

# T.P. 3 – Corrigé

## Space Invaders (partie 6)

### Étape 1

```

IsOutOfX      ; Sauvegarde les registres.
               move.l  d1,-(a7)

               ; Si l'abscisse est négative, le bitmap sort de l'écran.
               ; On renvoie true.
               tst.w   d1
               bmi    \true

               ; Abscisse à la droite du bitmap -> D1.W
               add.w  WIDTH(a0),d1

               ; Si l'abscisse à la droite du bitmap
               ; est supérieure à la largeur de l'écran,
               ; le bitmap sort de l'écran.
               ; On renvoie true.
               cmp.w  #VIDEO_WIDTH,d1
               bhi   \true

               ; Sinon, le bitmap ne sort pas de l'écran.
               ; On renvoie false.

\false        ; Sortie qui renvoie false (Z = 0).
               move.l (a7)+,d1
               andi.b  #%11111011,CCR
               rts

\true         ; Sortie qui renvoie true (Z = 1).
               move.l (a7)+,d1
               ori.b  #%00000100,CCR
               rts

```

```

IsOutOfY          ; Sauvegarde les registres.
                  move.l  d2,-(a7)

                  ; Si l'ordonnée est négative, le bitmap sort de l'écran.
                  ; On renvoie true.
                  tst.w   d2
                  bmi    \true

                  ; Ordonnée juste sous le bitmap -> D2.W
                  add.w   HEIGHT(a0),d2

                  ; Si l'ordonnée juste sous le bitmap
                  ; est supérieure à la hauteur de l'écran,
                  ; le bitmap sort de l'écran.
                  ; On renvoie true.
                  cmp.w   #VIDEO_HEIGHT,d2
                  bhi    \true

                  ; Sinon, le bitmap ne sort pas de l'écran.
                  ; On renvoie false.

\false           ; Sortie qui renvoie false (Z = 0).
                  move.l  (a7)+,d2
                  andi.b  #%11111011,ccr
                  rts

\true            ; Sortie qui renvoie true (Z = 1).
                  move.l  (a7)+,d2
                  ori.b   #%00000100,ccr
                  rts

```

```

IsOutOfScreen     ; Si le bitmap sort de l'axe des abscisses, on renvoie true.
                  jsr    IsOutOfX
                  beq    \quit

                  ; Si le bitmap sort de l'axe des ordonnées, on renvoie true.
                  ; Sinon on renvoie false.
                  jsr    IsOutOfY

\quit             rts

```

**Étape 2**

```

MoveSprite      ; Sauvegarde les registres.
                movem.l d1/d2/a0,-(a7)

                ; Nouvelle abscisse du sprite -> D1.W
                ; Nouvelle ordonnée du sprite -> D2.W
                add.w X(a1),d1
                add.w Y(a1),d2

                ; Adresse du bitmap 1 du sprite -> A0.L
                movea.l BITMAP1(a1),a0

                ; Si les nouvelles coordonnées font sortir le bitmap de l'écran,
                ; on renvoie false (sans modifier les coordonnées du sprite).
                jsr IsOutOfScreen
                beq \false

                ; Sinon, on modifie les coordonnées du sprite.
                move.w d1,X(a1)
                move.w d2,Y(a1)

                ; Et on renvoie true.

\true           ; Sortie qui renvoie true (Z = 1).
                ori.b  #%00000100,CCR
                bra \quit

\false         ; Sortie qui renvoie false (Z = 0).
                andi.b #%11111011,CCR
\quit         movem.l (a7)+,d1/d2/a0
                rts

```

**Étape 3**

```

; Touches du clavier
; -----
SPACE_KEY    equ    $420
LEFT_KEY     equ    $46f
UP_KEY       equ    $470
RIGHT_KEY    equ    $471
DOWN_KEY     equ    $472

```

```

MoveSpriteKeyboard ; Sauvegarde les registres.
                  movem.l d1/d2,-(a7)

                  ; Initialise le mouvement relatif à zéro.
                  clr.w   d1
                  clr.w   d2

\up               ; Si la touche "haut" est pressée,
                  ; décremente D2.W (déplacement d'un pixel vers le haut).
                  tst.b   UP_KEY
                  beq     \down
                  sub.w   #1,d2

\down            ; Si la touche "bas" est pressée,
                  ; incrémente D2.W (déplacement d'un pixel vers le bas).
                  tst.b   DOWN_KEY
                  beq     \right
                  add.w   #1,d2

\right           ; Si la touche "droite" est pressée,
                  ; incrémente D1.W (déplacement d'un pixel vers la droite).
                  tst.b   RIGHT_KEY
                  beq     \left
                  add.w   #1,d1

\left            ; Si la touche "gauche" est pressée,
                  ; décremente D1.W (déplacement d'un pixel vers la gauche).
                  tst.b   LEFT_KEY
                  beq     \next
                  sub.w   #1,d1

\next           ; Déplace le sprite (en fonction de D1.W et de D2.W).
                  jsr     MoveSprite

                  ; Restaure les registres puis sortie.
                  movem.l (a7)+,d1/d2
                  rts

```