Chapitre IX : Frame Rate

par Loka

Date de publication : 01/10/2006

Dernière mise à jour : 01/10/2006

IX - Frame Rate

Frame Rate veut dire littéralement taux d'images.

C'est le nombre d'images affichées en une seconde.

Plus le nombre est élevé, plus l'animation est fluide (à priori, la sensibilité et la persistance rétinienne impose une cadence d'au moins 24 images par seconde pour imposer une véritable impression de fluidité).

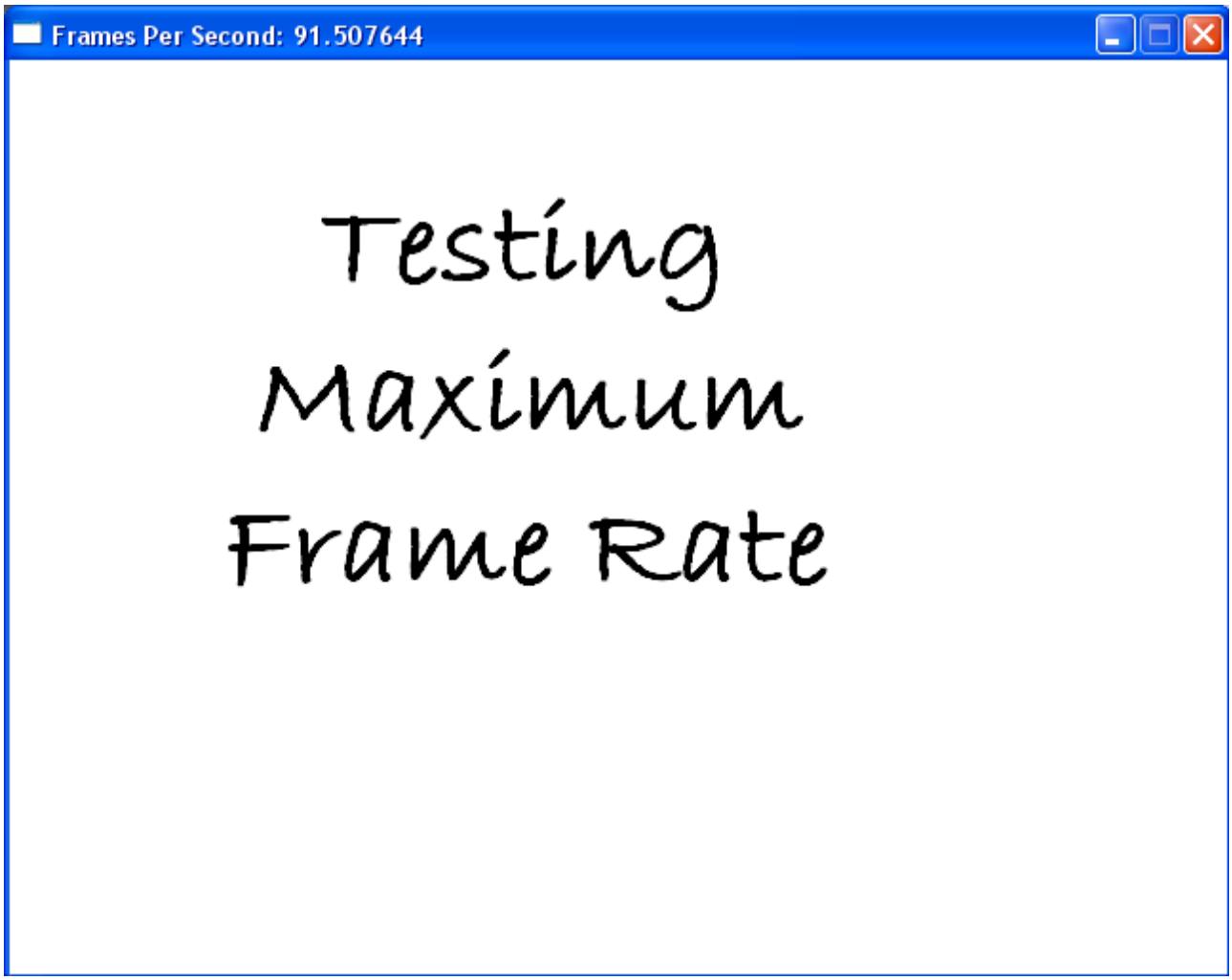
Nous allons voir dans un premier temps comment calculer ce nombre pour ensuite voir comment et pourquoi le réguler dans certains cas.

IX-A - Calculer le nombre de Frames par seconde

Si vous devez savoir à quelle nombre de fps(*frames per second*) votre jeu peut fonctionner, ou si vous vous demandez juste ce que vous pouvez tirer sur un rendement SDL, savoir comment calculer le nombre de frame par seconde est une compétence très utile.

Cette partie du tutoriel vous enseignera comment construire un simple test de calcul de fps.

Nous aurons quelque chose qui ressemble à ça :



Présenté comme dans ce tutoriel, le calcul ne sert pas à grand chose.

Pour le rendre plus intéressant, vous devrez, par vous même, ajouter quelques instructions et autres morceaux de programme que vous souhaitez tester afin de voir comment le nombre de fps évolue et ainsi en tirer les bonnes conclusions.

Nous allons utiliser la classe Timer que nous avons vu dans le tuto précèdent, je ne ferais donc pas de rappels sur cette partie.

Si vous avez des doutes sur le fonctionnement de certaines instructions propres au Timer, reportez vous simplement au chapitre VIII sur les timers.

Commençons donc par la déclaration des variables dont nous aurons besoin :

variables
```
int frame = 0;
```

variables
```
//Timer utilise pour calculer le nombre de frames par seconde
Timer fps;

//Timer utilise pour mettre à jour la barre de caption
Timer update;
```

Voici donc les trois variables clefs de notre test de calcul du nombre de fps.

La variable *frame* garde la trace du nombre de frames qui ont été rendu.

La variable *fps* est le timer qui garde la trace du temps qui a été perdu pendant le rendu.

La variable *update* est le timer que nous utilisons pour mettre à jour la barre de caption, c'est à dire l'endroit où nous afficherons le nombre de frames par seconde.

Après que tout a été initialisé et chargé, il nous faut mettre en route les timers et entrer dans la boucle principale :

Lancement des timers
```
//On demarre le timer update
update.start();

//On demarre le timer fps
fps.start();

//Tant que l'utilisateur n'a pas quitte
while( quit == false ) {
```

Il ne faut pas oublier de lancer les timers.

Par souci de simplicité, nous n'allons nous occuper d'aucun événement à part celui de la fermeture de la fenêtre SDL.

boucle événement
```
//Tant qu'il y a un événement
        while( SDL_PollEvent( &event ) )
        {
            //Si l'utilisateur a clique sur le X de la fenetre
            if( event.type == SDL_QUIT )
            {
                //On quitte le programme
                quit = true;
            }
        }
```

Après cette partie, nous appliquons la surface (le fond), mettons à jour la fenêtre, puis incrémentons le compteur *frame*.

après la boucle événement
```
//On pose le fond
        apply_surface( 0, 0, image, screen );

        //Mise à jour de l'écran
        if( SDL_Flip( screen ) == -1 )
        {
            return 1;
        }

        //On incremente le compteur de frames
```

après la boucle événement
```
            frame++;
```

Comme vu sur l'image de présentation de notre programme, le résultat du calcul du nombre de frames par seconde sera affiché sur la barre caption.

Voici le code correspondant :

calcul frame
```
   //Si une seconde est passee depuis la derniere mise à jour de la barre caption
            if( update.get_ticks() > 1000 )
            {
                //Une chaine de caracteres temporaire
                char caption[ 64 ];

                //On calcule le nombre de frames par seconde et on cree la chaine de caracteres
                sprintf( caption, "Frames Par Seconde: %f", (float)frame / ( fps.get_ticks() / 1000.f )
   );

                //On remet à zero la barre caption
                SDL_WM_SetCaption( caption, NULL );

                //On relance le timer update
                update.start();
            }
```

On souhaite ici mettre à jour le calcul toutes les secondes, on vérifie donc si une seconde s'est passée depuis la dernière mise à jour de la barre caption.

Je vous incite à vous amusez à changer cette valeur (par exemple 100 au lieu de 1000 afin de rafraîchir le calcul tout les dixième de seconde) et de voir le résultat.

Afin d'afficher le résultat du calcul, nous avons besoin d'une chaîne de caractères, dans notre code cette chaîne se nomme *caption*.

Notre calcul est donc ensuite recopié dans cette chaîne.

Le nombre de frames par seconde est calculé en prenant la quantité de frames rendu divisé par le temps mis afin de les rendre (en seconde, c'est pourquoi nous divisons par 1000, le temps étant en millisecondes).

On remet la barre caption à zéro puis nous relançons le timer update afin de permettre à nouveau une vérification correcte.

Voilà qui est fini, vous pouvez maintenant afficher le nombre de frames par seconde sur vos programmes.

Télécharger les sources du IX-A (161 ko)

IX-B - Réguler le nombre de Frames par seconde

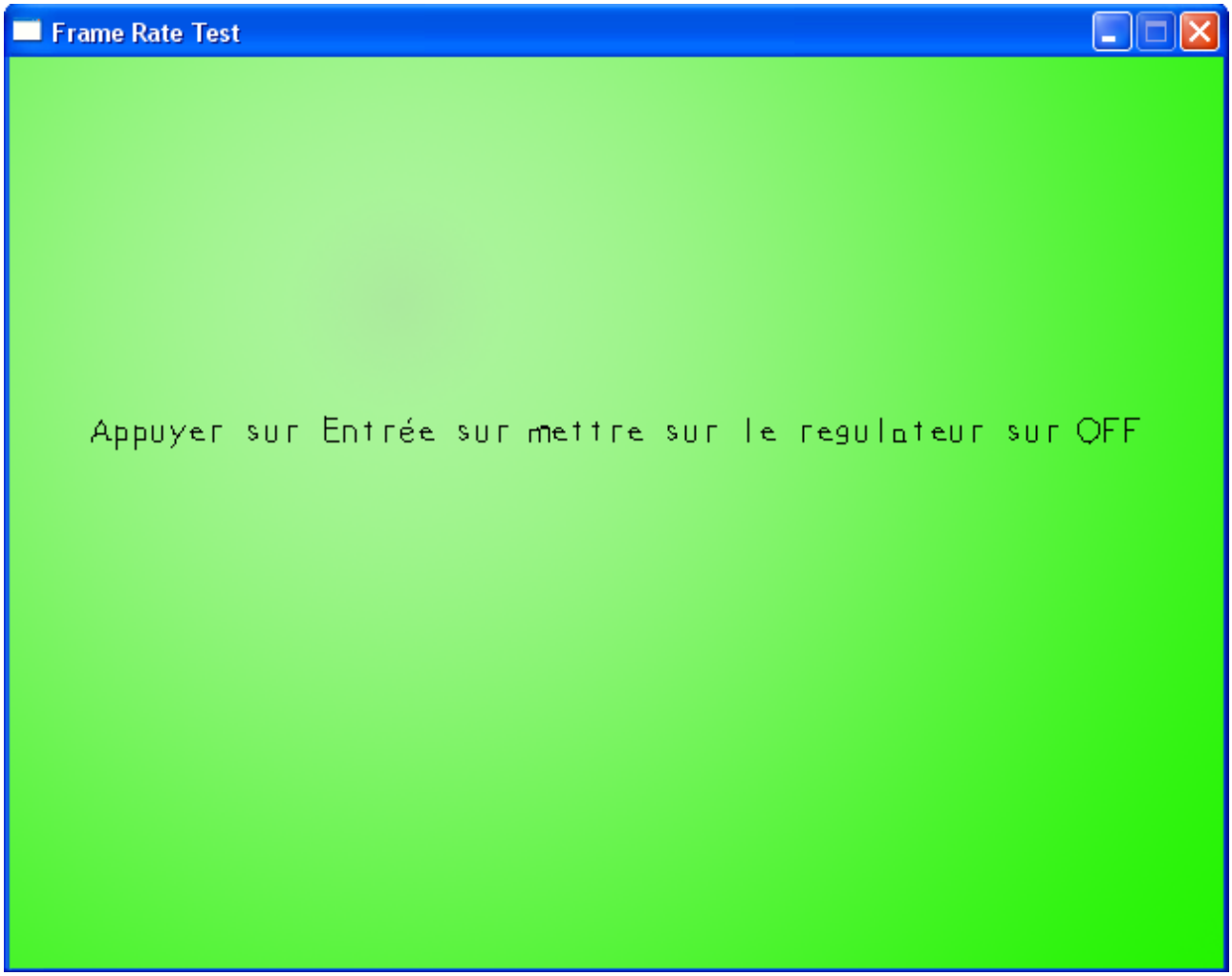
La régulation du nombre de frames par seconde est un élément important dans le développement de certains types de jeux vidéo ou d'applications.

En effet, si un jeu tourne trop rapidement, bien plus que le concepteur l'aurait voulu, celui-ci peut devenir totalement injouable.

En régulant le nombre de frames par seconde, vous évitez ainsi que le jeu ou l'application tourne trop rapidement, voir ceux-ci aient des variations de vitesse trop importantes.

Ce chapitre va vous apprendre comment faire ça en vous donnant un exemple concret.

Dans cet exemple, il y aura simplement un message qui traversera la fenêtre SDL de haut en bas comme sur l'image ci-dessous.



Le texte descendra à une certaine vitesse, régulière, lorsque le régulateur sera actif.

Vous pourrez alors appuyer sur la touche Entrée afin de désactiver le régulateur et voir ce qui se passe.

Le retour en vitesse réguler est bien évidemment aussi possible.

Tout d'abord nous allons définir notre nombre de frames par seconde "idéal" comme une constante globale.

variable global
```
//Le nombre de frames par seconde
const int FRAMES_PER_SECOND = 20;
```

Comme vous pouvez le voir, nous avons choisi 20 pour cet exemple.

Cette valeur a été décidé comme étant très bonne, cependant je ne dis pas qu'elle est la meilleure, tout dépend de votre jeu ou de votre application.

Encore une fois, je vous incite à changer cette valeur afin de trouver celle qui vous convient.

Nous allons définir encore quelques variables, cette fois-ci dans notre main :

main variables
```
int frame = 0;

//Ce qui va nous servir a activer/desactiver le regulateur
bool cap = true;

//Le regulateur
Timer fps;
```

frame sera notre compteur de frame, ce qui sera important pour savoir où poser notre surface message.

cap sera notre variable booléenne qui nous dira si le régulateur est activé ou non.

Enfin, *fps* sera notre timer régulateur.

Après initialisation de SDL et le chargement des fichiers, nous allons créer notre message qui traversera notre écran.

surface message
```
//Generation de la surface message
message = TTF_RenderText_Solid( font, "Appuyer sur Entrée sur mettre sur le regulateur sur OFF",
textColor );
```

Au départ, la variable *cap* étant à *true*, nous gênérons le bon message. Ensuite nous allons entrer dans la boucle principale.

Au début de chaque frame, nous devons lancer notre timer *fps*.

lancement fps
```
//Tant que l'utilisateur n'a pas quitte
while( quit == false )
{
        //On demarre le timer fps
        fps.start();
```

Ce qui va suivre est assez simple.

Nous allons vérifier si la touche entrée à été pressée et si c'est le cas, nous allons changer le statut de la variable *cap*.

Ensuite, nous allons vérifier la valeur de cette variable *cap* afin de générer le bon message traversant la fenêtre.

generation message
```
   //Tant qu'il y a un evenement
           while( SDL_PollEvent( &event ) )
           {
               //Si une touche est pressee
               if( event.type == SDL_KEYDOWN )
               {
                   //Si la touche Entrée est pressee
                   if( event.key.keysym.sym == SDLK_RETURN )
                   {
                       //On tourne le regulateur sur on/off
                       cap = ( !cap );

                       if (cap == true) {
                           //Generation de la surface message
                           message = TTF_RenderText_Solid( font, "Appuyer sur Entrée sur mettre sur le
   regulateur sur OFF", textColor );
                       }
                       else {
                           //Generation de la surface message
                           message = TTF_RenderText_Solid( font, "Appuyer sur Entrée sur mettre sur le
   regulateur sur ON", textColor );
                           }
                   }
               }

               //Si l'utilisateur a clique sur le X de la fenetre
               else if( event.type == SDL_QUIT )
               {
                   //On quitte the programme
                   quit = true;
               }
           }
```

Nous vérifions comme d'habitude aussi la fermeture de la fenêtre SDL par le bouton 'X' de la fenêtre.

Ensuite il nous faut poser la surface message sur l'écran.

surface message
```
   //On pose le message
   apply_surface( ( SCREEN_WIDTH - message->w) / 2,( ( SCREEN_HEIGHT + message->h * 2) /
   FRAMES_PER_SECOND) *
                  ( frame % FRAMES_PER_SECOND ) - message->h, message, screen );
```

Ne vous inquiétez pas sur tout ce code afin de simplement poser le message au bon endroit sur l'écran.

C'est juste une façon plus courte de faire ainsi :

```
   if( frame % FRAMES_PER_SECOND == 0 )
   {
   //poser ici
   }
   if( frame % FRAMES_PER_SECOND == 1 )
   {
   //poser ici
   }

   etc, etc.
```

Ensuite, il nous faut mettre à jour l'écran et incrémenter le compteur de frames.

maj écran + incrementer frame
```
   //Mise à jour de l'ecran
           if( SDL_Flip( screen ) == -1 )
           {
               return EXIT_FAILURE;
           }

           //On incremente le compteur de frames
           frame++;
```

Il nous reste encore une chose à faire avant de finir :

La partie de code nous permettant de réguler le nombre de frames par seconde.

maj écran + incrementer frame
```
   //Si nous voulons reguler le frame rate
           if( cap == true )
           {
               //Tant que le timer fps n'est pas assez haut
               while( fps.get_ticks() < 1000 / FRAMES_PER_SECOND )
               {
                   //On attend...
               }
           }
```

Quand nous démarrons une frame, nous démarrons un timer pour garder la trace du temps mit afin de rendre cette frame.

Afin que ce programme n'aille pas trop vite, chaque frame doit mettre une certaine quantité de temps.

Lorsque 20 frames ont été montrées par seconde, chaque frame doit mettre pas moins que 1/20ème d'une seconde.

Si le frame rate est à 60fps, chaque frame doit donc mettre pas moins de 1/60ème d'une seconde.

Finalement, ce que cette partie de code effectue est une attente de 1/20ème de seconde.

Télécharger les sources du IX-B (361 ko)

Version pdf (338 ko - 11 pages)

IX-C - Utilisation d'une librairie spécialisée dans le contrôle du taux de rafraîchissement

Il existe en effet une librairie SDL qui permet de contrôler le taux de rafraîchissement très simplement.

Cette librairie se nomme SDL_gfx

Les composants actuels de cette librairie sont :

- SDL_gfxPrimitves.h qui sont des primitives graphique
- SDL_rotozoom.h qui permet d'effectuer des zooms et des rotations
- SDL_imageFilter.h qui contient des filtres d'images
- SDL_framerate.h qui nous intéresse car c'est cette partie qui permet de contrôler le taux de rafraîchissement

Pour voir comment utiliser SDL_framerate.h, je vous conseille de suivre le tutoriel de fearyourself qui y consacre une partie ici

IX-D - Ressources et autres exemples

vitesse d'affichage - FAQ SDL

Programmation d'un Pong en SDL/OpenGL par Jean Christophe Beyler

Ce dernier nous montres ici la gestion de la fréquence d'affichage dans un programme plus complexe (un Pong).

Je vous invite à lire entièrement son tuto car j'ai eu plaisir à le lire.