Compte-rendu du jeu d'arcade "Space Invaders modifié"

Nom : Gaillard
Prénom : Alfred
Groupe : D2

Choix de conception

Etape 1 :

J'ai défini un premier timer grâce à l'un des td, à l'aide d'une formule explicative trouvée sur le site du zéro (et qui est sûrement dans le cours) j'ai calculé les valeurs nécessaires pour arriver au bout de la boucle 160 fois par secondes.
Grâce à un flag d'interruption (défini sous le nom UIF) on peut savoir lorsque l'on est à la fin de la boucle, la réinitialiser et augmenter l'abscisse si celui-ci ne se situe pas en dehors de l'écran

```
/*-------------------------------------------------------------
  Timer
 *-------------------------------------------------------------
void cfgTimer1(void){
    RCC->APB2ENR |= (1 << 11);
    TIM1->PSC = 9; //On choisi un PSC à 9
    TIM1->ARR = 45000; //72/10=7.2Mhz = 7.2*10^6Hz/160 = 45000
    TIM1->DIER |= (1<<0);
    TIM1->CR1 |= 0x0001;
    SETENA0 |= (1<<25);
}


/*-------------------------------------------------------------
  Interruptions timer
 *-------------------------------------------------------------
void TIM1_UP_TIM10_IRQHandler (void){
    if(TIM1->SR & UIF)
    {
        TIM1->SR &= ~UIF; //Abaissement du flag d'interruption
      //DEPLACEMENTS JOUEUR
        if (lAbscisse != (320 - LARGEUR_BOULE)) lAbscisse++;
    }
}
```

```
    while(1){

        GLCD_DrawBitmap(lAbscisse , lOrdonne , LARGEUR_BOULE , HAUTEUR_BOULE , (const unsigned char *)vaisseau);

        //Effacement derrière le vaisseau lorsqu'il se déplace en abcisse
        if(lAbscisse > dernierAbs){
            GLCD_SetForegroundColor(GLCD_COLOR_BLACK);
            GLCD_DrawVLine(lAbscisse - 1 , lOrdonne , HAUTEUR_BOULE);
            effacementJoueur = true;
        }
    }
}
```

De plus, pour ne pas laisser la marque du vaisseau sur tous l'écran au moment de son déplacement je trace une ligne noir grâce à une variable mémoire dans laquelle est stockée l'abscisse de sa position précédente, on analyse continuellement sa position grâce à une boucle while qui s'exécute en parallèles des timers.

Etape 2 :

Pour cette partie j'ai eu besoin de mettre en service et d'initialiser les GPIO
Pour les valeurs je suis allé les chercher dans le premier cours.

```
/*-------------------------------------------------------------------------------
  GPIO
 *-------------------------------------------------------------------------------*/
void Enable_GPIO(void){ //Mise en service
    RCC->APB2ENR |= (1<<5); //GPIOD
    RCC->APB2ENR |= (1<<8); //GPIOG
}

void Init_GPIO(void){ //Initialisation
    GPIOD->CRL = 0x00004000; //GPIOD \ DOWN
    GPIOG->CRH = 0x44400004; //GPIOG \ LEFT || RIGHT || UP || USER, déjà prêt
}
```

Il faut déclarer les boutons qui pointerons vers des valeurs des registres appropriés et permettrons ainsi d'analyser les actions de l'utilisateur en IDR (input).

```
    Déclaration boutons
*-----------------------------------*/
unsigned long bouton_DOWN; //DOWN
unsigned long bouton_LEFT; //LEFT
unsigned long bouton_RIGHT; //RIGHT
unsigned long bouton_UP; //UP
```

Puis dans la boucle while, on pointe enfin vers les registres des broches GPIO en IDR.

```
while(1){
/*-------------------------------------
Définition de l'idr (input) des boutons
*-------------------------------------*/

    bouton_DOWN = GPIOD->IDR & 0x0008;
    bouton_LEFT = GPIOG->IDR & 0x4000;
    bouton_RIGHT = GPIOG->IDR & 0x2000;
    bouton_UP = GPIOG->IDR & 0x8000;
```

On définit une valeur «APPUYE» qui sera la même pour tous les 4 boutons.

```
#define APPUYE 0x0000
```

Pour faciliter la compréhension du code (et surtout pour ma compréhension personnelle), j'ai utilisé

```
/*-------------------------------------------------------------------------------
  Etats
 *-------------------------------------------------------------------------------*/
enum EtatDirectionnel {UP = 0 , RIGHT = 1 , DOWN = 2 , LEFT = 3 , NONE = 6};
int currentDirectionState , dernierAbs , dernierOrd;
bool effacementJoueur = true;
```

beaucoup d'énumération, je l'ai également conseillé à 4 autres groupes, ça m'a permis de continuer à suivre ce que je faisais tout au long de la conception et je pense en utiliser plus souvent à l'avenir.

```
switch(currentDirectionState){
    case RIGHT:
        if (verificationDeplacementsJoueur(RIGHT)){
            dernierAbs = lAbscisse;
            lAbscisse++;
            effacementJoueur = false;
        }
        break;
    case LEFT:
        if (verificationDeplacementsJoueur(LEFT)){
            dernierAbs = lAbscisse;
            lAbscisse--;
            effacementJoueur = false;
        }
        break;
    case UP:
        if (verificationDeplacementsJoueur(UP)){
            dernierOrd = lOrdonne;
            lOrdonne--;
            effacementJoueur = false;
        }
        break;
    case DOWN:
        if (verificationDeplacementsJoueur(DOWN)){
            dernierOrd = lOrdonne;
            lOrdonne++;
            effacementJoueur = false;
        }
        break;
}
```

En fonction de la direction indiqué par les joysticks je déplace le joueur dans la direction voulue.

Pour cela j'utilise une fonction qui vérifia les conditions nécessaires au déplacements.

Là encore, en fonction de la direction choisie je vérifie si les déplacements sont autorisés, j'aurais pu le faire en une fois mais j'ai décidé de procéder à ce genre d'optimisations plus tard pour aller le plus loin possible (j'ai considérer qu'il s'agissait de besoin non fonctionnel).

```
/*-------------------------------------------------------------------------
  Vérification des conditions avant le déplacements du joueur
 *-------------------------------------------------------------------------*/
bool verificationDeplacementsJoueur(int orientation){
    switch(currentDirectionState){
        case RIGHT:
            if (!effacementJoueur) return false;
            else if (lAbscisse != (320 - LARGEUR_BOULE)) return true;
            break;
        case LEFT:
            if (!effacementJoueur) return false;
            else if (lAbscisse != 0) return true;
            break;
        case UP:
            if (!effacementJoueur) return false;
            else if (false) return false;
            if (lOrdonne != 9) return true;
            break;
        case DOWN:
            if (!effacementJoueur) return false;
            else if (lOrdonne != (240 - LARGEUR_BOULE)) return true;
            break;
    }
    return false;
}
```

Etape 3 :

Voici la déclaration des 2 nouveaux boutons

```
/*-------------------------------------
    Déclaration boutons
*-------------------------------------*/
unsigned long bouton_WAKEUP; //WAKEUP
unsigned long bouton_TEMPER; //TAMPER
```

La valeur d'appuie du bouton wakeup est de 1,
j'ai donc défini une valeur d'appui spécialement pour lui.

```
#define APPUYE_wakeup 1
```

J'ai déclaré une orientation, et grâce aux enum il sera facile de distinguer les cas

```
int currentOrientationState
```

Ainsi je met par défaut l'orientation du vaisseau vers le haut et sa direction est nulle.

```
currentOrientationState = UP; //Définition var orientation
currentDirectionState = NONE; //Définition var direction
```

Ici je détecte l'appuie du bouton wakeup et tamper (une faute d'orthographe est présente dans tout mon code mais ça lui donne du charme), j'utilise une variable booléenne pour savoir si il les a relâchés et ainsi procéder à la rotation une seule fois.

```
//WAKEUP
if (bouton_WAKEUP != APPUYE_wakeup) aEteRelache[WAKEUP] = true; //VALEUR D'APPUYE INVERSE POUR WAKEUP
else if (aEteRelache[WAKEUP]){
    currentOrientationState++;
    aEteRelache[WAKEUP] = false;
    if(currentOrientationState > 3) currentOrientationState = 0; //Retour pos0 si tour complet effectué
}

//TEMPER
if (bouton_TEMPER != APPUYE) aEteRelache[TEMPER] = true;
else if(aEteRelache[TEMPER]){
    currentOrientationState--;
    aEteRelache[TEMPER] = false;
    if(currentOrientationState < 0) currentOrientationState = 3; //Retour pos3 si on part de pos0
}
```

Pour bien comprendre le code précédent, je tiens à préciser que j'ai là encore ajouté des enum pour mieux m'y retrouver

```
/*-------------------------------------------------------------------------
  Etats
 *-------------------------------------------------------------------------*/
enum EtatDirectionnel {UP = 0 , RIGHT = 1 , DOWN = 2 , LEFT = 3 , NONE = 6};
enum EtatAppuie {WAKEUP = 0 , TEMPER = 1 , USER = 2};
```

J'ai également adapté l'effacement derrière le vaisseau pour que celui-ci ne laisse pas de traces dans l'ensemble de ses déplacements.

```
//Effacement derrière le vaisseau lorsqu'il se déplace en abcisse
if(lAbscisse > dernierAbs){
    GLCD_SetForegroundColor(GLCD_COLOR_BLACK);
    GLCD_DrawVLine(lAbscisse - 1 , lOrdonne , HAUTEUR_BOULE);
    effacementJoueur = true;
}else if(lAbscisse < dernierAbs){
    GLCD_SetForegroundColor(GLCD_COLOR_BLACK);
    GLCD_DrawVLine((lAbscisse + LARGEUR_BOULE) , lOrdonne , HAUTEUR_BOULE);
    effacementJoueur = true;
}

//Effacement derrière le vaisseau lorsqu'il se déplace en ordonne
if(lOrdonne > dernierOrd){
    GLCD_SetForegroundColor(GLCD_COLOR_BLACK);
    GLCD_DrawHLine(lAbscisse , (lOrdonne - 1) , HAUTEUR_BOULE);
    effacementJoueur = true;
}else if(lOrdonne < dernierOrd){
    GLCD_SetForegroundColor(GLCD_COLOR_BLACK);
    GLCD_DrawHLine(lAbscisse , (lOrdonne + HAUTEUR_BOULE) , HAUTEUR_BOULE);
    effacementJoueur = true;
}
```

Pour la rotation, il m’a paru plus logique, simple et surtout optimisé de ne pas utiliser d’opération de matrice que j’avais déjà utilisé dans un précédent projet, j’ai tout simplement crée un tableau de 4 images différentes avec un indice lié aux enum d’orientation, ce qui m’a permis de passer très rapidement cette étape (et entre nous, j’ai fais l’étape 3 avant la 2).

```
//UP = 0 ||| RIGHT = 1 ||| DOWN = 2 ||| LEFT = 3
const unsigned short vaisseau[4][HAUTEUR_BOULE*LARGEUR_BOULE] = {
    {
        0x0000,0x0000,0x0000,0x4A49,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x528A,0x0000,0x0000,0x0000,
        0x0000,0x0000,0x0020,0x9CF3,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0xA534,0x0000,0x0000,0x0000,
        0x0000,0x0000,0x0020,0xCE59,0xFFDF,0x0841,0x0000,0x0000,0x0000,0x0000,0x18C3,0xFFFF,0xC638,0x0000,0x0000,0x0000,
        0x0000,0x0000,0x0861,0xC638,0xFFFF,0xFFFF,0x73AE,0x7BEF,0x7BCF,0x7BEF,0xFFFF,0xFFFF,0xCE59,0x0000,0x0000,0x0000,
        0x0000,0x0000,0x0841,0xBDF7,0xBDF7,0xFFFF,0xF79E,0x94B2,0x9492,0xFFDF,0xFFFF,0xAD75,0xC638,0x0000,0x0000,0x0000,
        0x0000,0x0000,0x0841,0xAD55,0x0000,0xFFFF,0xEF5D,0x5ACB,0x5AEB,0xF7BE,0xFFFF,0x0000,0xAD75,0x0000,0x0000,0x0000,
        0x0000,0x0000,0x0000,0x9CD3,0x0000,0x632C,0xFFFF,0x630C,0x632C,0xFFFF,0x4A49,0x0000,0x9CF3,0x0000,0x0000,0x0000,
        0x0000,0x0000,0x0000,0xB5B6,0x0000,0x0000,0xDEDB,0x8410,0x9492,0xC618,0x0000,0x0000,0xB596,0x0000,0x0000,0x0000,
        0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x94B2,0xBDD7,0xC638,0x8C51,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,
        0x0000,0x0000,0x0000,0x0000,0x0000,0xD69A,0xB596,0x9CD3,0x94B2,0xBDD7,0xCE79,0x0000,0x0000,0x0000,0x0000,0x0000,
        0x0000,0x0000,0x0000,0x0000,0x8430,0xC618,0x39E7,0xCE59,0xC5F7,0x39E7,0xCE59,0x7BCF,0x0000,0x0000,0x0000,0x0000,
        0x0000,0x0000,0x0000,0x0000,0xC638,0x738E,0xAD55,0xB596,0xB596,0x9CD3,0x8430,0xBDD7,0x0000,0x0000,0x0000,0x0000,
        0x0000,0x0000,0x0000,0x0000,0xC5F8,0x7BEF,0x9492,0xB5B6,0xBDD7,0x8410,0x9492,0xB596,0x0000,0x0000,0x0000,0x0000,
        0x0000,0x0000,0x0000,0x0000,0x7BCF,0xCE79,0x4228,0x73AE,0x738E,0x528A,0xCE79,0x6B6D,0x0000,0x0000,0x0000,0x0000,
        0x0000,0x0000,0x0000,0x0000,0x0000,0xC618,0xCE79,0x94B2,0x9CD3,0xD69A,0xBDD7,0x0000,0x0000,0x0000,0x0000,0x0000,
        0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x5AEB,0x9CD3,0x94B2,0x5ACB,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000
    },{
        0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,
        0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,
        0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,
        0x528A,0xA534,0xC638,0xCE59,0xC638,0xAD75,0x9CF3,0xB596,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,
        0x0000,0x0000,0xFFFF,0xFFFF,0xAD75,0x0000,0x0000,0x0000,0x0000,0x0000,0x7BCF,0xBDD7,0xB596,0x6B6D,0x0000,0x0000,
        0x0000,0x0000,0x18C3,0xFFFF,0xFFFF,0xFFFF,0x4A49,0x0000,0x0000,0xCE79,0xCE59,0x8430,0x9492,0xCE79,0xBDD7,0x0000,
        0x0000,0x0000,0x0000,0x7BEF,0xFFDF,0xF7BE,0xFFFF,0xC618,0x8C51,0xBDD7,0x39E7,0x9CD3,0x8410,0x528A,0xD69A,0x5ACB,
        0x0000,0x0000,0x0000,0x7BCF,0x9492,0x5AEB,0x632C,0x9492,0xC638,0x94B2,0xC5F7,0xB596,0xBDD7,0x738E,0x9CD3,0x94B2,
        0x0000,0x0000,0x0000,0x7BEF,0x94B2,0x5ACB,0x630C,0x8410,0xBDD7,0x9CD3,0xCE59,0xB596,0xB5B6,0x73AE,0x94B2,0x9CD3,
        0x0000,0x0000,0x0000,0x73AE,0xF79E,0xEF5D,0xFFFF,0xDEDB,0x94B2,0xB596,0x39E7,0xAD55,0x9492,0x4228,0xCE79,0x5AEB,
        0x0000,0x0000,0x0841,0xFFFF,0xFFFF,0xFFFF,0x632C,0x0000,0x0000,0xD69A,0xC618,0x738E,0x7BEF,0xCE79,0xC618,0x0000,
        0x0000,0x0000,0xFFDF,0xFFFF,0xBDF7,0x0000,0x0000,0x0000,0x0000,0x0000,0x8430,0xC638,0xC5F8,0x7BCF,0x0000,0x0000,
        0x4A49,0x9CF3,0xCE59,0xC638,0xBDF7,0xAD55,0x9CD3,0xB5B6,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,
        0x0000,0x0020,0x0020,0x0861,0x0841,0x0841,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,
        0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,
        0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000
    },{
        0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x5ACB,0x94B2,0x9CD3,0x5AEB,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,
        0x0000,0x0000,0x0000,0x0000,0x0000,0xBDD7,0xD69A,0x9CD3,0x94B2,0xCE79,0xC618,0x0000,0x0000,0x0000,0x0000,0x0000,
        0x0000,0x0000,0x0000,0x0000,0x6B6D,0xCE79,0x528A,0x738E,0x73AE,0x4228,0xCE79,0x7BCF,0x0000,0x0000,0x0000,0x0000,
        0x0000,0x0000,0x0000,0x0000,0xB596,0x9492,0x8410,0xBDD7,0xB5B6,0x9492,0x7BEF,0xC5F8,0x0000,0x0000,0x0000,0x0000,
        0x0000,0x0000,0x0000,0x0000,0xBDD7,0x8430,0x9CD3,0xB596,0xB596,0xAD55,0x738E,0xC638,0x0000,0x0000,0x0000,0x0000,
        0x0000,0x0000,0x0000,0x0000,0x7BCF,0xCE59,0x39E7,0xC5F7,0xCE59,0x39E7,0xC618,0x8430,0x0000,0x0000,0x0000,0x0000,
        0x0000,0x0000,0x0000,0x0000,0x0000,0xCE79,0xBDD7,0x94B2,0x9CD3,0xB596,0xD69A,0x0000,0x0000,0x0000,0x0000,0x0000,
        0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x8C51,0xC638,0xBDD7,0x94B2,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,
        0x0000,0x0000,0x0000,0xB596,0x0000,0x0000,0xC618,0x9492,0x8410,0xDEDB,0x0000,0x0000,0xB5B6,0x0000,0x0000,0x0000,
        0x0000,0x0000,0x0000,0x9CF3,0x0000,0x4A49,0xFFFF,0x632C,0x630C,0xFFFF,0x632C,0x0000,0x9CD3,0x0000,0x0000,0x0000,
        0x0000,0x0000,0x0000,0xAD75,0x0000,0xFFFF,0xF7BE,0x5AEB,0x5ACB,0xEF5D,0xFFFF,0x0000,0xAD55,0x0841,0x0000,0x0000,
        0x0000,0x0000,0x0000,0xC638,0xAD75,0xFFFF,0xFFDF,0x9492,0x94B2,0xF79E,0xFFFF,0xBDF7,0xBDF7,0x0841,0x0000,0x0000,
        0x0000,0x0000,0x0000,0xCE59,0xFFFF,0xFFFF,0x7BEF,0x7BCF,0x7BEF,0x73AE,0xFFFF,0xFFFF,0xC638,0x0861,0x0000,0x0000,
        0x0000,0x0000,0x0000,0xC638,0xFFFF,0x18C3,0x0000,0x0000,0x0000,0x0000,0x0841,0xFFDF,0xCE59,0x0020,0x0000,0x0000,
        0x0000,0x0000,0x0000,0xA534,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x9CF3,0x0020,0x0000,0x0000,
        0x0000,0x0000,0x0000,0x528A,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x4A49,0x0000,0x0000,0x0000
    },{
        0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,
        0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,
        0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0841,0x0841,0x0861,0x0020,0x0020,0x0000,
        0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0xB5B6,0x9CD3,0xAD55,0xBDF7,0xC638,0xCE59,0x9CF3,0x4A49,
        0x0000,0x0000,0x7BCF,0xC5F8,0xC638,0x8430,0x0000,0x0000,0x0000,0x0000,0x0000,0xBDF7,0xFFFF,0xFFDF,0x0000,0x0000,
        0x0000,0xC618,0xCE79,0x7BEF,0x738E,0xC618,0xD69A,0x0000,0x0000,0x632C,0xFFFF,0xFFFF,0xFFFF,0x0841,0x0000,0x0000,
        0x5AEB,0xCE79,0x4228,0x9492,0xAD55,0x39E7,0xB596,0x94B2,0xDEDB,0xFFFF,0xEF5D,0xF79E,0x73AE,0x0000,0x0000,0x0000,
        0x9CD3,0x94B2,0x73AE,0xB5B6,0xB596,0xCE59,0x9CD3,0xBDD7,0x8410,0x630C,0x5ACB,0x94B2,0x7BEF,0x0000,0x0000,0x0000,
        0x94B2,0x9CD3,0x738E,0xBDD7,0xB596,0xC5F7,0x94B2,0xC638,0x9492,0x632C,0x5AEB,0x9492,0x7BCF,0x0000,0x0000,0x0000,
        0x5ACB,0xD69A,0x528A,0x8410,0x9CD3,0x39E7,0xBDD7,0x8C51,0xC618,0xFFFF,0xF7BE,0xFFDF,0x7BEF,0x0000,0x0000,0x0000,
        0x0000,0xBDD7,0xCE79,0x9492,0x8430,0xCE59,0xCE79,0x0000,0x0000,0x4A49,0xFFFF,0xFFFF,0xFFFF,0x18C3,0x0000,0x0000,
        0x0000,0x0000,0x6B6D,0xB596,0xBDD7,0x7BCF,0x0000,0x0000,0x0000,0x0000,0x0000,0xAD75,0xFFFF,0xFFFF,0x0000,0x0000,
        0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0xB596,0x9CF3,0xAD75,0xC638,0xCE59,0xC638,0xA534,0x528A,
        0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,
        0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,
        0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000
    }
};
```

Etape 4 :

Commençons par la déclaration du nouveau bonton :

```
unsigned long bouton_USER; //USER
```

Ainsi que son ajout dans l’enum correspondant :

```
enum EtatAppuie {WAKEUP = 0 , TEMPER = 1 , USER = 2};
```

La condition que nous n’appuyons pas sur le bouton user est ajoutée dans la fonction de vérification des conditions de déplacements du joueur.

```
/*--------------------------------------------------------------------------
  Vérification des conditions avant le déplacements du joueur
 *--------------------------------------------------------------------------*/
bool verificationDeplacementsJoueur(int orientation){
    switch(currentDirectionState){
        case RIGHT:
            if (bouton_USER == APPUYE || !effacementJoueur) return false;
            else if (lAbscisse != (320 - LARGEUR_BOULE)) return true;
            break;
        case LEFT:
            if (bouton_USER == APPUYE || !effacementJoueur) return false;
            if (lAbscisse != 0 && bouton_USER != APPUYE) return true;
            break;
        case UP:
            if (bouton_USER == APPUYE || !effacementJoueur) return false;
            else if (false) return false;
            if (lOrdonne != 9 && bouton_USER != APPUYE) return true;
            break;
        case DOWN:
            if (bouton_USER == APPUYE || !effacementJoueur) return false;
            if (lOrdonne != (240 - LARGEUR_BOULE) && bouton_USER != APPUYE) return true;
            break;
    }
    return false;
}
```

On défini un 2ème timer pour faire vibrer la broche PA4

```
void cfgTimer2(void){
 RCC->APB1ENR |= (1 << 0);
 TIM2->PSC = 5;
 TIM2->ARR = 700; //Un ARR à 9000 sera plus approprié, on a ici une machine à casser des oreilles
 TIM2->DIER |= UIE;
 SETENA0 |= (1<<28);
}
```

Dans l'interruption on est plus en input mais output puisque le son émane de la carte émane de la carte et non l'inverse !

```
void TIM2_IRQHandler (void){
    if(TIM2->SR & UIF){
        GPIOA->ODR ^= (1<<4); // Si appui sur le boutton, la carte emet un son
        TIM2->SR &= ~UIF; // on baisse le drapeau
    }
}
```

Je déclare dans le main des variables mémoires pour les tires afin de pouvoir les effacer au bon endroit étant donné que le timer continue de boucler en parallèles de la boucle while et que les valeurs risqueraient donc de changer.

```
int abcTire , ordTire , longTire; //Déclarations var mémoire coordonnées
```

Dans le main encore je lance le 2ème timer et je relâche tous les boutons par défaut, j'ai également une variable booléenne mémoire pour l'effacement du lazer une fois le bouton relâché dont je défini la valeur par défaut ici.

```
cfgTimer2();

for (compteur[0] = 0 ; compteur[0] != 3 ; compteur[0]++) aEteRelache[compteur[0]] = true; //Boutons sont tous relache par defaut
for (compteur[0] = 0 ; compteur[0] != 4 ; compteur[0]++) lazerEfface[compteur[0]] = true; //Lazer pas encore tracés donc nul besoin de les effacer
```

Dans la boucle while vient s'ajouter la définition du bouton user pour écouter l'input :

```
while(1){
/*-----------------------------------------------
Définition de l'idr (input) des boutons
*-----------------------------------------------*/
    bouton_WAKEUP = GPIOA->IDR & 0x0001;
    bouton_TEMPER = GPIOC->IDR & 0x2000;
    bouton_USER = GPIOG->IDR & 0x0100;
    bouton_DOWN = GPIOD->IDR & 0x0008;
    bouton_LEFT = GPIOG->IDR & 0x4000;
    bouton_RIGHT = GPIOG->IDR & 0x2000;
    bouton_UP = GPIOG->IDR & 0x8000;
```

Grâce au switch le laser est tracé en fonction de l'orientation du vaisseau, et pour chaque orientation je reconnais 4 cas :

Le premier cas est l'appuie continuel du bouton user, dans ce cas là je fais vibrer la broche pour avoir le son strident du siffleur tout au long de l'appuie.

```
if(bouton_USER == APPUYE ){
    TIM2->CR1 |= 0x0001; //SON
}else TIM2->CR1 &= 0x0000; //SON
```

Le deuxième cas est le relâchement ou l'absentéisme d'appuie du bouton user, dans ce cas je redéfinis la variable de relâchement à true et si le laser du tir précédent n'a pas encore été effacé (et il y a aussi une faute dans tout mon code), eh bien je l'efface et je le notifie grâce à la variable booléenne déclaré préalablement : *lazerEfface[l'orientation qui correspond (et qui n'est donc rien autre qu'un indice en int)]*

```
if(bouton_USER != APPUYE){
    aEteRelache[USER] = true;
    if (!lazerEfface[currentOrientationState]){
        GLCD_SetForegroundColor(GLCD_COLOR_BLACK);
        GLCD_DrawHLine(abcTire , ordTire , longTire);
        lazerEfface[currentOrientationState] = true;
    }
```

Le troisième cas est l'appuie du bouton user alors que la boucle while effectue son premier passage (il y a énormément d'itération de la boucle while avant qu'il ai le temps de relâcher), à ce moment on trace le laser.

```
}else if (aEteRelache[USER]){
```

Le quatrième cas est le passage de ma boucle while à la première itération APRÈS que l'utilisateur ai relâche le bouton user, dans ce cas on ne fait rien.

```
//USER
switch(currentOrientationState){
    case RIGHT:
        if(bouton_USER != APPUYE){
            aEteRelache[USER] = true;
            if (!lazerEfface[currentOrientationState]){
                GLCD_SetForegroundColor(GLCD_COLOR_BLACK);
                GLCD_DrawHLine(abcTire , ordTire , longTire);
                lazerEfface[currentOrientationState] = true;
            }
        }else if (aEteRelache[USER]){
            abcTire = lAbscisse + LARGEUR_BOULE;
            ordTire = lOrdonne + (HAUTEUR_BOULE/2) - 1;
            longTire = 320 - lAbscisse + LARGEUR_BOULE;
            GLCD_SetForegroundColor(GLCD_COLOR_BLUE);
            GLCD_DrawHLine(abcTire , ordTire , longTire);
            lazerEfface[currentOrientationState] = false;
        }
        if(bouton_USER == APPUYE ){
            TIM2->CR1 |= 0x0001; //SON
        }else TIM2->CR1 &= 0x0000; //SON
        break;
    case LEFT:
        if(bouton_USER != APPUYE){
            aEteRelache[USER] = true;
            if (!lazerEfface[currentOrientationState]){
                GLCD_SetForegroundColor(GLCD_COLOR_BLACK);
                GLCD_DrawHLine(abcTire , ordTire , longTire);
                lazerEfface[currentOrientationState] = true;
            }
        }else if (aEteRelache[USER]){
            abcTire = 0;
            ordTire = lOrdonne + (HAUTEUR_BOULE/2) - 1;
            longTire = lAbscisse;
            GLCD_SetForegroundColor(GLCD_COLOR_BLUE);
            GLCD_DrawHLine(abcTire , ordTire , longTire);
            lazerEfface[currentOrientationState] = false;
        }
        if(bouton_USER == APPUYE ){
            TIM2->CR1 |= 0x0001; //SON
        }else TIM2->CR1 &= 0x0000; //SON
        break;
    case UP:
        if(bouton_USER != APPUYE){
            aEteRelache[USER] = true;
            if (!lazerEfface[currentOrientationState]){
                GLCD_SetForegroundColor(GLCD_COLOR_BLACK);
                GLCD_DrawVLine(abcTire , ordTire , longTire);
                lazerEfface[currentOrientationState] = true;
            }
        }else if (aEteRelache[USER]){
            abcTire = lAbscisse + (LARGEUR_BOULE/2) - 1;
            ordTire = 9;
            longTire = lOrdonne - (HAUTEUR_BOULE/2);
            GLCD_SetForegroundColor(GLCD_COLOR_BLUE);
            GLCD_DrawVLine(abcTire , ordTire , longTire);
            lazerEfface[currentOrientationState] = false;
        }
        if(bouton_USER == APPUYE ){
            TIM2->CR1 |= 0x0001; //SON
        }else TIM2->CR1 &= 0x0000; //SON
        break;
    case DOWN:
        if(bouton_USER != APPUYE){
            aEteRelache[USER] = true;
            if (!lazerEfface[currentOrientationState]){
                GLCD_SetForegroundColor(GLCD_COLOR_BLACK);
                GLCD_DrawVLine(abcTire , ordTire , longTire);
                lazerEfface[currentOrientationState] = true;
            }
        }else if (aEteRelache[USER]){
            abcTire = lAbscisse + (LARGEUR_BOULE/2) - 1;
            ordTire = lOrdonne + HAUTEUR_BOULE;
            longTire = 240 - (lOrdonne + HAUTEUR_BOULE);
            GLCD_SetForegroundColor(GLCD_COLOR_BLUE);
            GLCD_DrawVLine(abcTire , ordTire , longTire);
            lazerEfface[currentOrientationState] = false;
        }
        if(bouton_USER == APPUYE ){
            TIM2->CR1 |= 0x0001; //SON
        }else TIM2->CR1 &= 0x0000; //SON
        break;
}
```

Etape 5 :

Structure pour les énnemis dans un nouveau fichier ce qui me permettra d’en

```c
#include "ext_globales.h"
#include "GLCD_Config.h"
//#include "ennemi.h"

struct Ennemi{
		int abscisse , ordonne , abcissePrecedent , ordonnePrecedent , angle;
		bool alive;
};
```

créer beaucoup et facilement.

Définitions des nouvelles images :

```c
//IMAGE DE L'ENNEMI
const unsigned short ennemi[16*16] = {
		0xFDFB,0xFE3C,0x2104,0x0081,0xFDDA,0xFE3C,0x0882,0x2104,0x2104,0x10C2,0xFE3C,0xFDFB,0x0881,0x10C3,0xFE3C,0
		0xFE1C,0xFE7F,0x0000,0x0000,0xFDFC,0xFE7E,0x0000,0x0000,0x0000,0x0000,0xFE7F,0xFE3D,0x0000,0x0000,0xFE7F,0
		0x20E4,0x0000,0xFD38,0xFDBB,0x0000,0x0000,0xFDBA,0xFD39,0xFD39,0xFD7A,0x0000,0x0000,0xFDBB,0xFD79,0x0000,0
		0x18C3,0x0000,0xFDFC,0xFEBF,0x0000,0x0000,0xFE9F,0xFDFC,0xFE1C,0xFE5E,0x0000,0x0000,0xFE9F,0xFE5D,0x0000,0
		0x18E3,0x0000,0x0000,0x0000,0xFD59,0xFDBB,0x0000,0x0000,0x0000,0x0000,0xFDDB,0xFD9A,0x0000,0x0000,0x0000,0
		0x0041,0x0000,0x0000,0x0000,0xFD7A,0xFDFC,0x0000,0x0000,0x0000,0x0000,0xFE1C,0xFDBB,0x0000,0x0000,0x0000,0
		0xFDBA,0xFD59,0xFD59,0xFD59,0xFD59,0xFD79,0xFD59,0xFD59,0xFD59,0xFD59,0xFD59,0xFD59,0xFD59,0xFD59,0xFD59,0
		0xFDBA,0xFD59,0xFD59,0xFD59,0xFDFC,0xFDFC,0xFD79,0xFD59,0xFD59,0xFD79,0xFDFC,0xFDFC,0xFD79,0xFD59,0xFD59,0
		0xFDBA,0xFD59,0xFD59,0xFDDC,0x0000,0x0000,0xFDDB,0xFD59,0xFD59,0xFDBB,0x0000,0x0000,0xFDDC,0xFD59,0xFD59,0
		0xFE3C,0xFDFC,0xFD59,0xFDFC,0x0000,0x0000,0xFDDC,0xFD59,0xFD59,0xFDBB,0x0000,0x0000,0xFDFC,0xFD59,0xFDFC,0
		0x10A2,0x0000,0xFD59,0xFD7A,0xFD7A,0xFD7A,0xFD59,0xFD59,0xFD59,0xFD59,0xFD7A,0xFD7A,0xFD79,0xFD9A,0x0000,0
		0x18C3,0x0000,0xFDBB,0xFDDB,0xFD79,0xFD59,0xFD59,0xFD59,0xFD59,0xFD59,0xFD59,0xFD79,0xFDDB,0xFE1C,0x0000,0
		0x18E3,0x0000,0x0000,0x0000,0xFD79,0xFD79,0xFD59,0xFD59,0xFD59,0xFD59,0xFD59,0xFDBB,0x0000,0x0000,0x0000,0
		0x18E3,0x0000,0x0000,0x0000,0xFDDB,0xFDDB,0xFD59,0xFD59,0xFD59,0xFD59,0xFDBB,0xFE1C,0x0000,0x0000,0x0000,0
		0x18E3,0x0000,0x0000,0x0000,0x0000,0x0000,0xFDDC,0xFD59,0xFD59,0xFDBB,0x0000,0x0000,0x0000,0x0000,0x0000,0
		0x2124,0x0841,0x0841,0x0841,0x0841,0x0000,0xFDFC,0xFD79,0xFD79,0xFDDB,0x0000,0x0841,0x0841,0x0841,0x0841,0
};

//IMAGE DE LA RESERVE
const unsigned short reserve[16*16] = {
		0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0xC679,0xBE18,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0
		0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x3000,0x1000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0
		0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x4000,0xF000,0xF000,0x1800,0x0000,0x0000,0x0000,0x0000,0x0000,0
		0x0000,0x0000,0x0000,0x0000,0x0000,0x5041,0xF800,0xF800,0xF800,0xE800,0x2800,0x0000,0x0000,0x0000,0x0000,0
		0x0000,0x0000,0x0000,0x0000,0x5861,0xE800,0xF800,0xF800,0xF800,0xF800,0xE000,0x3800,0x0000,0x0000,0x0000,0
		0x0000,0x0000,0x0000,0x6082,0xE800,0xF800,0xF800,0xF800,0xF800,0xF800,0xF800,0xD800,0x4800,0x0000,0x0000,0
		0x0000,0x0000,0x6882,0xE000,0xF800,0xF800,0xF800,0xF800,0xF800,0xF800,0xF800,0xF800,0xD000,0x5800,0x0000,0
		0x0000,0x0000,0x8000,0xF800,0xF800,0xF800,0xF800,0xF800,0xF800,0xF800,0xF800,0xF800,0xE800,0x9000,0x0000,0
		0xA596,0x7082,0xE800,0xF800,0xF800,0xF800,0xF800,0xF800,0xF800,0xF800,0xF800,0xF800,0xF800,0xD000,0x7020,0
		0x7C71,0x7800,0xF924,0xF8A2,0xF800,0xF800,0xF800,0xF800,0xF800,0xF800,0xF800,0xF800,0xF800,0xE800,0x8800,0
		0x8492,0x7800,0xFC71,0xF9E7,0xF800,0xF800,0xF800,0xF800,0xF800,0xF800,0xF800,0xF800,0xF800,0xE000,0x8800,0
		0x0000,0x7800,0xF9A6,0xFE38,0xF945,0xF800,0xF800,0xF800,0xF800,0xF800,0xF800,0xF800,0xF800,0xE800,0x9000,0
		0x0000,0x0000,0x9000,0xF986,0xFEBA,0xFC30,0xF841,0x0000,0x0000,0xF800,0xF800,0xF800,0xF800,0xC000,0x0904,0
		0x0000,0x0000,0x0000,0x9800,0xF800,0xF800,0x0000,0xF7FF,0x0000,0x0000,0xF800,0xF800,0xD000,0x00C3,0x0000,0
		0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0xEFBE,0x0000,0x0000,0x0000,0x0000,0x0000,0x00C3,0x0000,0x0000,0
		0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0
};

//IMAGE NOIR
const unsigned short noir[16*16] = {
		0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0
		0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0
		0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0
		0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0
		0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0
		0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0
		0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0
		0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0
		0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0
		0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0
		0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0
		0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0
		0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0
		0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0
		0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0
		0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0
};
```

```
/*------------------------------------------------------------------------
  Réapparition Alien
 *-----------------------------------------------------------------------*/
void calculPosEnFonctionDelAngle(int unCompteur){
    switch(listeEnnemis[unCompteur].angle){
        case hautGauche:
            listeEnnemis[unCompteur].abscisse = 0;
            listeEnnemis[unCompteur].ordonne = 10;
            break;
        case hautDroit:
            listeEnnemis[unCompteur].abscisse = 320 - LARGEUR_BOULE;
            listeEnnemis[unCompteur].ordonne = 10;
            break;
        case basGauche:
            listeEnnemis[unCompteur].abscisse = 0;
            listeEnnemis[unCompteur].ordonne = 240 - HAUTEUR_BOULE;
            break;
        case basDroit:
            listeEnnemis[unCompteur].abscisse = 320 - LARGEUR_BOULE;
            listeEnnemis[unCompteur].ordonne = 240 - HAUTEUR_BOULE;
            break;
    }
}

void naissanceAlien(void){
    int nbNaissance = 0; //
    int angleProche[2] = {0 , 0};
    double lesDistances[4];
    lesDistances[hautGauche] = sqrt(pow((0 - lAbscisse) , 2) + pow(9 - lOrdonne , 2));
    lesDistances[hautDroit] = sqrt(pow(((320 - LARGEUR_BOULE) - lAbscisse) , 2) + pow(9 - lOrdonne , 2));
    lesDistances[basGauche] = sqrt(pow((0 - lAbscisse) , 2) + pow(((240 - HAUTEUR_BOULE) - lOrdonne) , 2));
    lesDistances[basDroit] = sqrt(pow((320 - LARGEUR_BOULE) - lAbscisse , 2) + pow(((240 - HAUTEUR_BOULE) - lOrdonne) , 2));
    for (compteur[1] = 1 ; compteur[1] < 3 ; compteur[1]++){
        if (lesDistances[angleProche[0]] > lesDistances[compteur[1]]) angleProche[0] = compteur[1];
    }
    if (angleProche[0] == 0)
    {
        angleProche[1]++;
    }
    for (compteur[1] = 1 ; compteur[1] < 4 ; compteur[1]++){
        if (lesDistances[angleProche[1]] > lesDistances[compteur[1]] && compteur[1] != angleProche[0]) angleProche[1] = compteur[1];
    }
    for (compteur[1] = 0 ; compteur[1] < ENNEMIS ; compteur[1]++){
        if (nbNaissance < 2 && !listeEnnemis[compteur[1]].alive) //nbNaissance est un indice donc nbNaissance = 1 équivaut à 2 naissances
        {
            listeEnnemis[compteur[1]].angle = angleProche[nbNaissance];
            calculPosEnFonctionDelAngle(compteur[1]);
            listeEnnemis[compteur[1]].alive = true;
            nbNaissance++;
        }
    }
}
```

Voici 2 fonctions appelés à la mort d'un alien, l'une (naissanecAlien) calcule la distance la lus proche grâce à la librairie math.h et réanime les alien mort par défaut ou déjà tué, et l'autre (calculPosEnFonctionDelAngle) calcule la position de ceux-ci en fonction de leur angle d'apparition, cette fonction est par ailleur aussi appelé lors de la première apparition d'un alien avec le paramètre « hazard », que nous verrons dans une interruptions tout à l'heure.

Les conditions de déplacements sont mise à jour pour que le joueur ne puisse pas toucher la réserve

```
/*-------------------------------------------------------------------------
  Vérification des conditions avant le déplacements du joueur
 *-------------------------------------------------------------------------*/
bool verificationDeplacementsJoueur(int orientation){
    switch(currentDirectionState){
        case RIGHT:
            if (bouton_USER == APPUYE || !effacementJoueur) return false;
            else if (lAbscisse == 134 && lOrdonne < 126 && lOrdonne > 94) return false;
            else if (lAbscisse != (320 - LARGEUR_BOULE)) return true;
            break;
        case LEFT:
            if (bouton_USER == APPUYE || !effacementJoueur) return false;
            else if (lAbscisse == 166 && lOrdonne < 126 && lOrdonne > 94) return false;
            if (lAbscisse != 0 && bouton_USER != APPUYE) return true;
            break;
        case UP:
            if (bouton_USER == APPUYE || !effacementJoueur) return false;
            else if (lOrdonne == 126 && lAbscisse < 166 && lAbscisse > 134) return false;
            else if (false) return false;
            if (lOrdonne != 9 && bouton_USER != APPUYE) return true;
            break;
        case DOWN:
            if (bouton_USER == APPUYE || !effacementJoueur) return false;
            else if (lOrdonne == 94 && lAbscisse < 166 && lAbscisse > 134) return false;
            if (lOrdonne != (240 - LARGEUR_BOULE) && bouton_USER != APPUYE) return true;
            break;
    }
    return false;
}
```

Un nouveau timer pour le déplacement des énnemis

```
void cfgTimer3(void){
 RCC->APB1ENR |= (1 << 1);
 TIM3->PSC = 29;  //On choisi un PSC à 29 sinon l'arr sera trop grand
 TIM3->ARR = 60000; //72/30=2.4Mhz = 2.4*10^6Hz/40 = 60000
 TIM3->DIER |= UIE;
 TIM3->CR1 |= 0x0001;
 SETENA0 |= (1<<29);
}
```

Le calcul de l’angle d’apparition du premier alien est fait dans l’interruption du timer 1 :

```
//APPARITION ANGLE AU HAZARD
if (hazard == 3)
{
    hazard = 0;
}else hazard++;

if (currentDirectionState != NONE && !premiereApparition)
{
    premiereApparition = true;
    listeEnnemis[0].angle = hazard;
    calculPosEnFonctionDelAngle(0);
    listeEnnemis[0].alive = true;
}
```

Les ennemis vont automatiquement vers la réserve en ordonné puis en abscisse (mais je trouve ça vraiment moins beau qu'en diagonale)

```
void TIM3_IRQHandler (void){
    if(TIM3->SR & UIF){
        TIM3->SR &= ~UIF; // on baisse le drapeau
        //DEPLACEMENTS ENNEMIS
        for (compteur[5] = 0 ; compteur[5] < ENNEMIS ; compteur[5]++){
            if (listeEnnemis[compteur[5]].alive)
            {
                if (listeEnnemis[compteur[5]].ordonne < 110){
                        listeEnnemis[compteur[5]].ordonnePrecedent = listeEnnemis[compteur[5]].ordonne;
                        listeEnnemis[compteur[5]].ordonne++;
                }else if(listeEnnemis[compteur[5]].ordonne > 110){
                        listeEnnemis[compteur[5]].ordonnePrecedent = listeEnnemis[compteur[5]].ordonne;
                        listeEnnemis[compteur[5]].ordonne--;
                }else if (listeEnnemis[compteur[5]].ordonne == 110){
                    if (listeEnnemis[compteur[5]].abscisse < 134){
                            listeEnnemis[compteur[5]].abcissePrecedent = listeEnnemis[compteur[5]].abscisse;
                            listeEnnemis[compteur[5]].abscisse++;
                    }else if(listeEnnemis[compteur[5]].abscisse > 134){
                            listeEnnemis[compteur[5]].abcissePrecedent = listeEnnemis[compteur[5]].abscisse;
                            listeEnnemis[compteur[5]].abscisse--;
                    }
                }
            }
        }
    }
}
```

La fonction de collision avec l'alien compare tout simplement les coordonnées du joueur avec ceux de l'alien dont l'indice doit être placé en paramètre :

```
/*-----------------------------------------------------------------------------
  Coordonnées en commun (collision avec l'Ã©nnemi)
 *-----------------------------------------------------------------------------*/
bool positionCommune(int unCompteur){
    for (compteur[1] = 0; compteur[1] < 16 ; compteur[1]++){
        for (compteur[2] = 0; compteur[2] < 16 ; compteur[2]++){
            if (listeEnnemis[unCompteur].abscisse + compteur[1] == lAbscisse + compteur[2]){
                for (compteur[3] = 0; compteur[3] < 16 ; compteur[3]++){
                    for (compteur[4] = 0; compteur[4] < 16; compteur[4]++){
                        if (listeEnnemis[unCompteur].ordonne + compteur[3] == lOrdonne + compteur[4]) return true;
                    }
                }
            }
        }
    }
    return false;
}
```

La fonction pour savoir si un ennemi est touché compare aussi les coordonnés :

```
/*-----------------------------------------------------------------------------
  A t'on touché un énnemi ? TRUE FALSE
 *-----------------------------------------------------------------------------*/
bool touche(int unCompteur){
    switch(currentOrientationState){
        case RIGHT:
            if ((listeEnnemis[unCompteur].abscisse + LARGEUR_BOULE) > lAbscisse){
                    if ((lOrdonne + 8 <= listeEnnemis[unCompteur].ordonne + 16 && lOrdonne + 8 >= listeEnnemis[unCompteur].ordonne)) return true;
            }
            break;
        case LEFT:
            if ((listeEnnemis[unCompteur].abscisse + 16) < lAbscisse){
                for (compteur[1] = 0; compteur[1] < 16 ; compteur[1]++){
                    if (listeEnnemis[unCompteur].ordonne + 8 == lOrdonne + compteur[1]) return true;
                }
            }
            break;
        case UP:
            if (listeEnnemis[unCompteur].ordonne < lOrdonne + 16){
                for (compteur[1] = 0; compteur[1] < 16 ; compteur[1]++){
                    if (listeEnnemis[unCompteur].abscisse + 8 == lAbscisse + compteur[1]) return true;
                }
            }
            break;
        case DOWN:
            if (listeEnnemis[unCompteur].ordonne > lOrdonne + HAUTEUR_BOULE){
                for (compteur[1] = 0; compteur[1] < 16 ; compteur[1]++){
                    if (listeEnnemis[unCompteur].abscisse + 8 == lAbscisse + compteur[1]) return true;
                }
            }
            break;
    }
    return false;
}
```

Cette fois le procédé est différent au lieu de faire un for je compare les valeurs minimum et maximum (j'aurais pu le faire partout mais j'ai manqué de temps, le dépôt est dans 20 minutes)

```
/*-------------------------------------------------------------------------
  Coordonnées touchés pour la réserve !
 *-------------------------------------------------------------------------*/
bool toucheReserve(){
    switch(currentOrientationState){
        case RIGHT:
            if (150 >= lAbscisse && lOrdonne + 8 <= 110 + 16 && lOrdonne + 8 >= 110) return true;
                /*for (compteur[1] = 0; compteur[1] < 16 ; compteur[1]++)
                {
                    if (110 + 8 == lOrdonne + compteur[1])
                    {
                        return true;
                    }
                }*/
            break;
        case LEFT:
            if (166 <= lAbscisse && lOrdonne + 8 <= 110 + 16 && lOrdonne + 8 >= 110) return true;
                /*for (compteur[1] = 0; compteur[1] < 16 ; compteur[1]++)
                {
                    if (110 + 8 == lOrdonne + compteur[1])
                    {
                        return true;
                    }
                }*/
            break;
        case UP:
            if (110 < lOrdonne && lAbscisse + 8 <= 150 + 16 && lAbscisse + 8 >= 150) return true;
                /*for (compteur[1] = 0; compteur[1] < 16 ; compteur[1]++)
                {
                    if (150 + 8 == lAbscisse + compteur[1])
                    {
                        return true;
                    }
                }*/
            break;
        case DOWN:
            if (110 > lOrdonne && lAbscisse + 8 <= 150 + 16 && lAbscisse + 8 >= 150) return true;
                /*for (compteur[1] = 0; compteur[1] < 16 ; compteur[1]++)
                {
                    if (150 + 8 == lAbscisse + compteur[1])
                    {
                        return true;
                    }
                }*/
            break;
    }
    return false;
}
```

On a ensuite l'effacement des traces des ennemis qui fonctionnent sur le même principe que ceux du joueur :

```
/*-------------------------------------------------------------------------
  Effacement des traces de déplacements de l'énnemi
 *-------------------------------------------------------------------------*/
void effacementEnnemi(void){
//Effacement derrière l'énnemi lorsqu'il se déplace en abcisse
    if(listeEnnemis[compteur[0]].abscisse > listeEnnemis[compteur[0]].abcissePrecedent){
        GLCD_SetForegroundColor(GLCD_COLOR_BLACK);
        GLCD_DrawVLine(listeEnnemis[compteur[0]].abscisse - 1 , listeEnnemis[compteur[0]].ordonne , 16);
    }else if(listeEnnemis[compteur[0]].abscisse < listeEnnemis[compteur[0]].abcissePrecedent){
        GLCD_SetForegroundColor(GLCD_COLOR_BLACK);
        GLCD_DrawVLine((listeEnnemis[compteur[0]].abscisse + 16) , listeEnnemis[compteur[0]].ordonne , 16);
    }
    //Effacement derrière l'ennemi lorsqu'il se déplace en ordonne
    if(listeEnnemis[compteur[0]].ordonne > listeEnnemis[compteur[0]].ordonnePrecedent){
        GLCD_SetForegroundColor(GLCD_COLOR_BLACK);
        GLCD_DrawHLine(listeEnnemis[compteur[0]].abscisse , (listeEnnemis[compteur[0]].ordonne - 1) , 16);
    }else if(listeEnnemis[compteur[0]].ordonne < listeEnnemis[compteur[0]].ordonnePrecedent){
        GLCD_SetForegroundColor(GLCD_COLOR_BLACK);
        GLCD_DrawHLine(listeEnnemis[compteur[0]].abscisse , (listeEnnemis[compteur[0]].ordonne + 16) , 16);
    }
}
```

Le même code étant utilisé plusieurs fois j'ai pensé bon de créer une fonction gérant l'atteinte des cibles, qui lance les instructions en fonction des évènements :

```
/*-------------------------------------------------------------------------
  Gestion des évènements si on a touché un élément avec le laser
 *-------------------------------------------------------------------------*/
void touchePrincipal(void) {
    if (toucheReserve()){
        GLCD_SetForegroundColor(GLCD_COLOR_WHITE);
        GLCD_DrawString(30 , 110 , "touche Reserve");
    }
    for (compteur[0] = 0; compteur[0] < ENNEMIS; compteur[0]++){
        if(listeEnnemis[compteur[0]].alive && touche(compteur[0])){
            listeEnnemis[compteur[0]].alive = false;
            effacementEnnemi();
            GLCD_DrawBitmap(listeEnnemis[compteur[0]].abscisse , listeEnnemis[compteur[0]].ordonne , 16 , 16 , (const unsigned char *)noir);
            listeEnnemis[compteur[0]].ordonne = 0;
            listeEnnemis[compteur[0]].abscisse = 0;
            fonctionToucheExecutee = true;
            naissanceAlien();
        }
    }
}
```

En plus d'émettre un son, on lance à chaque fois la fonction lié à l'atteinte des cibles si elle n'a pas déjà été lancée (sinon les alien pourraient naître 15 fois avant que l'utilisateur ne relâche le bouton, et il n'aurait même pas à rappuyer pour tuer d'autres alien qui passeraient par là)

```
if(bouton_USER == APPUYE ){
    TIM2->CR1 |= 0x0001; //SON
    if (!fonctionToucheExecutee) touchePrincipal();
}else TIM2->CR1 &= 0x0000; //SON
break;
```

Finalement dans le main on dessine tous les éléments :

```
for (compteur[0] = 0 ; compteur[0] < ENNEMIS ; compteur[0]++)
{
    if (listeEnnemis[compteur[0]].alive)
    {
        GLCD_DrawBitmap(listeEnnemis[compteur[0]].abscisse , listeEnnemis[compteur[0]].ordonne , 16 , 16 , (const unsigned char *)ennemi);
    }
}

GLCD_DrawBitmap(150 , 110 , 16 , 16 , (const unsigned char *)reserve);
GLCD_DrawBitmap(lAbscisse , lOrdonne , LARGEUR_BOULE , HAUTEUR_BOULE , (const unsigned char *)vaisseau[currentOrientationState]);
```

A la fin du main, détecte les collisions :

```
for (compteur[0] = 0 ; compteur[0] < ENNEMIS ; compteur[0]++){
    if (listeEnnemis[compteur[0]].alive){
        effacementEnnemi();
        if (positionCommune(compteur[0]))//Si un monstre touche le joueur
        {
            GLCD_SetForegroundColor(GLCD_COLOR_WHITE);
            GLCD_DrawString(30 , 110 , "Collision");
            currentGameState = gameOver;
        }

    }
}
```

Conclusion :

Mon problème dans ce projet fut le temps, je le savais dès le départ j'ai donc décidé d'implémenter les fonctionnalités demandés en premier et le plus vite possible, si j'avais eu plus de temps j'aurais découpé les fichiers (ou si le code avait fait plus de 1000 lignes), optimisé le code, puis implémenté le bonus et enfin commenté plus en détail le code.

Ce que j'aurais voulu faire si j'avais eu plus de temps :

1. faire un lazer multicolor (incrémenter la valeur de la couleur dans le timer, ça ne prend que quelques lignes mais bon).
2. Proposer la solution alternative des matrices en commentaire
3. faire la musique de star wars avec un 4ème timer lorsque l'on reste appuyé sur le bouton user
4. refaire les images qui sont à peu près toutes moches
5. faire des bonus/combos pour le score
6. faire des alien qui décideraient au hazard de se diriger vers la réserve OU vers le joueur

Et pour finir merci pour cette année, je me suis bien amusé j'ai appris beaucoup de choses qui m'ont plu et j'ai pu faire ce que j'aime.

Références à la documentation :

Broche	Bouton associé
UP	PG15
DOWN	PD3
RIGHT	PG13
LEFT	PG14
SELECT	PG7
TEMPER (encore une faute dans mon code)	PC13
WAKEUP	PA0
USER	PG8

IDR : port *input data register*
ODR : port *output data register*
CRL : port *configuration low* (de 0 à 7)
CRH : port *configuration high* (de 8 à 15)
PSC (prescaler) : effectue une première division afin de ralentir la fréquence de comptage de l'horloge afin de ralentir le comptage du timer
ARR : autoreload (contient la valeur à laquelle le compteur va déborder)