

# TP 8 – Corrigé

## Space Invaders (partie 11)

### Étape 1

```

InitInvaders      ; Sauvegarde les registres.
                  movem.l d1/d2/a0-a2,-(a7)

                  ; 1re ligne d'envahisseurs.
                  move.w InvaderX,d1
                  move.w InvaderY,d2
                  lea Invaders,a0
                  lea InvaderC1_Bitmap,a1
                  lea InvaderC2_Bitmap,a2
                  jsr InitInvaderLine

                  ; 2e ligne d'envahisseurs.
                  add.w #32,d2
                  adda.l #SIZE_OF_SPRITE*INVADER_PER_LINE,a0
                  lea InvaderB1_Bitmap,a1
                  lea InvaderB2_Bitmap,a2
                  jsr InitInvaderLine

                  ; 3e ligne d'envahisseurs.
                  add.w #32,d2
                  adda.l #SIZE_OF_SPRITE*INVADER_PER_LINE,a0
                  jsr InitInvaderLine

                  ; 4e ligne d'envahisseurs.
                  add.w #32,d2
                  adda.l #SIZE_OF_SPRITE*INVADER_PER_LINE,a0
                  lea InvaderA1_Bitmap,a1
                  lea InvaderA2_Bitmap,a2
                  jsr InitInvaderLine

                  ; 5e ligne d'envahisseurs.
                  add.w #32,d2
                  adda.l #SIZE_OF_SPRITE*INVADER_PER_LINE,a0
                  jsr InitInvaderLine

                  ; Restaure les registres puis sortie.
                  movem.l (a7)+,d1/d2/a0-a2
                  rts

```

```

SwapBitmap        ; Échange les contenus de BITMAP1 et BITMAP2.
                  move.l BITMAP1(a1),-(a7)
                  move.l BITMAP2(a1),BITMAP1(a1)
                  move.l (a7)+,BITMAP2(a1)

                  ; Sortie du sous-programme.
                  rts

```

```
MoveAllInvaders    ; Sauvegarde les registres.
                   movem.l d1/d2/a1/d7,-(a7)

                   ; Récupère les déplacements relatifs dans D1.W et D2.W.
                   ; (La position globale est mise à jour.)
                   jsr      GetInvaderStep

                   ; Fait pointer A1.L sur le premier envahisseur.
                   lea      Invaders,a1

                   ; Nombre d'envahisseurs - 1 (car DBRA) -> D7.W
                   move.w   #INVADER_COUNT-1,d7

\loop              ; Si l'envahisseur n'est pas affiché, on passe au suivant.
                   cmp.w   #HIDE,STATE(a1)
                   beq      \continue

                   ; Déplace l'envahisseur et permute ses bitmaps.
                   jsr      MoveSprite
                   jsr      SwapBitmap

\continue         ; Pointe sur le prochain envahisseur.
                   adda.l   #SIZE_OF_SPRITE,a1

                   ; On reboucle tant qu'il reste des envahisseurs.
                   dbra     d7,\loop

\quit             ; Restaure les registres puis sortie.
                   movem.l (a7)+,d1/d2/a1/d7
                   rts
```

**Étape 2**

```

; =====
; Données
; =====

; ...

InvaderCount    dc.w    INVADER_COUNT

; ...

```

```

DestroyInvaders    ; Sauvegarde les registres.
                   movem.l d7/a1/a2,-(a7)

                   ; Fait pointer A1.L sur les envahisseurs.
                   ; Fait pointer A2.L sur le tir du vaisseau.
                   lea    Invaders,a1
                   lea    ShipShot,a2

                   ; Nombre d'itérations - 1 (car DBRA) -> D7.W
                   move.w #INVADER_COUNT-1,d7

\loop              ; Si le tir n'entre pas en collision
                   ; avec l'envahisseur, on passe au suivant.
                   jsr    IsSpriteColliding
                   bne    \next

\colliding         ; S'il y a une collision,
                   ; on efface le tir et l'envahisseur.
                   ; Puis on décrémente le nombre d'envahisseurs.
                   move.w #HIDE,STATE(a1)
                   move.w #HIDE,STATE(a2)
                   subq.w #1,InvaderCount

\next              ; Passe à l'envahisseur suivant.
                   adda.l #SIZE_OF_SPRITE,a1
                   dbra   d7,\loop

\quit              ; Restaure les registres.
                   movem.l (a7)+,d7/a1/a2
                   rts

```

**Étape 3**

```

; =====
; Données
; =====

; ...

InvaderSpeed      dc.w      8                      ; Vitesse (1 -> 8)
SpeedLevels       dc.w      1,5,10,15,20,25,35,50    ; Paliers de vitesse

; ...

```

```

MoveInvaders      ; Décrémente la variable "skip",
                  ; et ne fait rien si elle n'est pas nulle.
subq.w #1,\skip
bne      \quit

                  ; Réinitialise "skip" à sa valeur maximale.
move.w  InvaderSpeed,\skip

                  ; Appel de MoveAllInvaders.
jsr     MoveAllInvaders

\quit             ; Sortie du sous programme.
rts

\skip             ; Compteur d'affichage des envahisseurs
dc.w      1

```

```

SpeedInvaderUp    ; Sauvegarde les registres.
movem.l d0/a0,-(a7)

                  ; Initialise le compteur de vitesse.
clr.w  InvaderSpeed

                  ; Nombre d'envahisseurs en cours d'affichage -> D0.W
move.w  InvaderCount,d0

                  ; Fait pointer A0.L sur le tableau des paliers de vitesse.
lea     SpeedLevels,a0

\loop             ; Incrémente le compteur de vitesse.
addq.w #1,InvaderSpeed

                  ; Compare le nombre d'envahisseurs à un palier du tableau.
                  ; Si le nombre d'envahisseurs est plus grand,
                  ; on passe au palier suivant.
cmp.w   (a0)+,d0
bhi     \loop

                  ; Restaure les registres puis sortie.
movem.l (a7)+,d0/a0
rts

```