Introduction :

Bienvenue dans ce tutorial qui va vous permettre d'apprendre à concevoir un casse brique (oui idée plus qu'original mais efficace pour apprendre) pour GP32.
La première phase pour développer un jeu c'est d'avoir un bon environnement de développement et ensuite avoir quelque base en programmation sinon vous courez droit dans le mur. Ensuite nous verrons les différentes étapes pour arriver à programmer notre casse brique.
Vous êtes prêt alors c'est partit !

Qu'est qu'un casse brique ?

Pour une meilleure compréhension je vais rappelle brièvement le principe du casse briques. Il s'agit d'un jeu mettant en scène, en bas de l'écran, une raquette (ou barre) que l'on peut déplacer horizontalement, et en haut de l'écran un amas de briques. Le but du jeu est, à l'aide d'une balle, de heurter ces briques afin de toutes les faire disparaître, cela nous permettant de passer au niveau suivant. Pour simplifier, nous dirons ici que le jeu se terminera quand toutes les briques auront disparu. La balle servant de projectile rebondit sur les côtés de l'écran, sur les briques et sur le « plafond », mais par sur le « sol ». Le joueur doit donc, à l'aide de la raquette, empêcher la balle de « tomber ».

Grâce à ce rappel, on peut maintenant distinguer les différentes étapes qui composeront notre programme : déplacement de la raquette suivant la direction imposée par l'utilisateur, déplacement de la balle : collision avec les côtés de l'écran (haut, gauche, droite), et la barre du joueur, l'affichage des différents sprites (raquette, balle, background et briques) et enfin faire disparaître les briques.

Mise en place de votre environnement de développement

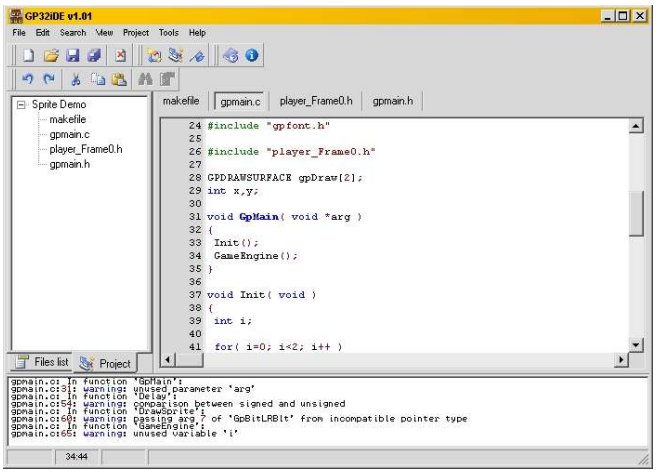
Casse brique :

Niveau > Initié
Langage : C
IDE : Gp32Ide
Outils : Gp32Converter
Tool Kits : DevKitAdv avec Sdk officiel pour Gp32

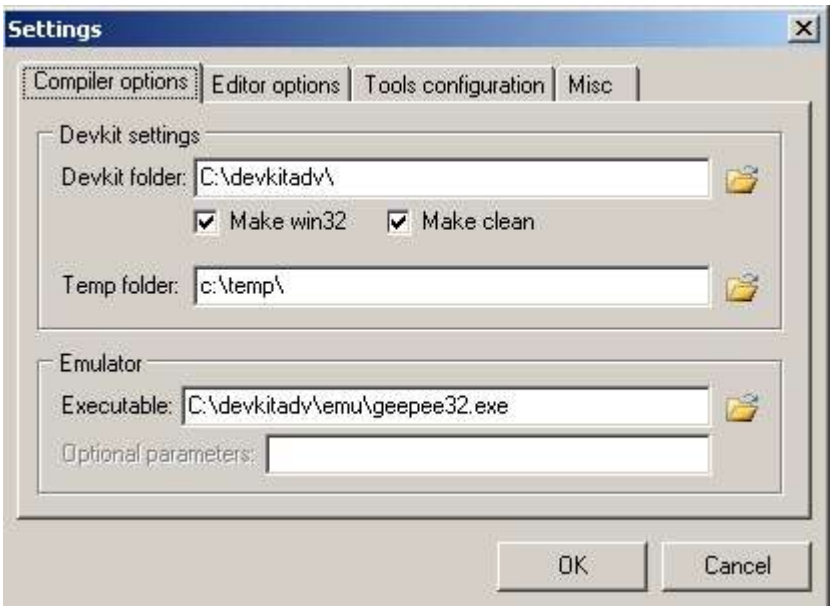
Gp32Ide :

Pour me faciliter la tâche j'ai décidé d'utiliser le GP32Ide qui est pour moi très simple d'emploi. Le GP32Ide est une application qui permet de coder tout simplement et surtout qui permet de ne pas s'occuper du problème de compilation qui je suis sur en a déjà bloqué plus d'un. Cette application a été crée par Alessandro Manno.

Fenêtre de Gp32ide :



Avant d'installer GP32Ide, il vous faut récupérer Devkitadv qui permet de compiler pour l'installer ce n'est pas très compliqué, il suffit de l'extraire sur votre disque dur.
Cela fait, vous pouvez installer le Gp32ide, pour régler le compilateur il faut lancer gp32ide, et aller dans le menu « view » et choisir settings (ou alors f8 tout simplement), cette fenêtre va apparaître :



Ici vous devez régler le chemin où se trouve Devkitadv et l'émulateur gp32 (Geepee32 celui-ci se trouve dans Devkitadv), après suffit de valider le tout de redémarrer Gp32ide et c'est parti !!!

Ensuite pour créer un projet il suffit de faire File->New->GP Project Wizard, là un squelette de programme apparaît dans la fenêtre principale vous pouvez étudier un peu le code vous verrez que ça permet d'afficher le fameux message « Hello world ! », on peut voir aussi que le programme est bien structurée en fonction : Initialisation de l'écran, le cœur du programme c'est-à-dire l'affichage de « Hello world ! », et du fameux Gpmain().

Voila maintenant que tout votre environnement est près passons aux choses sérieuses.

Headers :

Un header signifie en français une entête. En C ils sont identifiables par une extension ".h". Ces fichiers peuvent être assimilés à des interfaces.

Un header peut être constitué de plusieurs choses :

- De déclaration de constantes
- De déclaration de structures
- De prototype de fonctions

Il est conseillé de garder cet ordre car en général les constantes sont utilisées dans la déclaration des structures. Et enfin les prototypes de fonctions utilisent ces structures.

Application :

Ici on inclus dans un seul fichier toutes les déclarations que l'on a besoin :

Gpmain.h

```
#ifndef gpmain_h
#define gpmain_h

        #include <stdlib.h>
        #include "gpdef.h"
        #include "gpstdlib.h"
        #include "gpgraphic.h"
        #include "gpmain.h"
        #include "gpstdio.h"
        #include "gpfont.h"

        /*Déclaration des fonctions du programme*/

        void GpMain(void * arg);

#endif
```

Dans ce gpmain.h nous trouvons les déclarations de toutes les bibliothèques qui peuvent être utilisés dans notre programme et aussi les déclarations de nos fonctions sauf que ici les fonctions étant placées avant le gpmain() nous n'avons pas besoin de les déclarer. Par contre si vous les placées après n'oubliez pas de les déclarer dans le gpmain.h.

Voila maintenant lorsque nous ajouterons les sprites et les variables il faudra les déclarer dans notre gpmain.h. Et pour inclure ce gpmain.h dans notre programme il suffit tout simplement de faire un : **#include** "gpmain.h" dans gpmain.c.

Image en fond :

Maintenant que vous avez compris le concept des entêtes nous allons voir comment afficher une image en fond.

Pour avoir une image de fond quelques règles sont à respecter :

- Maximum 256 couleurs (car nous travaillons en 8bits)
- Taille de l'image 320*240
- Couleur indexée (avec Adobe Photoshop on utilise facilement le RGB)

Plus de détails pour la création d'image de fond avec photoshop :

Lancer Photoshop

Soit vous avez déjà une image de fond soit vous en créez une :

" Fichier" > " Nouveau"
Longueur : 320 pixels, Hauteur : 240 pixels
Mode : RGB

Ensuite :

"Image" > "Mode" > "Couleur Indexée "
Palette: Uniforme
Couleurs: 256
Appuyer sur "OK"

Après vous l'enregistrer en format BMP pour que Gp32Converter puisse la convertir en header.

Une fois votre image de fond crée, il vous faut la convertir en header c'est la qu'intervient Gp32Converter.
Il est très simple d'emploi le seul inconvénient est qu'il faut que les images soient en *.BMP car il ne reconnaît que ce format.

Maintenant

Aide pour convertir un BMP en header sous Gp32Converter :

"Load image file " choisir votre BMP à convertir
A gauche on peut voir les caractéristiques de l'image et en dessous le nom de la variable donné à l'image notées les sur un papier elles serviront plus tard dans notre exemple.

à ça :

Exemple de bg.h :

```
#define bg_width      320
#define bg_height     240

#define bg_palnb      256

#include "gpgraphic.h"

GP_PALETTEENTRY bg_Pal[256] = {
      0x1, 0x8001, 0x401, 0x8401, 0x21, 0x8021, 0x421, 0xC631, 0xC6F1, 0xA67D, 0x4101,
0x6101, 0x8101, 0xA101, 0xC101, 0xE101,
      0x201, 0x2201, 0x4201, 0x6201, 0x8201, 0xA201, 0xC201, 0xE201, 0x301, 0x2301,
0x4301, 0x6301, 0x8301, 0xA301, 0xC301, 0xE301,
      0x401, 0x2401, 0x4401, 0x6401, 0x8401, 0xA401,
                        .. .. ..
}

const unsigned char bg[76800] = {
      0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0,
      0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0,
      0x0, 0x0, 0x0, 0x0, 0x0,
                  .. .. ..
}
```

Pour pouvoir utiliser votre header il faut le copier dans votre répertoire de votre projet. Ensuite il faut rattacher votre header à votre programme pour cela il suffit de rajouter à votre gpmain.h la ligne suivante : #include lenomdevotreheader.h, et dans votre header rajouter : #include gpmain.h. Pour tous les autres sprites que vous rajouterez dans votre programme il faudra faire la même chose.

Maintenant entrons dans le vif du sujet comment faire pour afficher cette image de fond. Pour exécuter cette tâche il faut la décomposer de plusieurs étapes : tout d'abord il faut initialiser l'écran (créer des surfaces d'affichage), et seulement ensuite nous verrons comment faire pour afficher ce sprite.

Initialisation de l'écran :

L'initialisation de l'écran commence par la création d'une surface c'est à dire qu'on crée une zone sur l'écran où l'on pourra affiche ce que l'on veut.

Nous ici ce qui nous intéresse c'est de pouvoir faire déplacer notre barre et notre boules sans qu'il y ai d'effet de clignotement, il faut pour cela créer un buffer vidéo.

Gpmain.h

```
#ifndef gpmain_h
#define gpmain_h

        #include <stdlib.h>
        #include "gpdef.h"
        #include "gpstdlib.h"
        #include "gpgraphic.h"
        #include "gpmain.h"
        #include "gpstdio.h"
        #include "gpfont.h"

        int nflip=1;            /* index du buffer*/
        GPDRAWSURFACE gpDraw[2];    /* Buffer video */

        /*Déclaration des fonctions du programme*/

        void GpMain(void * arg);

#endif
```

Comme
vidéo (nflip) et le buffer lui même (GPDRAWSURFACE).
GPDRAWSURFACE est une structure, elle est défini dans le sdk, on l'utilise ici pour créer deux surfaces d'écran (gpDraw[2]) qui permettent d'avoir un buffer vidéo et ainsi d'éviter l'effet de clignotement. La variable nflip permet d'accéder à ces deux surfaces d'écran.

Maintenant voyons comment initialiser ces 2 surfaces. Pour cela on crée une fonction Init_Ecran.

Gpmain.c

```
#include  "gpmain.h"

/*Initialisation de l'écran */

void Init_Ecran(void)
{
int i;

for(i=0;i<2;i++)
{
        GpLcdSurfaceGet(&gpDraw[i],i);
}

GpSurfaceSet(&gpDraw[0]);
}

/*fonction principale */

void GpMain(void *arg)
{

        Init_Ecran();  /*appel de la fonction Init_Ecran*/

 while(1) /*boucle tant que , elle n'a pas de conditions elle ne s'arrête donc jamais */
        {
                GpSurfaceFlip(&gpDraw[nflip++]); /*permet de change les surfaces grâce a
                nflip &= 0x01;                                  l'index buffer (nflip)*/
        }
}
```

Description de la fonction Init_Ecran :

Vous pouvez voir que la fonction Init_Ecran est très basique, elle comporte juste une boucle for avec une seule instruction.
Ici la fonction est appelé au tout début de notre main, on obtient grâce à cette fonction 2 surfaces d'écran.

Pour que celles-ci soient tout le temps active dans notre programme on a crée dans le main une boucle « while » qui ne s'arrête jamais (plus tard il faudra change de condition, pour instaurer une condition d'arrêt). Dans cette boucle « while « , nous appelons la fonction GpSurfaceFlip elle permet de changer de surface (« Flip » comme renverser) par

l'intermédiaire de notre index, on peut voir que celui ci est envoyé comme paramètre dans la fonction : GpSurfaceFlip(&gpDraw[nflip++]).

Affichage d'une image de fond :

Gpmain.c

```
void Init_Sprites(void)
{
        /** Affichage de l'image a la position 0,0 de taille 320,240 **/
        GpBitBlt(NULL,&gpDraw[nflip], 0, 0, 320, 240, (unsigned char*)bg, 0, 0 ,  320, 240);

}
```

GpBiltBlt permet d'afficher l'image qui s'appelle bg (le nom est celui que vous avez donné a votre image lors de la conversion de celle-ci en « .h » il s'agit d'un pointeur qui fait référence a votre image), celle ci sera affiche a la position 0,0 et aura une taille de 320 par 240. On donne aussi en paramètre de la fonction sur quelle surface d'écran l'on souhaite que notre image s'affiche ici nous avons mis sur gpDraw[nflip] .

Votre gpmain.c devrait ressemble un peu a cela maintenant :

Gpmain.c

```
#include  "gpmain.h"

/*Initialisation de l'écran */

void Init_Ecran(void)
{
int i;

for(i=0;i<2;i++)
{
        GpLcdSurfaceGet(&gpDraw[i],i);
}

GpSurfaceSet(&gpDraw[0]);
}

/* affichage de l'image de fond */

void Init_Sprites(void)
{
        /** Affichage de l'image a la position 0,0 de taille 320,240 **/
        GpBitBlt(NULL,&gpDraw[nflip], 0, 0, 320, 240, (unsigned char*)bg, 0, 0 ,  320,
240);

}

/*fonction principale */

void GpMain(void *arg)
{

        Init_Ecran();  /*appel de la fonction Init_Ecran*/

 while(1) /*boucle tant que , elle n'a pas de conditions elle ne s'arrête donc jamais */
        {
                Init_Sprites(); /* appelle la fonction pour initialise l'ecran */
                GpSurfaceFlip(&gpDraw[nflip++]); /*permet de change les surfaces
grâce a
                nflip &= 0x01;                          l'index buffer (nflip)*/
        }
}
```