisse y avoir qu'un seul éclat de lentille qui fonctionne à la fois, et s'il y en a déjà un qui fonctionne, nous retournerons dés ce point.

```
if(qlensflare == 1) // lens flare already started
{
	return;
}
```

Nous donnons au code **lensflare_create()** le temps pour créer et mettre la variable **qlensflare** pour montrer que le code d'éclat de lentille fonctionne.

```
wait(1); // tient compte du temps de préparation pour "lensflare_create"
qlensflare = 1; // marque l'éclat de lentille comme fonctionnant
```

La boucle While principale pour que l'effet puisse fonctionner à chaque cycle d'encadrement avant que quelque chose ne change la valeur de qlensflare à quelque chose d'autre que 1.

```
// place les effets de lentilles
while(qlensflare == 1)
{
// Anime les effets de lentilles
```

Nous emploierons la commande **vec_to_screen()** pour faire un contrôle rapide pour voir si la source de lumière apparaît dans la vue (nous devons en faire une copie d'abord afin de ne pas changer sa valeur). Si le point n'est pas dans la vue de la caméra nous cachons les sprites d'éclat en employant notre fonction d'aide **flare_visible(off)**.

```
vec_set(temp,flare_sun_pos);
if(vec_to_screen(temp,camera) == 0)
{
// en dehors du cône de la vue... enlève les éclats de lentilles
flare_visible(off);
}
else // controle la trace vers le soleil
{
```

Si la source lumineuse passe le test du **vec_to_screen()** nous pouvons vérifier pour voir s'il est visible de la position de CAMÉRA actuelle en employant la commande **trace()**. Comme je l'ai déjà dit auparavant, il est important que **flare_sun_pos** soit placé quelque part pour que la caméra puisse tracer au moins une partie du temps. Dans le cas de la carte test incluse avec cet atelier, un bloc de ciel passable entoure le monde si bien que **flare_sun_pos** étant mis très loin nous pouvons tracer de tous les endroits ensoleillés de la carte. Vous pouvez devoir ajuster la valeur de **flare_sun_pos** ou les paramètres **flare_trace_mode** pour faire ce travail dans votre niveau.

```
// trace vers le soleil
trace_mode = flare_trace_mode;
// contrôle si la ligne vers le soleil est bloquée
if (trace(camera.x,flare_sun_pos) != 0)
{
```

Si quelque chose bloque la CAMÉRA vers soleil nous cachons les sprites d'éclat en employant notre fonction d'aide **flare_visible(off)** comme nous avons fait auparavant.

```
// quelque chose nous bloque…cacher l'éclat de lentille
flare_visible(off);
}
else
{
```

Si nous avons avancé jusqu'à cette section du code jusqu'ici c'est que nous faisons face à la source de lumière du soleil et que rien ne la bloque. Nous emploierons la position d'écran XY du soleil (calculé plus tôt avec l'appel de **vec_to_screen()**) et le décalerons d'une moitié d'écran en largeur et en hauteur pour obtenir la distance XY d'écran réelle du centre d'écran.

```
	// temp contient maintenant la position XY de l'écran soleil
	// soustrait le centre de l'écran, nécessaire pour flare_place()
	temp.x -= 0.5 * screen_size.x;
	temp.y -= 0.5 * screen_size.y;
```

C'est la valeur temporaire employée par la fonction d'aide **flare_place()** pour placer les 8 sprites d'éclat et du soleil.

```
	// place les éclats selon leur déviation et leur pivot distance
	flare_place(flaresun_ent);
	flare_place(flare0_ent);
	flare_place(flare1_ent);
	flare_place(flare2_ent);
	flare_place(flare3_ent);
	flare_place(flare4_ent);
	flare_place(flare5_ent);
	flare_place(flare6_ent);
	flare_place(flare7_ent);
	}
}
```

Attendre un cycle d'encadrement avant de faire cela encore une fois.

```
	wait(1);	// anime chaque cycle
}
```

Nous atteindrons cette section de code seulement si quelque chose change **qlensflare** à une valeur autre que 1. Dans ce cas nous devons nous assurer que les éclats de lentille sont cachés (en appelant la fonction **flare_visible()**) et mettre **qLensFlare** à la valeur 0 (off) pour signaler que c'est fait.

```
	// enlève les éclats de lentilles
	flare_visible(off);
	qlensflare = 0; // marque les éclats de lentille comme éteint
}
```

Cette dernière fonction est la plus simple. Pour éteindre tous les éclats de lentille tout ce que nous devons faire est de mettre la valeur de qlenflare à une valeur autre que 1. En mettant la valeur à .5 ici la boucle while principale dans **lensflare_start()** sortira.

```
	// Desc: arrête les éclats de lentille
	function lensflare_stop()
	{
	qlensflare = .5; // signal pour arrêter
	}
```

## Comment employer ce code

Maintenant que nous terminé ce code c'est vraiment très simple de l'employer pour ajouter des éclats de lentille à n'importe quel niveau. Assurez-vous que **lflare.wdl** est dans votre chemin et incluez-le (**include**) ensuite au début de votre scénario principal (directement après l'inclusion des autres calibres). Pour commencer l'effet d'éclat de lentille appelez simplement **lensflare_start();** et appelez **lensflare_stop();** pour arrêter l'effet. Si vous projetez de commencer l'effet ailleurs qu'au début du jeu vous pourriez vouloir appeler **lensflare_create** dans votre fonction principale pour éviter des ralentissements dans le jeu causés en mettant les valeurs alpha des sprites d'éclat.

```
////////////////////////////////////////////////////////////////////////
// Niveau de test des éclats de lentille
////////////////////////////////////////////////////////////////////////
include <lflare.wdl>;// notre code éclats

////////////////////////////////////////////////////////////////////////
// le moteur démarre dans la résolution donnée par les variables suivantes.
var video_mode = 6;
var video_depth = 16;
////////////////////////////////////////////////////////////////////////
// La fonction principale est appelée au début du jeu
function main()
{
	load_level(<flr_test.wmb>);
	_move_straight(); // mouvement de caméra prédéfini
	sky_clip = -65;
	lensflare_create(); // initialise les éclats de lentilles
	lensflare_start(); // démarre les éclats de lentille
	wait(2);
	// démarre avec les éclats dans la frame
	camera.pan = 25;
	camera.tilt = 15;
}

on_2 = lensflare_start;
on_3 = lensflare_stop;
```

## Conclusion

j'espère que vous avez aimé le premier mini atelier. Si vous avez des questions, commentaires, ou des améliorations pour cela ou un autre atelier postez-les s'il vous plaît au Forum d'Utilisateur Conitec.


**Exécution de notre niveau terminé**