T.P. 4 – Corrigé
Space Invaders (partie 7)

Étape 1

```
GetRectangle        ; Sauvegarde les registres.
                    move.l  a0,-(a7)

                    ; Abscisse du point supérieur gauche -> D1.W
                    move.w  X(a0),d1

                    ; Ordonnée du point supérieur gauche -> D2.W
                    move.w  Y(a0),d2

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

                    ; Abscisse du point inférieur droit -> D3.W
                    move.w  WIDTH(a0),d3
                    add.w   d1,d3
                    subq.w  #1,d3

                    ; Ordonnée du point inférieur droit -> D4.W
                    move.w  HEIGHT(a0),d4
                    add.w   d2,d4
                    subq.w  #1,d4

                    ; Restaure les registres puis sortie.
                    movea.l (a7)+,a0
                    rts
```

Étape 2

```
IsSpriteColliding   ; Sauvegarde les registres.
                    movem.l d1-d4/a0,-(a7)

                    ; Si les sprites ne sont pas visibles, on quitte.
                    ; Le BNE saute si Z = 0, on renvoie donc false.
                    ; On ne peut pas effectuer un BNE \false tout de suite,
                    ; car ce dernier passe par le nettoyage de la pile.
                    cmp.w   #SHOW,STATE(a1)
                    bne     \quit
                    cmp.w   #SHOW,STATE(a2)
                    bne     \quit

                    ; Coordonnées du rectangle 1 -> Pile
                    ; D1.W ->  (a7) ; x1 = Abscisse du point supérieur gauche
                    ; D2.W -> 2(a7) ; y1 = Ordonnée du point supérieur gauche
                    ; D3.W -> 4(a7) ; X1 = Abscisse du point inférieur droit
                    ; D4.W -> 6(a7) ; Y1 = Ordonnée du point inférieur droit
                    movea.l a1,a0
                    jsr     GetRectangle
                    movem.w d1-d4,-(a7)

                    ; Coordonnées du rectangle 2 -> D1-D4
                    ; D1.W = x2 = Abscisse du point supérieur gauche
                    ; D2.W = y2 = Ordonnée du point supérieur gauche
                    ; D3.W = X2 = Abscisse du point inférieur droit
                    ; D4.W = Y2 = Ordonnée du point inférieur droit
                    movea.l a2,a0
                    jsr     GetRectangle

                    ; Si x2 > X1, on renvoie false.
                    cmp.w   4(a7),d1
                    bgt     \false

                    ; Si y2 > Y1, on renvoie false.
                    cmp.w   6(a7),d2
                    bgt     \false

                    ; Si X2 < x1, on renvoie false.
                    cmp.w   (a7),d3
                    blt     \false

                    ; Si Y2 < y1, on renvoie false.
                    cmp.w   2(a7),d4
                    blt     \false

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

\false              ; Sortie qui renvoie false (Z = 0).
                    andi.b  #%11111011,ccr

\cleanStack         ; Dépile les coordonnées du rectangle 1.
                    ; (L'instruction ADDA ne modifie pas les flags.)
                    adda.l  #8,a7

\quit               ; Restaure les registres puis sortie.
                    movem.l (a7)+,d1-d4/a0
                    rts
```

Étape 3

Il faut simplement modifier la structure du sprite mobile et ajouter le bitmap `ShipShot_Bitmap`.

```
                    ; ==============================
                    ; Données
                    ; ==============================

                    ; Sprites
                    ; ------------------------------

MovingSprite        dc.w    SHOW
                    dc.w    0,152
                    dc.l    ShipShot_Bitmap
                    dc.l    0

FixedSprite         ; ...

                    ; Bitmaps
                    ; ------------------------------

InvaderA_Bitmap     ; ...

InvaderB_Bitmap     ; ...

InvaderC_Bitmap     ; ...

Ship_Bitmap         ; ...

ShipShot_Bitmap     dc.w    2,6
                    dc.b    %11000000
                    dc.b    %11000000
                    dc.b    %11000000
                    dc.b    %11000000
                    dc.b    %11000000
                    dc.b    %11000000
```