```
# Utilidades.s
# Archivo con distintas funciones utilizadas en Main.s
#
# Autores: Ka Fung & Christopher Gomez
# Fecha: 10-ene-2022

# Funcion: Convierte una coordenada (x, y) en una
#          direccion del Bitmap Display.
#          Toma como origen (0, 0) la esquina inferior izquierda.
# Entrada: $a0: Coordenada x.
#          $a1: Coordenada y.
# Salida:  $v0: Direccion del Bitmap Display
#               correspondiente a (x, y).
# Planificacion de registros:
# $t0: Auxiliar
coord_a_dir_bitmap:
    # Prologo
    sw   $fp, ($sp)
    move $fp,  $sp
    addi $sp,  $sp, -4

    # Formula: (32*(31 - y) + x)*4 + MAT
    li  $t0, 31
    sub $v0, $t0, $a1   # a = 31 - y
    sll $v0, $v0, 5     # a = a * 32
    add $v0, $v0, $a0   # a = a + x
    sll $v0, $v0, 2     # a = a * 4
    lw  $t0, MAT
    add $v0, $v0, $t0   # a = a + MAT

    # Epilogo
    move $sp,  $fp
    lw   $fp, ($sp)

    jr $ra

# Funcion: Retorna la dirección del Bitmap Display correspondiente
#          al pixel de arriba en el tablero de 32x32.
# Entrada: $a0: Direccion del Bitmap Display.
# Salida:  $v0: Direccion del Bitmap Display.
obtener_dir_arriba:
    # Prologo
    sw   $fp, ($sp)
    move $fp,  $sp
    addi $sp,  $sp, -4

    add $v0, $a0, -128 # DIRRETORNO = DIRACTUAL - 32*4

    # Epilogo
    move $sp,  $fp
    lw   $fp, ($sp)

    jr $ra

# Funcion: Retorna la dirección del Bitmap Display correspondiente
#          al pixel de noreste en el tablero de 32x32.
# Entrada: $a0: Direccion del Bitmap Display.
# Salida:  $v0: Direccion del Bitmap Display.
obtener_dir_noreste:
    # Prologo
    sw   $fp, ($sp)
    move $fp,  $sp
    addi $sp,  $sp, -4

    add $v0, $a0, -124 # DIRRETORNO = DIRACTUAL - 32*4 + 4

    # Epilogo
    move $sp,  $fp
    lw   $fp, ($sp)

    jr $ra

# Funcion: Retorna la dirección del Bitmap Display correspondiente
#          al pixel de derecha en el tablero de 32x32.
# Entrada: $a0: Direccion del Bitmap Display.
# Salida:  $v0: Direccion del Bitmap Display.
obtener_dir_derecha:
    # Prologo
    sw   $fp, ($sp)
    move $fp,  $sp
    addi $sp,  $sp, -4

    add $v0, $a0, 4 # DIRRETORNO = DIRACTUAL + 4

    # Epilogo
    move $sp,  $fp
    lw   $fp, ($sp)

    jr $ra

# Funcion: Retorna la dirección del Bitmap Display correspondiente
#          al pixel de sureste en el tablero de 32x32.
# Entrada: $a0: Direccion del Bitmap Display.
# Salida:  $v0: Direccion del Bitmap Display.
obtener_dir_sureste:
    # Prologo
```

```
    sw   $fp, ($sp)
    move $fp,  $sp
    addi $sp,  $sp, -4

    add $v0, $a0, 132 # DIRRETORNO = DIRACTUAL + 32*4 + 4

    # Epilogo
    move $sp,  $fp
    lw   $fp, ($sp)

    jr $ra

# Funcion: Retorna la dirección del Bitmap Display correspondiente
#          al pixel de abajo en el tablero de 32x32.
# Entrada: $a0: Direccion del Bitmap Display.
# Salida:  $v0: Direccion del Bitmap Display.
obtener_dir_abajo:
    # Prologo
    sw   $fp, ($sp)
    move $fp,  $sp
    addi $sp,  $sp, -4

    add $v0, $a0, 128 # DIRRETORNO = DIRACTUAL + 32*4

    # Epilogo
    move $sp,  $fp
    lw   $fp, ($sp)

    jr $ra

# Funcion: Retorna la dirección del Bitmap Display correspondiente
#          al pixel del suroeste en el tablero de 32x32.
# Entrada: $a0: Direccion del Bitmap Display.
# Salida:  $v0: Direccion del Bitmap Display.
obtener_dir_suroeste:
    # Prologo
    sw   $fp, ($sp)
    move $fp,  $sp
    addi $sp,  $sp, -4

    add $v0, $a0, 124 # DIRRETORNO = DIRACTUAL + 32*4 - 4

    # Epilogo
    move $sp,  $fp
    lw   $fp, ($sp)

    jr $ra

# Funcion: Retorna la dirección del Bitmap Display correspondiente
#          al pixel de izquierda en el tablero de 32x32.
# Entrada: $a0: Direccion del Bitmap Display.
# Salida:  $v0: Direccion del Bitmap Display.
obtener_dir_izquierda:
    # Prologo
    sw   $fp, ($sp)
    move $fp,  $sp
    addi $sp,  $sp, -4

    add $v0, $a0, -4 # DIRRETORNO = DIRACTUAL - 4

    # Epilogo
    move $sp,  $fp
    lw   $fp, ($sp)

    jr $ra

# Funcion: Retorna la dirección del Bitmap Display correspondiente
#          al pixel de noroeste en el tablero de 32x32.
# Entrada: $a0: Direccion del Bitmap Display.
# Salida:  $v0: Direccion del Bitmap Display.
obtener_dir_noroeste:
    # Prologo
    sw   $fp, ($sp)
    move $fp,  $sp
    addi $sp,  $sp, -4

    add $v0, $a0, -132 # DIRRETORNO = DIRACTUAL - 32*4 - 4

    # Epilogo
    move $sp,  $fp
    lw   $fp, ($sp)

    jr $ra

# Funcion: Pinta un tablero 32x32 desde una cadena de caracteres terminada
#          en nulo, donde cada caracter representa una celda, con el siguiente
#          formato:
#             El tamaño de la cadena es de 1025 bytes (32*32 + 1).
#                'G' representa gris oscuro (pared).
#                ' ' representa blanco (alimento).
#                'N' representa naranja (portal).
#                'P' representa amarillo (Pac-Man).
#                'R' representa rojo (Blinky).
#                'M' representa marron (Pinky).
#                'A' representa azul (Inky).
#                'V' representa verde (Clyde).
#          Usa los colores definidos en Main.s
```

```
#          Cuenta y actualiza a su vez la cantidad de alimento en el mapa.
# Entrada: $a0: Direccion de la cadena que contiene el tablero.
#          $a1: Direccion de contador de alimentos restantes.
#          $a2: Direccion de contador de alimentos totales.
# Salida:  $v0: negativo si ocurrio algun error.
# Planificacion de registros:
# $t0: Auxiliar.
# $t1: Direccion a escribir en el Bitmap Display.
# $t2: Color del pixel a pintar.
# $t3: colorPared.
# $t4: colorComida.
pintar_tablero:
    # Prologo
    sw   $fp,    ($sp)
    move $fp,     $sp
    addi $sp,     $sp, -4

    lw $t1, MAT
    lw $t3, colorPared
    lw $t4, colorComida
    pintar_tablero_for_pixel:
        lb  $t0, ($a0)
        beq $t0, $zero, pintar_tablero_fin
        add $a0, $a0, 1

        # Pintar tablero
        beq $t0, ' ', pintar_tablero_blanco
        beq $t0, 'G', pintar_tablero_gris
        beq $t0, 'N', pintar_tablero_naranja
        beq $t0, 'P', pintar_tablero_amarillo
        beq $t0, 'R', pintar_tablero_rojo
        beq $t0, 'M', pintar_tablero_marron
        beq $t0, 'A', pintar_tablero_azul

        # Si no es ninguno de los demas es verde
            lw $t2, colorClyde
            sw $t2, ($t1)
            j  pintar_tablero_aumentar_contador
        pintar_tablero_gris:
            sw $t3, ($t1)
            j  pintar_tablero_for_pixel_sig
        pintar_tablero_blanco:
            sw $t4, ($t1)
            j  pintar_tablero_aumentar_contador
        pintar_tablero_naranja:
            lw $t2, colorPortal
            sw $t2, ($t1)
            j  pintar_tablero_for_pixel_sig
        pintar_tablero_amarillo:
            lw $t2, colorPacman
            sw $t2, ($t1)
            j  pintar_tablero_for_pixel_sig
        pintar_tablero_rojo:
            lw $t2, colorBlinky
            sw $t2, ($t1)
            j  pintar_tablero_aumentar_contador
        pintar_tablero_marron:
            lw $t2, colorPinky
            sw $t2, ($t1)
            j  pintar_tablero_aumentar_contador
        pintar_tablero_azul:
            lw $t2, colorInky
            sw $t2, ($t1)
            j  pintar_tablero_aumentar_contador
    pintar_tablero_aumentar_contador:
        # Aumenta contador de alimentos restantes
        lw  $t0, ($a1)
        add $t0,  $t0, 1
        sw  $t0, ($a1)
    pintar_tablero_for_pixel_sig:
        add $t1, $t1, 4
        j   pintar_tablero_for_pixel
pintar_tablero_fin:
    # Copia al contador de alimentos total
    lw $t0, ($a1)
    sw $t0, ($a2)

    # Epilogo
    move $sp,     $fp
    lw   $fp,    ($sp)

    jr $ra

# Funcion: Escoge una palabra pseudo-aleatoriamente del arreglo de entrada.
# Entrada: $a0: Numero de elementos del arreglo (1-3).
#          $a1: Direccion de arreglo de $a0 palabras
# Salida:  $v0: Opcion elegida pseudo-aleatoriamente.
# Planificacion de registros:
# $t0: Auxiliar.
# $t1: Opcion 1.
escoger_aleatorio:
    # Prologo
    sw   $fp, ($sp)
    move $fp,  $sp
    addi $sp,  $sp, -4
```

```
    move $t0, $a0
    move $t1, $a1

    beq  $a0, 1, escoger_aleatorio_primero

    # (tiempo del sistema) mod (numero de opciones)
    li   $v0, 30
    syscall
    abs  $a0, $a0
    div  $a0, $t0
    mfhi $t0

    beqz $t0, escoger_aleatorio_primero
    beq  $t0, 1, escoger_aleatorio_segundo

    # Si no es 0 o 1, es 2
    lw $v0, 8($t1)
    j  escoger_aleatorio_fin
    escoger_aleatorio_primero:
        lw $v0, ($t1)
        j  escoger_aleatorio_fin
    escoger_aleatorio_segundo:
        lw $v0, 4($t1)

escoger_aleatorio_fin:
    # Epilogo
    move $sp,  $fp
    lw   $fp, ($sp)

    jr $ra
```

```
# Funcion: Imprime la puntuacion actual del juego.
# Entrada: $a0: Titulo de la puntuacion (Victoria, Derrota, Juego Finalizado)
# Planificacion de registros:
# $t0: Auxiliar.
# $t1: Auxiliar.
imprimir_puntuacion:
    # Prologo
    sw   $fp,   ($sp)
    move $fp,    $sp
    addi $sp,    $sp, -4

    # Imprime titulo
    li $v0, 4
    syscall
    # Imprime vidas
    la $a0, msgVidas
    li $v0, 4
    syscall
    lw $a0, V
    li $v0, 1
    syscall
    la $a0, nuevaLinea
    li $v0, 4
    syscall
    # Calcula puntuacion (porcentaje alimento)
    la $a0, msgComida
    li $v0, 4
    syscall
    lw  $t0, alimRestante
    lw  $t1, alimTotal
    sub $t0, $t1, $t0
    mul $t0, $t0, 100
    div $t0, $t0, $t1
```

```
    move $a0, $t0
    li   $v0, 1
    syscall
    la $a0, msgComida2
    li $v0, 4
    syscall
    la $a0, nuevaLinea
    li $v0, 4
    syscall
    # Calcula tiempo del juego
    la $a0, msgTiempo
    li $v0, 4
    syscall
    # Tiempo actual - tiempo inicial
    li  $v0, 30
    syscall
    lw  $t0, tiempo
    sub $t0, $a0, $t0

    # Convertir de ms a segs
    li $t1, 1000
    divu $t0, $t1
    mflo $a0
    li $v0, 1
    syscall
    la $a0, msgTiempo2
    li $v0, 4
    syscall
    # Imprime puntos
    la $a0, puntos
    li $v0, 4
```

```
syscall

# Epilogo
move $sp,  $fp
lw   $fp, ($sp)

jr $ra
```