Key to Practical 9
Space Invaders (Part 2)

Step 1

```
WhiteSquare32       ; Save registers on the stack.
                    movem.l d7/a0,-(a7)

                    ; A0 points to the address of the square
                    ; ------------------------------
                    ; Horizontal Centering:
                    ; The width below is in bytes.
                    ; Full width = width of the window = BYTE_PER_LINE
                    ; Width of the square = 4 bytes (32 pixels)
                    ; Horizontal displacement in bytes
                    ; = (Full width - width of the square) / 2
                    ; ------------------------------
                    ; Vertical Centering:
                    ; The height below is in pixels.
                    ; Full height = Height of the window = VIDEO_HEIGHT
                    ; Height of the square = 32 pixels
                    ; Vertical displacement in pixels
                    ; = (Full height - Height of the square) / 2
                    ; Vertical displacement in bytes
                    ; = Vertical displacement in pixels x BYTE_PER_LINE
                    ; ------------------------------
                    ; Address of the square
                    ; = VIDEO_START + (Horizontal displacement) + (Vertical displacement)
                    lea     VIDEO_START+((BYTE_PER_LINE-4)/2)+(((VIDEO_HEIGHT-
                            32)/2)*BYTE_PER_LINE),a0

                    ; Initialize the loop counter (D7.W).
                    ; Number of iterations = Number of lines of the square (32).
                    ; D7.W = Number of iterations - 1 (DBRA).
                    move.w  #32-1,d7

\loop               ; Copy 32 white pixels into video memory
                    ; and increment the address.
                    move.l  #$ffffffff,(a0)
                    adda.l  #BYTE_PER_LINE,a0
                    dbra    d7,\loop

                    ; Restore registers from the stack and return from subroutine.
                    movem.l (a7)+,d7/a0
                    rts
```

Step 2

```
WhiteSquare128      ; Save registers on the stack.
                    movem.l d7/a0,-(a7)

                    ; A0 points to the address of the square
                    ; ----------------------------
                    ; Horizontal Centering:
                    ; The width below is in bytes.
                    ; Full width = width of the window = BYTE_PER_LINE
                    ; Width of the square = 16 bytes (128 pixels)
                    ; Horizontal displacement in bytes
                    ; = (Full width - width of the square) / 2
                    ; ----------------------------
                    ; Vertical Centering:
                    ; The height below is in pixels.
                    ; Full height = Height of the window = VIDEO_HEIGHT
                    ; Height of the square = 128 pixels
                    ; Vertical displacement in pixels
                    ; = (Full height - Height of the square) / 2
                    ; Vertical displacement in bytes
                    ; = Vertical displacement in pixels x BYTE_PER_LINE
                    ; ----------------------------
                    ; Address of the square
                    ; = VIDEO_START + (Horizontal displacement) + (Vertical displacement)
                    lea     VIDEO_START+((BYTE_PER_LINE-16)/2)+(((VIDEO_HEIGHT-
                            128)/2)*BYTE_PER_LINE),a0

                    ; Initialize the loop counter (D7.W).
                    ; Number of iterations = Number of lines of the square (128).
                    ; D7.W = Number of iterations - 1 (DBRA).
                    move.w  #128-1,d7

\loop               ; Copy 128 white pixels into video memory
                    ; and increment the address.
                    move.l  #$ffffffff,(a0)
                    move.l  #$ffffffff,4(a0)
                    move.l  #$ffffffff,8(a0)
                    move.l  #$ffffffff,12(a0)
                    adda.l  #BYTE_PER_LINE,a0
                    dbra    d7,\loop

                    ; Restore registers from the stack and return from subroutine.
                    movem.l (a7)+,d7/a0
                    rts
```

Step 3

```
WhiteLine           ; Save registers on the stack.
                    movem.l d0/a0,-(a7)

                    ; Number of iterations = Size of the line in bytes
                    ; D0.W = Number of iterations - 1 (DBRA)
                    subq.w  #1,d0

\loop               ; Copy 8 white pixels and increment the address.
                    move.b  #$ff,(a0)+
                    dbra    d0,\loop

                    ; Restore registers from the stack and return from subroutine.
                    movem.l (a7)+,d0/a0
                    rts
```

```
WhiteSquare         ; Save registers on the stack.
                    movem.l d0-d2/a0,-(a7)

                    ; D2.W = Size of the square in pixels.
                    move.w  d0,d2
                    lsl.w   #3,d2

                    ; A0 points to the video memory.
                    lea     VIDEO_START,a0

                    ; Horizontal centering.
                    ; A0 + (Full width - width of the square) / 2
                    move.w  #BYTE_PER_LINE,d1
                    sub.w   d0,d1
                    lsr.w   #1,d1
                    adda.w  d1,a0

                    ; Vertical centering.
                    ; A0 + ((Full height - Height of the square) / 2) * BYTE_PER_LINE
                    move.w  #VIDEO_HEIGHT,d1
                    sub.w   d2,d1
                    lsr.w   #1,d1
                    mulu.w  #BYTE_PER_LINE,d1
                    adda.w  d1,a0

                    ; Number of iterations = Size in pixels
                    ; D2.W = Number of iterations - 1 (DBRA)
                    subq.w  #1,d2

\loop               ; Draw the line in progress and go to next line.
                    jsr     WhiteLine
                    adda.l  #BYTE_PER_LINE,a0
                    dbra    d2,\loop

                    ; Restore registers from the stack and return from subroutine.
                    movem.l (a7)+,d0-d2/a0
                    rts
```