Tutoriel Camera

Ou : Comment faire pour que ces vues fonctionnent ?

Par Gnometech

Traduction Alain Brégeon

NDT : Nous parlons de vue 1ère personne et 3ème personne.

La vue 1ère personne signifie que vous êtes le joueur, et que la caméra se trouve dans votre tête (ou à la place de vos yeux). Elle montre donc ce que vous verriez si vous êtiez dans le jeu et on ne voit pas le joueur.

La vue 3ère personne signifie que c'est une 3ème personne qui a la caméra dans sa tête (ou à la place de ses yeux). Cette personne vous suit, on ne la voit pas mais on voit donc le joueur de dos.

Chapitre 1: Créons la carte et préparons la session

Mon nom est Gnometech et voici mon tutoriel sur les caméras. Je ne vous ferai pas la promesse de vous dire que vous serez définitivement capable de créer vos propres vues de caméras après avoir travaillé ce tutoriel mais il y a des chances que les choses soient un peu plus claires après ceci.

Démarrez en créant une petite carte ou utilisez une de vos cartes exemples. Je suppose que vous savez le faire et je ne donnerais donc pas d'explication ici. Placez votre modèle joueur n'importe où dans le niveau et assignez lui votre action de mouvement favorite comme l'action prédéfinie player_move ou celle que vous avez écrite vous même.

Pour que les caméras fonctionnent il est indispensable que l'entité synonyme "player" soit définie et connue. C'est le calibre movement.wdl qui l'assigne aussi si vous avez écrit votre propre script de mouvement assurez-vous de faire de même.

Puisque vous voulez appeler votre propre code de caméra, soyez sûr de changer la ligne où vous appelez l'actuel "move_view" par le nouveau que nous appellerons "update_views". Encore une fois, si vous n'employez pas les scripts prédéfinis, peu importe d'où vous appelez cette routine, assurez-vous simplement qu'elle soit appelée à chaque cycle d'affichage.

Créez maintenant un nouveau fichier de WDL appelé "cam.wdl" et incluez-le dans votre script principal. Si vous êtes prêt, on y va!

Chapitre 2: La vue 1ère personne

Celle-ci est tout à fait simple. Mettez les lignes de programmation suivantes dans le script WDL:

```
view 1st_person //vue 1ère personne
	{
		layer = 1;
		pos_x = 0;
		pos_y = 0;
	}

	function init_cameras()
	{
		camera.visible = off;
		1st_person.size_x = screen_size.x;
		1st_person.size_y = screen_size.y;
		1st_person.genius = player;
		1st_person.visible = on;
	}

	function update_views()
	{
		1st_person.x = player.x;
		1st_person.y = player.y;
		1st_person.z = player.z;
		1st_person.pan = player.pan;
		1st_person.roll = player.roll;
		1st_person.tilt = player.tilt;
	}
```

Maintenant, incluez l'appel "init_cameras()" quelque part dans votre fonction principale, après le chargment du niveau. Sauvegardez votre script et exécutez votre niveau. Si vous avez fait tout correctement, la caméra devrait suivre les mouvements du joueur.

Qu'avons-nous fait ici? Bien, nous avons défini une nouvelle vue. Imaginez une vue qui serait votre raccordement au monde de jeu. C'est une "fenêtre", qui permet celà. Vous pouvez définir la position et la taille de la fenêtre sur l'écran en utilisant les variables pos_x, pos_y, size_x et size_y. Comme vous pouvez voir, le coin gauche supérieur de notre vue est égal au coin gauche supérieur de l'écran et la taille est exactement la taille d'écran.

La fonction update_views est mise à jour à chaque cycle d'affichage et place de nouvelles valeurs de x, y et z de la caméra (la vue) dans le monde. Ici, nous les plaçons juste aux coordonnées du joueur, de ce fait la caméra est toujours "à l'intérieur" du joueur.
Le script n'est pas encore parfait. Par exemple, la caméraest placée au centre du modèle du joueur et pas dans sa tête, ce qui fait que la vue est un peu près du plancher. Et nous ne pouvons pas regarder vers le haut ou vers le bas pour l'instant.

La première chose est rapidement faisable. Nous définissons juste une variable au début du nouveau script :

```
var eye_height = 20;
```

Et puis, au lieu de

```
1st_person.z = player.z;
```

nous écrivons

```
1st_person.z = player.z + eye_height;
```

ce qui place la caméra plus haute de 20 quants. Si la valeur n'est pas adaptée à vos besoins, ajustez-la, si nécessaire même pendant le jeu.

Maintenant, nous voulons que le joueur puisse regarder vers le haut ou vers le bas. Pour cela nous avons besoin d'autres variables :

```
var tilt_1st = 0;
var cam_turnspeed = 2;
var max_tilt_1st = 40;
```

Maintenant changez la ligne suivante :

```
1st_person.tilt = player.tilt;
```

par

```
1st_person.tilt = player.tilt + tilt_1st;
```

Puis ajoutez les fonctions :

```
function look_up()
{
	if (tilt_1st < max_tilt_1st) { tilt_1st += cam_turnspeed; }
}

function look_down()
{
	if (tilt_1st > -max_tilt_1st) { tilt_1st -= cam_turnspeed; }
}
```

à votre script. Ces fonctions doivent être appelées pendant le jeu. Par exemple, you pouvez définir:

```
on_pgup = look_up;//appui sur la touche page suivante
on_pgdn = look_down;//appui sur la touche page précédente;
```

Sauvegardez votre niveau et exécutez votre niveau. Hmmm... pas mal, mais ce n'est pas l'effet désiré. Si vous pressez une des touches et là maintenez enfoncée, la vue change un petit peu mais pas plus. En fait vous devez appuyer plusieurs fois pour que la vue change et encore tout cela est très lent. Comment régler ce problème ? Changeons les définitions de on_pgup et on_pgdn par:

```
on_pgup = handle_pageup;
on_pgdn = handle_pagedown;
```

Puis nous définissons 2 fonctions supplémentaires:

```
function handle_pageup()
{
	while (key_pgup)
	{
		look_up();
		wait(1);
	}
}
function handle_pagedown()
{
	while (key_pgdn)
	{
		look_down();
		wait(1);
	}
}
```

De cette façon les fonctions seront appelées à chaque cycle d'affichage aussi lontemps que la touche sera pressée. Si le mouvement de la caméra est trop rapide ou trop lent pour vous, ajustez la valeur cam_turnspeed. Si vous trouvez également que son angle de vue est trop limité, vous pouvez modifier la valeur de max_tilt_1st.
Avec une valeur de 90 il peut regarder directement vers le haut et vers le bas.

Nous en avons terminé avec ce chapitre. La vue 1ère personne fonctionne correctement, nous allons passer à quelque chose de différent, la vue 3ème personne.

Chapitre 3: Vue 3ème personne en rotation libre

A présent nous voulons créer une vue 3ème personne en rotation libre. Il y a deux possibilités pour une vue 3ème personne :

La première qui consiste à créer une caméra à l'extérieur du joueur et qui NE tourne PAS même lorsqu'elle lui fait face à partir d'une certaine direction.

La seconde serait une caméra qui tournerait lorsque le joueur tournerait, de ce fait qui serait toujours derrière lui (ou devant lui, ou sur son côté ou.) et je rendrai cette caméra libre en rotation et libre en zoom. Si vous avez juste besoin d'une vue de dessus pour votre jeu ou une vue de côté, pas de problème non plus, il suffit de choisir les bonnes valeurs (vous verrez lesquelles) et d'inclure les fonctions qui les changent. ;-)

On y va. Pour commencer nous allons créer une vue 3ème personne qui fait face au joueur, toujours dans la même direction et qui vous dira ce qu'il faut changer lorsque vous souhaitez que la vue tourne avec le joueur.

Définissons une seconde vue maintenant:

```
view 3rd_person
{
	layer = 1;
	pos_x = 0;
	pos_y = 0;
}
```

Ajoutez les lignes suivantes quelque part à l'intérieur de la fonction "init_cameras":

```
...
3rd_person.size_x = screen_size.x;
3rd_person.size_y = screen_size.y;....
```

Nous ne la rendons pas visible pour l'instant. Au lieu de cela nous allons faire en sorte de pouvoir basculer les 2 vues pendant le jeu. Définissons cette routine de basculement maintenant :

```
var cam_mode = 0; // 0 indique une vue 1ère personne, 1 indique une vue 3ème personne
function toggle_cams()
{
	if (cam_mode == 0)
	{ // Change en vue 3ème personne
		1st_person.visible = off;
		3rd_person.visible = on;
		cam_mode = 1;
	}
	else
	{ // Change en vue 1ère personne
		3rd_person.visible = off;
		1st_person.visible = on;
		cam_mode = 0;
	}
}
on_f8 = toggle_cams;// en pressant F8 vous basculez la vue
```

En pressant la touche F8 à partir de maintenant, nous sommes capables de basculer entre les 2 vues. Ce qui ne présente aucune utilité vue que nous n'avons défini nulle part les paramètres x, y et z de la 3ème personne ! Nous le faisons donc en modifiant la fonction "update_views". (Souvenez-vous qu'elle est appelée à chaque cycle d'affichage.)

Mais avant tout, nous devons réfléchir ... comment pouvons-nous définir une caméra 3ème personne qui puisse tourner librement autour du joueur? Elle restera (du moins pour l'instant) sur le même plan, aussi sa valeur Z ne changera pas. Mais comment calculer avec précision les positions x et y ? Nous avons besoin d'un peu de mathématiques pour cela ! Imaginez le joueur vu d'en haut. La meilleure chose serait de prendre un morceau de papier et d'y mettre un point dessus, quelque part, pour indiquer la position du joueur vue d'en haut. Tracez un cercle autour de ce point. Ceci devrait être le cercle sur lequel la caméra peut être déplacée autour du joueur. Ce cercle a un certain rayon. Si vous décrétez qu'un point quelconque de ce cercle à la valeur "0" (par exemple le haut, mais c'est sans importance) vous pouvez définir n'importe quel autre point "P" sur ce cercle simplement par un arc entre la ligne qui va du point "0" au centre et du point "P" vers le centre.

Ainsi, en définissant la distance à partir du joueur (le rayon du cercle) et de l'angle dans le plan, la position de la caméra est déterminée. Rassurez-vous, je vous donnerai la formule, mais vous pourriez l'obtenir facilement vous-même avec des petites maths:

```
var dist_planar = 300; // distance depuis le joueur
var cam_angle = 0;
function update_views()
{
	if (cam_mode == 0)
	{
		1st_person.x = player.x;
		1st_person.y = player.y;
		1st_person.z = player.z + eye_height;
		1st_person.pan = player.pan;
		1st_person.roll = player.roll;
		1st_person.tilt = player.tilt + tilt_1st;
```

```
        }
        else
        {
                3rd_person.x = player.x - cos (cam_angle) * dist_planar;
                3rd_person.y = player.y - sin (cam_angle) * dist_planar;
                3rd_person.z = player.z;
                3rd_person.pan = cam_angle;.3rd_person.roll = 0;
                3rd_person.tilt = 0;
        }
}
```

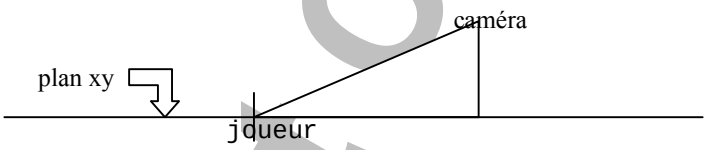
Sauvegardez le script exécutez le niveau. Pressez F8 pour basculer la vue. Vous pouvez à présent voir votre joueur de l'extérieur. Si vous vous déplacez dans votre niveau la caméra devrait toujours garder la même distance au joueur et ne changera pas même de position quand le joueur tourne.

Si vous changez les valeurs de dist_planar pendant le jeu (par l'intermédiaire de la touche Tab par exemple ou si vous définissez une fonction pour les changer en frappant une touche. .. vous devriez pouvoir créer une telle fonction vous-même maintenant), la caméra fera un zoom avant ou arrière. Si vous changez cam_angle, elle tournera autour du joueur, toujours face à lui. (3rd_person.pan = cam_angle s'assure de cela)

Il y a cependant un problème avec les murs. la caméra passera directement par les murs et le plus souvent, vous aurez des obstacles qui bloqueront votre vison du joueur. Nous règlerons ce problème dans le prochain chapitre, d'abord nous changerons un peu le code de la caméra... nous voulons maintenant pouvoir déplacer la caméra vers le haut, tout en faisant toujours face au joueur. Il devrait également garder la même distance, de ce fait ne se déplaçant pas sur un cercle autour de lui, mais sur une sphère.

Comment pouvons-nous réaliser ceci? Bien, imaginez votre joueur maintenant étant vu de côté. Si la caméra est au-dessus du plan XY du joueur et que vous tracez une ligne d'elle au joueur, vous obtenez un angle entre le plan XY et la ligne. Cet angle définit la position de la caméra précisément.

Naturellement, si vous voulez garder la même distance totale, la distance concernant le plan XY change! Une image pour illustrer ceci:



Maintenant les ajustements de code :

```
var dist_total = 300; // Changz CETTE valeur pour le Zoom avant ou arrière
var tilt_3rd = 0;
function update_views()
{
        ...
else
{
        dist_planar = cos (tilt_3rd) * dist_total;
        3rd_person.x = player.x - cos (cam_angle) * dist_planar;
        3rd_person.y = player.y - sin (cam_angle) * dist_planar;
        3rd_person.z = player.z + sin (tilt_3rd) * dist_total;
        3rd_person.pan = cam_angle;
        3rd_person.roll = 0;
        3rd_person.tilt = - tilt_3rd;
}
```

En changeant les valeurs tilt_3rd, cam_angle et dist_total nous pouvons maintenant déplacer librement la caméra autour du joueur.

Ielle se déplace sur une sphère où le rayon peut être changé en modifiant la valeur dist_total.

Il vous appartient d'écrire des fonctions pour changer ces valeurs pendant le jeu et pour lui assigner des touches. En outre, vous devrez les borner (donner des limites), afin d'empêcher le joueur de faire des zooms trop important ou de trop incliner la caméra.

Dans le prochain chapitre, nous traiterons le problème du mur.

Chapitre 4: Comment empêcher la caméra de passer à travers les murs

Tout d'abord une précision : il y a différentes façons pour faire ceci et je n'ai pas la prétention de dire que la mienne est la meilleure. Cependant elle fonctionne très bien pour moi jusqu'à présent, aussi jetez-y un coup d'œil, elle n'est pas très difficile à comprendre.

L'idée est d'envoyer une trace du joueur à la nouvelle position de la caméra et de voir si un obstacle est percuté sur son chemin. Si c'est le cas, la distance entre le joueur et la caméra sera changée... par la somme d'espace entre le joueur et l'obstacle rendant ainsi la manipulation de la caméra plus lisse. Aucun mouvement saccadé. :)

Regardons dans le détail comment cela est fait. Incluez ces lignes après les calculs concernant la vue 3ème personne (voir le chapitre précédant) :

```
function update_views()
{
	...
	3rd_person.roll = 0;
	3rd_person.tilt = - tilt_3rd;
	validate_view();
	}
	...
}
```

Et nous définissons la fonction:

```
var dist_traced;
function validate_view()
{
	my = player;
	trace_mode = ignore_me + ignore_passable;
	dist_traced = trace (player.x, 3rd_person.x);
	if (dist_traced == 0) { return; } // aucun obstacle de rencontré... parfait
	if (dist_traced < dist_total)
	{
		dist_traced -= 5; // se déplace hors du mur
		dist_planar = cos (tilt_3rd) * dist_traced;
		3rd_person.x = player.x - cos (cam_angle) * dist_planar;
		3rd_person.y = player.y - sin (cam_angle) * dist_planar;
		3rd_person.z = player.z + sin (tilt_3rd) * dist_traced;
	}
}
```

La fonction pour déterminer la position de la caméra est exactement la même, elle utilise juste une nouvelle distance du joueur...dist_traced, obtenue par trace. Elle est un peu modifiée, pour empêcher la camérad'être directement à l'intérieur du mur.

Maintenant la caméra devrait éviter les murs et autres obstacles en vue 3ème personne sans à-coup.

Il reste une chose à faire avant que je vous abandonne... laisser la caméra derrière le joueur en permanence.

C'est très simple à faire, vous devez juste remplacer ces lignes:

```
3rd_person.x = player.x - cos (cam_angle) * dist_planar;
3rd_person.y = player.y - sin (cam_angle) * dist_planar;.
```

Par

```
3rd_person.x = player.x - cos (cam_angle + player.pan) * dist_planar;
3rd_person.y = player.y - sin (cam_angle + player.pan) * dist_planar;
```

Ainsi, tout ce que vous devez faire est d'ajouter la valeur pan du joueur à cam_angle dans le calcul. Soyez sûr de faire la même chose dans validate_view ou vous obtiendrez des erreurs étranges!

Si pour l'instant la caméra n'est pas derrière le joueur au début du jeu, mais lui fait face, ajustez cam_angle jusqu'à ce qu'il convienne à vos besoins... et alors ne laissez plus le joueur changer cette valeur . Ainsi la caméra restera derrière le joueur tout le temps.

Que dire encore? J'espère que ce tutoriel vous a plu et qu'il vous a appris quelque chose. Vous ne devriez plus avoir de problèmes maintenant pour créer de nouvelles vues, les placer dans le niveau, les faire suivre votre joueur, lui faire face etc.

Avec un peu de réflexion vous devriez être capable de créer des caméras stationnaires, des caméras contrôlables par l'utilisateur, des vues de dessus, de côté …

Si vous avez des questions concernant ce tutorial, posez les sur le forum 3DGS ou envoyez moi un eMail à:
gnometech@gmx.de

Gnometech