

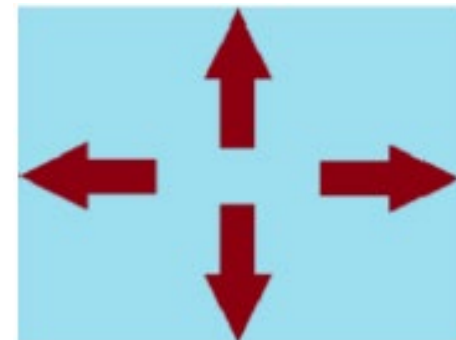
2024/2025 Mid-term Exam

Code review

Systèmes Embarqués Microprogrammés

Display the demo image on the bottom screen

- Show the provided demo image on the bottom screen
 - configure the background correctly (Ext. Rotoscale in 8-bit mode)
 - configure *arrows.grit* with correct flags
 - transfer palette and bit map to correct location



```
void configureGraphics_Sub() {
    // Configure the SUB engine in mode 5 and activate background 2
    REG_DISPCNT_SUB = MODE_5_2D | DISPLAY_BG2_ACTIVE;
    // Configure the corresponding VRAM memory bank correctly
    VRAM_C_CR = VRAM_ENABLE | VRAM_C_SUB_BG;

    void configBG2_Sub() {
        // Configure background BG2 in extended rotoscale mode using 8bit
        BGCTRL_SUB[2] = BG_BMP_BASE(0) | BG_BMP8_256x256;

        // Transfer image and palette to the corresponding memory location
        swiCopy(arrowsBitmap, BG_GFX_SUB, arrowsBitmapLen/2);
        swiCopy(arrowsPal, BG_PALETTE_SUB, arrowsPalLen/2);

        // Set up affine matrix
        REG_BG2PA_SUB = 256;
        REG_BG2PC_SUB = 0;
        REG_BG2PB_SUB = 0;
        REG_BG2PD_SUB = 256;
```

Mode 5 / BG2 +
VRAM C for sub

BG configuration

Bitmap/palette copy
from grit header

Affine transformation matrix
Default values → no transf.

controls.grit

-g
-gb
-gB8
-p

grit flags

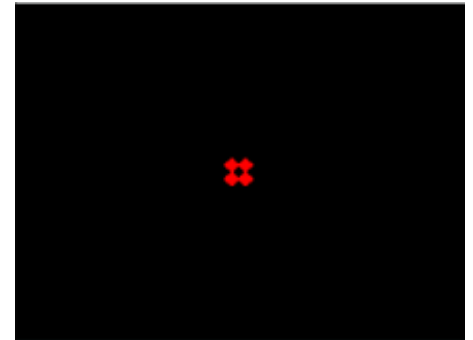
Exercise 2

Character on main screen

- Configure the main screen in tiled mode and draw the character in center
 - Configure the background correctly
 - Construct the character's tiles
 - Fill with empty tiles (provided)
 - Draw the character in the center

```
// Configure the MAIN engine in mode 5 activa
REG_DISPCNT = MODE_5_2D | DISPLAY_BG0_ACTIVE;
// Configure the VRAM bank A accordingly
VRAM_A_CR = VRAM_ENABLE | VRAM_A_MAIN_BG;
// Configure background BG0 in 32x32 tiled
BGCTRL[0] = BG_MAP_BASE(1) | BG_TILE_BASE(0)
            | BG_32x32 | BG_COLOR_256;
```

1. Mode 5 / BG 0
2. VRAM A for main
3. BG configuration
 - separate tiles / map



```
u8 player_Tile[64] = {
    0,0,0,1,1,0,0,0,
    0,0,1,1,1,1,0,0,
    0,1,1,1,1,1,1,0,
    1,1,1,1,1,1,1,1,
    1,1,1,1,1,1,1,1,
    0,1,1,1,1,1,1,0,
    0,0,1,1,1,1,0,0,
    0,0,0,1,1,0,0,0,
};
```



Diamond tile definition

Tile copies, palette and map filling

```
// Copy the tile(s) to the VRAM
```

```
dmaCopy(empty_Tile, (u8*)BG_TILE_RAM(0), 64);
dmaCopy(player_Tile, (u8*)BG_TILE_RAM(0) + 64, 64);
```

copy 2 tiles of 64 bytes,
continuous in VRAM

```
// Set color(s) in the palette
```

```
BG_PALETTE[1] = RED;
```

Write RED to palette pos. 1
(Pos. 0 is transparent color)

```
// Fill screen with transparent tiles
```

```
for(int i=0;i<24;i++) {
    for(int j=0;j<32;j++) {
        BG_MAP_RAM(1)[i*32+j] = 0;
    }
}
```

Fill the screen

- screen is 24x32 tiles
- fill all with empty (tile 0)
- fill center with player (tile 1)
- 4 central tiles:
 - (11, 15)
 - (11, 16)
 - (12, 15)
 - (12, 16)

```
// Set the main character in the 4 central tiles of
```

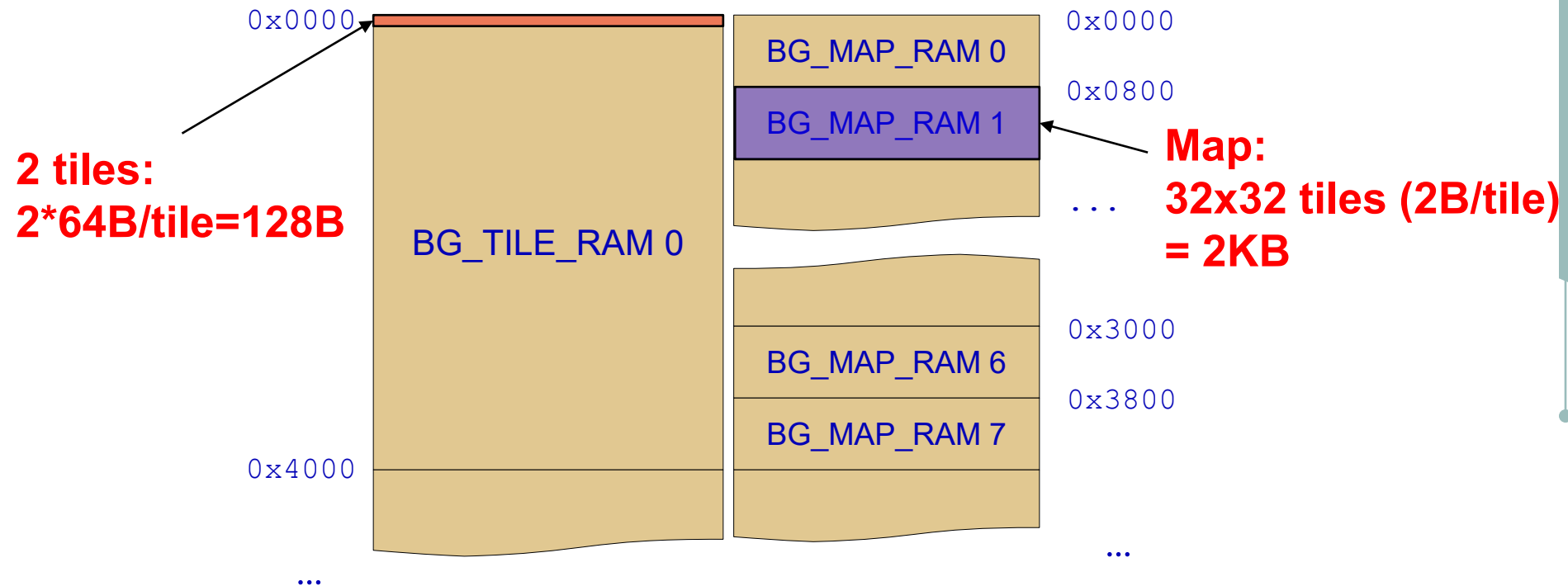
```
BG_MAP_RAM(1)[11*32+15] = 1;
BG_MAP_RAM(1)[11*32+16] = 1;
BG_MAP_RAM(1)[12*32+15] = 1;
BG_MAP_RAM(1)[12*32+16] = 1;
```

Note that we use **BG_MAP_RAM(1)** and **BG_TILE_RAM(0)** since we configured the BG with **BG_MAP_BASE(1) | BG_TILE_BASE(0)**

Exercise 2

Separating tiles and map in VRAM

- How the VRAM memory is mapped and used?
- BG_TILE_BASE(x)** increases address multiple of **16KB** (0x4000)
- BG_MAP_BASE(x)** increases address multiple of **2KB** (0x800)

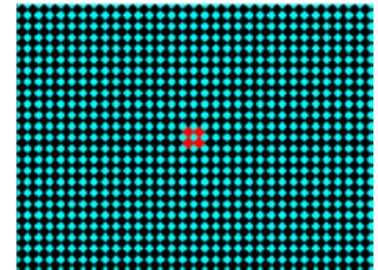


Use timer interrupts to initialize the game

- Use a timer to count a timeout, press X / START in time
 - Use a 200ms tick to count 3 seconds
 - Associate the *timerISR*
 - Use the best possible resolution
 - If X/START pressed → *initGame()*



timeout



initGame()

Divider	F=(33.514/DIV)MHz	T _{MAX} =2 ¹⁶ / F
TIMER_DIV_1	33.514 MHz	1.955 ms
TIMER_DIV_64	523.656 kHz	125.151 ms
TIMER_DIV_256	130.914 kHz	500.603 ms
TIMER_DIV_1024	32.729 kHz	2.002 s

- Only div. 256/1024 can count 200ms
- Best resolution: lowest divider
- Best divider: 256

```
TIMER0_DATA = TIMER_FREQ_256(5);
TIMER0_CR = TIMER_DIV_256 | TIMER_IRQ_REQ | TIMER_ENABLE;
irqSet(IRQ_TIMER0, &timerISR);
irqEnable(IRQ_TIMER0);
```

- 5 Hz → T = 1/5 = 0.2s = 200ms
- Associate *timerISR()*

```
void timerISR() {
    timer_ticks++;
    if(timer_ticks >= 15){
        irqDisable(IRQ_TIMER0);
        playerLoses();
    }
}
```

- increase tick
- at 15 ticks (3 seconds):
 - disable the timer interrupt
 - call *playerLoses()*

X / START press, *initGame()* - *playerLoses()*

```
scanKeys();
keys = keysDown();
if (keys & (KEY_START | KEY_A)) {
    initGame();
}
```

1. Read keys pressed with polling
2. Check if *START* or *A* was pressed
- call ***initGame()***

How do the functions *initGame()* / *playerLoses* fill the screen?

```
void playerLoses() {
    playerLost = 1;

    // Copy the red tile to the VRAM
    dmaCopy(red_Tile, (u8*) BG_TILE_RAM(0) + 128, 64);

    // Fill the screen with red tiles
    for(int i=0;i<24;i++) {
        for(int j=0;j<32;j++) {
            BG_MAP_RAM(1)[i*32+j] = 2;
        }
    }
}
```

```
void initGame() {
    // Copy the tile(s) to the VRAM
    dmaCopy(empty_newTile, (u8*) BG_TILE_RAM(0), 64);
    dmaCopy(player_newTile, (u8*) BG_TILE_RAM(0) + 64, 64);

    // Fill the screen with CYAN tiles apart from the player
    for(int i=0;i<24;i++) {
        for(int j=0;j<32;j++) {
            BG_MAP_RAM(1)[i*32+j] = 0;
        }
    }

    // Set the updated player tiles in the center of the screen
    BG_MAP_RAM(1)[11*32+15] = 1;
    BG_MAP_RAM(1)[11*32+16] = 1;
    BG_MAP_RAM(1)[12*32+15] = 1;
    BG_MAP_RAM(1)[12*32+16] = 1;
}
```

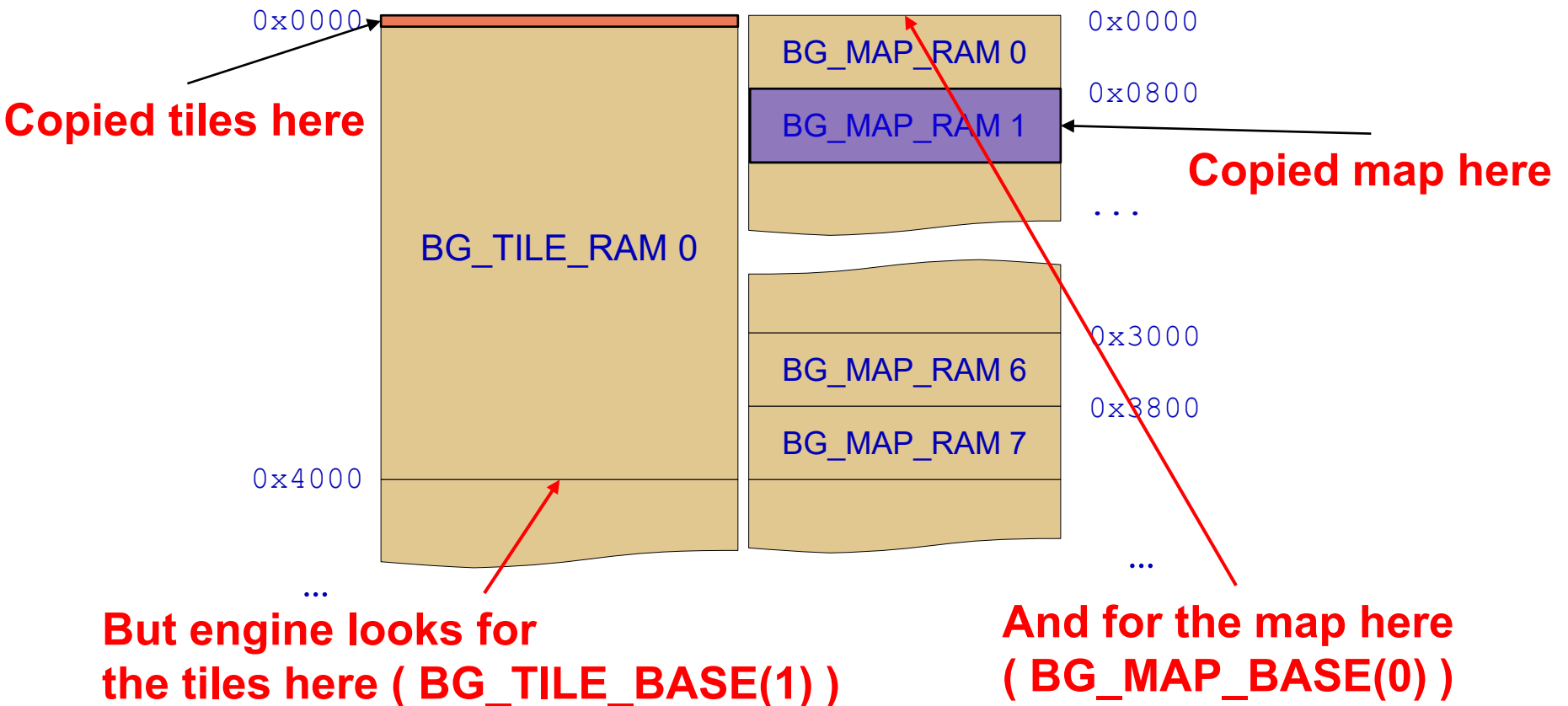
Only works if:

```
BGCTRL[0] = BG_MAP_BASE(1) | BG_TILE_BASE(0)
            | BG_32x32 | BG_COLOR_256;
```

Mismatch between _RAM and _BASE macros

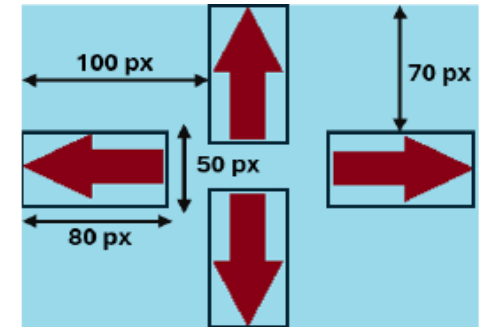
What happens if we use:

- **BG_TILE_BASE(1)** and **BG_TILE_RAM(0)**
- **BG_MAP_BASE(0)** and **BG_MAP_RAM(1)**



Use touchscreen arrows to move player

- Move the main character when touching the arrows
 - Four directions are available: left, right, up, and down
 - Use touchscreen polling coordinates to decide action
 - Move tiles according to *movePlayer<direction>()*



```

if (keys & KEY_TOUCH) {
    touchRead(&touch);

    if ((touch.px >= 100) && (touch.px < 150)) {
        if (touch.py < 80)
            movePlayerUp();
        else if (touch.py >= 110)
            movePlayerDown();
    }

    if ((touch.py >= 70) && (touch.py < 120)) {
        if (touch.px < 85)
            movePlayerLeft();
        else if (touch.px >= 180)
            movePlayerRight();
    }
  }

```

1. Check if a touch was made
2. Read the touch coordinates
 1. Isolate to $100 \leq x < 150$
 2. Isolate y for Up/Down arrows
 3. Call the corresponding function
1. Isolate to $70 \leq y < 120$
2. Isolate x for Right/Left arrows
3. Call the corresponding function

Moving player's tiles and screen borders

- Player consists of 4 tiles in square

- top-left: (x1, y1)
- bottom-right: (x2, y2)



```
void movePlayerRight() {  
    // Check if player is at the right edge  
    if (playerX2 == 31) return;  
    // Clear previous position using tile 0  
    BG_MAP_RAM(1)[playerY1*32+playerX1] = 0;  
    BG_MAP_RAM(1)[playerY2*32+playerX1] = 0;  
    // Move player and update the tile map  
    playerX1 += 1;  
    playerX2 += 1;  
    BG_MAP_RAM(1)[playerY1*32+playerX2] = 1;  
    BG_MAP_RAM(1)[playerY2*32+playerX2] = 1;  
}
```

Respect right edge
no action → return;

Clear **left-most** tiles (no trace)

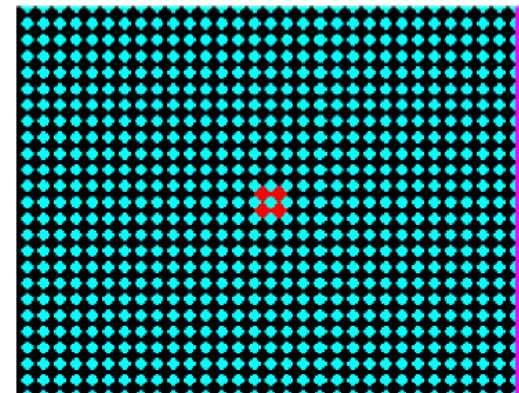
Advance x coordinates to the right
Retain y coordinates
Draw new **right-most** tiles
(1 tile displacement)

Same logic applies to Left, Up, Down displacements:

1. **Only one coordinate advances** (x or y)
2. **Respect screen borders**

Multiple backgrounds for portal teleportation

- Draw a portal at the right screen edge to teleport the character
 - Use an overlapping Rotoscale background on main screen
 - Change BG priorities so portal appears on top of the grid
 - Draw a portal with width equal to 1 tile at right edge
 - Teleport the character to left edge when passing through



```
REG_DISPCNT = MODE_5_2D | DISPLAY_BG0_ACTIVE | DISPLAY_BG2_ACTIVE;
```

```
BGCTRL[2] = BG_BMP_BASE(1) | BG_BMP(16, 256x256) | BG_PRIORITY(0);
```

```
REG_BG2PA = 256;
```

```
REG_BG2PC = 0;
```

```
REG_BG2PB = 0;
```

```
REG_BG2PD = 256;
```

```
for(int i=0; i<192; i++) {
    for(int j=248; j<256; j++) {
        BG_BMP_RAM(1)[i*256+j] = MAGENTA;
    }
}
```

- Skip tiles/tile-map of BG 0
- BG_BMP_BASE**: offset 16 KB
- 16-bit mode: framebuffer emul.
- Priority 0 (highest)

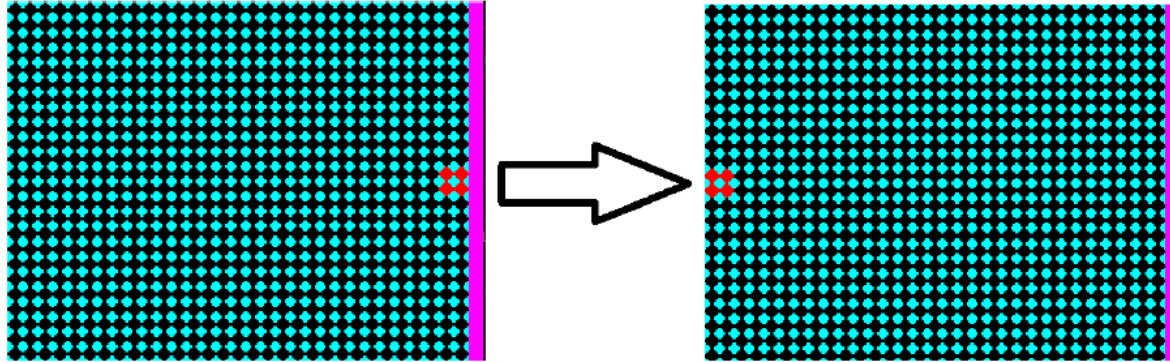
Draw final 8 pixels of each row (1-tile width)

```
BGCTRL[0] = BG_MAP_BASE(1) | BG_TILE_BASE(0) | BG_32x32 |
            BG_COLOR_256 | BG_PRIORITY(1);
```

Lower priority of BG 0 to 1

Exercise 5

Teleportation implementation



movePlayerRight()

```
if (playerX2 == 30) {
```

```
// Clear the previous position using tiles
```

```
BG_MAP_RAM(1)[playerY1*32+playerX1] = 0;
BG_MAP_RAM(1)[playerY2*32+playerX1] = 0;
BG_MAP_RAM(1)[playerY1*32+playerX2] = 0;
BG_MAP_RAM(1)[playerY2*32+playerX2] = 0;
```

```
// Update coordinates and teleport to the
```

```
playerX1 = 0;
playerX2 = 1;
BG_MAP_RAM(1)[playerY1*32+playerX1] = 1;
BG_MAP_RAM(1)[playerY1*32+playerX2] = 1;
BG_MAP_RAM(1)[playerY2*32+playerX1] = 1;
BG_MAP_RAM(1)[playerY2*32+playerX2] = 1;
```

```
return;
```

```
}
```

- Teleportation can only happen:
 - when moving right
 - AND player touches portal (x2 = 30)
- Clear previous position (all 4 tiles)
- Instantiate player at the left:
 - same y coordinates
 - completely visible (x1 = 0, x2 = 1)

Questions?

