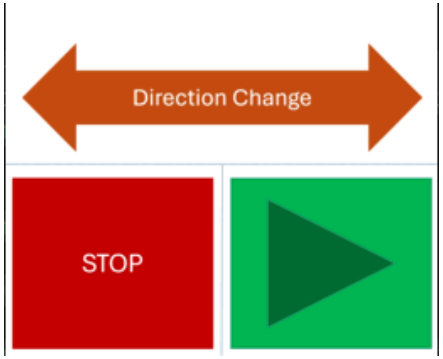
# 2025/2026 Mid-term Exam
## Code review

Systèmes Embarqués Microprogrammés

Prof. David Atienza Alonso, SEL-STI

# Exercise 1
## 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 *keys.grit* with correct flags
  - transfer palette and bit map to correct location




```
void configureGraphics_Sub() {
    // Configure the SUB engine in mode 5 and activat
    REG_DISPCNT_SUB = MODE_5_2D | DISPLAY_BG2_ACTIVE;
    // Configure the corresponding VRAM memory bank c
    VRAM_C_CR = VRAM_ENABLE | VRAM_C_SUB_BG;
void configBG2_Sub() {
    // Configure background BG2 in extended rotoscale
    BGCTRL_SUB[2] = BG_BMP_BASE(0) | BG_BMP8_256x256;
    // Transfer image and palette to the correspondin
    swiCopy(keysBitmap, BG_GFX_SUB, keysBitmapLen/2);
    swiCopy(keysPal, BG_PALETTE_SUB, keysPalLen/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.

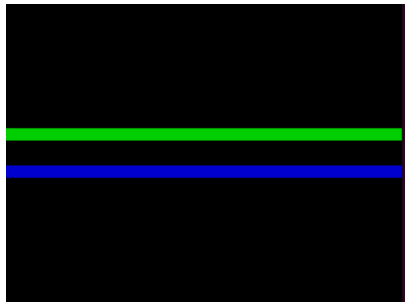
***controls.grit***

```
-g
-gb
-gB8
-p
```

- grit flags

# Exercise 2
# Horizontal lanes in 16-bit Ext. Rotoscale

- Draw Green/Blue lanes in 16-bit Ext. Rotoscale mode
  - Configure the background with bitmap at **16KB offset**
  - Draw the 8-pixel-wide lanes leaving the two central tiles empty

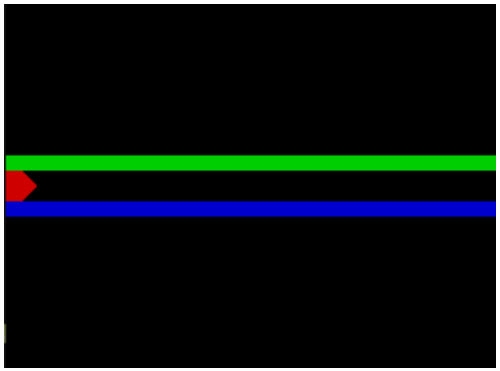


```
REG_DISPCNT = MODE_5_2D | DISPLAY_BG2_ACTIVE;
// Configure the VRAM bank A accordingly
VRAM_A_CR = VRAM_ENABLE | VRAM_A_MAIN_BG;

BGCTRL[2] = BG_BMP_BASE(1) | BgSize_B16_256x256;

u16* myGFX = (u16*) BG_BMP_RAM(1);
int row, col;
// 1. Green lines on top middle rows
for (row = 80; row < 88; row++) {
    for (col = 0; col < 256; col++) {
        myGFX[row*256 + col] = GREEN;
    }
}
// 2. Blue lines at bottom middle rows
for (row = 104; row < 112; row++) {
    for (col = 0; col < 256; col++) {
        myGFX[row*256 + col] = BLUE;
    }
}
```

- Mode 5 / BG2 + VRAM A
- 16KB offset + 16-bit mode
- 2-byte helper pointer at VRAM_A + 16KB address
- Middle pixels x: 192/2 = 96
- 96-8 < Empty pixels < 96+8
- 88 < Empty pixels < 104
- Assign color per pixel (defined in colors.h)

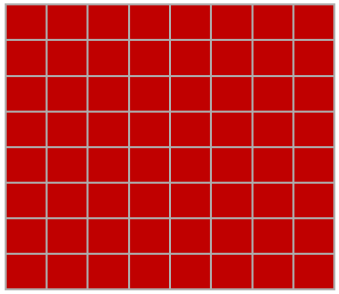
# Exercise 3
# Character on main screen

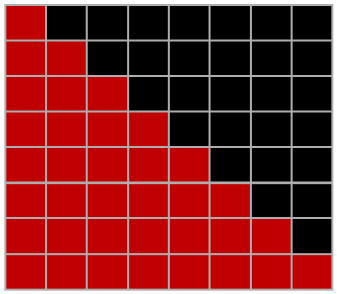
- Main screen in tiled mode and draw the character
  - Configure BG with tiles at start and map at 2KB offset
  - Construct the character's tiles + empty tile
  - Transfer tiles
  - Draw the character in the center

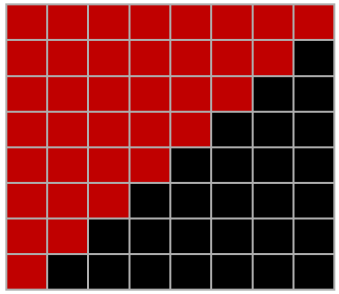


```
REG_DISPCNT = MODE_5_2D | DISPLAY_BG2_ACTIVE | DISPLAY_BG0_ACTIVE;
BGCTRL[0] =  BG_MAP_BASE(1) | BG_TILE_BASE(0) | BG_32x32 | BG_COLOR_256;
```

- Add BG 0
- Map at 2KB offset
- Tiles at 0KB offset







✓ Black color also accepted for player's non-red pixels!

```
u8 playerTile_rear[64] = {
    1,1,1,1,1,1,1,1,
    1,1,1,1,1,1,1,1,
    1,1,1,1,1,1,1,1,
    1,1,1,1,1,1,1,1,
    1,1,1,1,1,1,1,1,
    1,1,1,1,1,1,1,1,
    1,1,1,1,1,1,1,1,
    1,1,1,1,1,1,1,1,
};
```

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

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

```
u8 empty_Tile[64] = {
    0,0,0,0,0,0,0,0,
    0,0,0,0,0,0,0,0,
    0,0,0,0,0,0,0,0,
    0,0,0,0,0,0,0,0,
    0,0,0,0,0,0,0,0,
    0,0,0,0,0,0,0,0,
    0,0,0,0,0,0,0,0,
    0,0,0,0,0,0,0,0,
};
```

# Exercise 3
## Tile copies, palette and map filling

```
// Copy tiles into VRAM
dmaCopy(empty_Tile, (u8*)BG_TILE_RAM(0), 64);
dmaCopy(playerTile_rear, (u8*)BG_TILE_RAM(0) + 64, 64);
dmaCopy(playerTile_frontUp, (u8*)BG_TILE_RAM(0) + 2*64, 64);
dmaCopy(playerTile_frontDown, (u8*)BG_TILE_RAM(0) + 3*64, 64);
// Assign red color to palette
BG_PALETTE[1] = RED;
// Fill screen with empty tile
for(int i=0;i<24;i++)
    for(int j=0;j<32;j++)
        BG_MAP_RAM(1)[i*32+j] = 0;
// Draw character at left-most central tiles
BG_MAP_RAM(1)[11*32+0] = 1;
BG_MAP_RAM(1)[11*32+1] = 2;
BG_MAP_RAM(1)[12*32+0] = 1;
BG_MAP_RAM(1)[12*32+1] = 3;
```

- Copy 4 tiles of 64 bytes, continuous in VRAM
- Write RED to palette pos. 1 (Pos. 0 is transparent color)

Fill the screen:

- screen is 24x32 tiles
- fill all with empty (tile 0)
- Fill character tiles
- 4 left-most central tiles:
  - (11, 0) -- rear
  - (11, 1) – frontUp
  - (12, 0) -- rear
  - (12, 1) -- frontDown

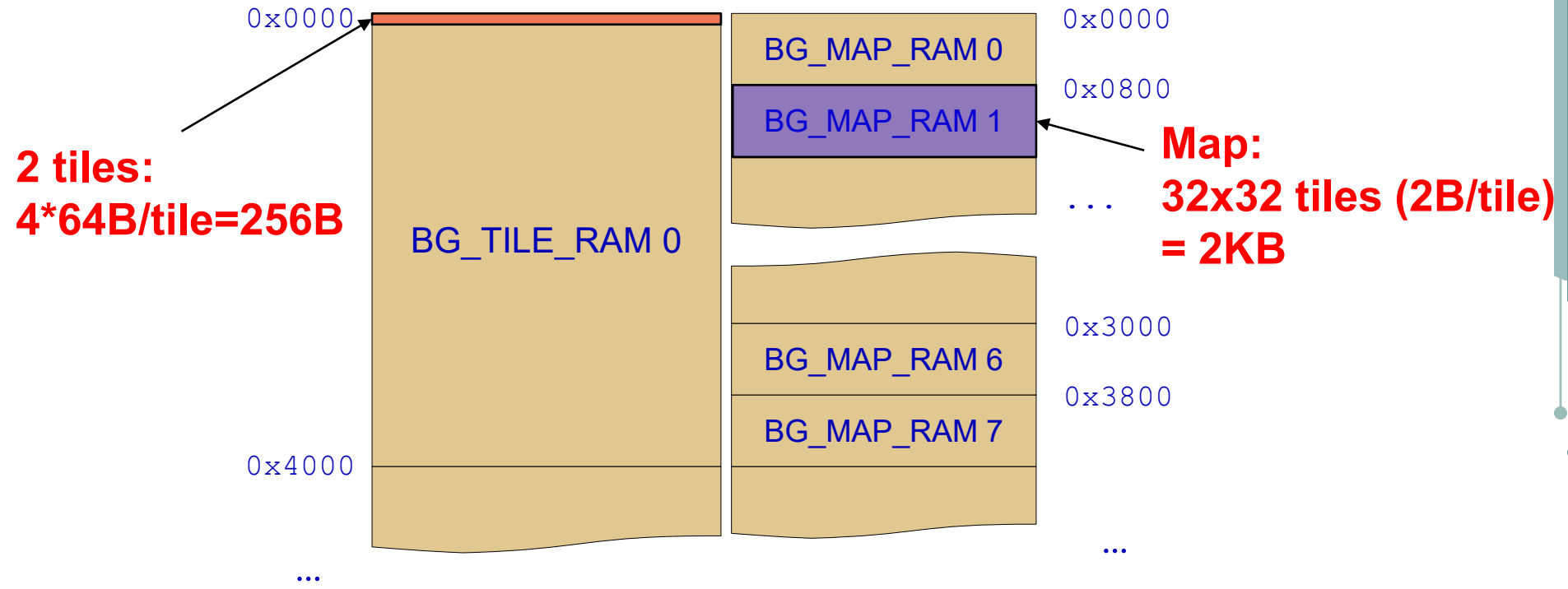
Note that we use ***BG_MAP_RAM(1)*** and ***BG_TILE_RAM(0)*** since we configured the BG with ***BG_MAP_BASE(1) | BG_TILE_BASE(0)*** )

# Exercise 3
# Separating tiles and map in VRAM

- How the VRAM memory is mapped and used?

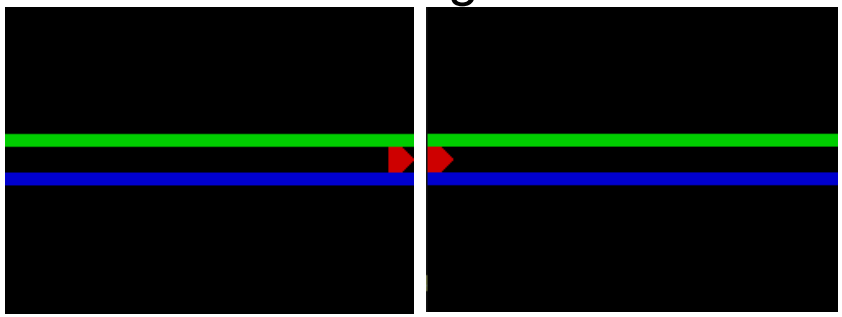
- ***BG_TILE_BASE(x)*** increases address multiple of **16KB** (0x4000)

- ***BG_MAP_BASE(x)*** increases address multiple of **2KB** (0x800)



# Exercise 4
## Use timer interrupts to move the character

- Use a timer with interrupt to move main character to the right
  - Use 125ms tick and associate the *timerISR*
  - Call movePlayer() inside timerISR
  - Move right – one tile displacement
  - Pass through edge without splitting



| 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 |

- Div. 1 cannot count 125ms

```
TIMER0_DATA = TIMER_FREQ_256(8);
TIMER0_CR = TIMER_DIV_256 | TIMER_IRQ_REQ | TIMER_ENABLE;
// Associate the ISR (timerISR) to the interrupt line and
irqSet(IRQ_TIMER0, &timerISR);
irqEnable(IRQ_TIMER0);
```

- 8 Hz → T = 1/8 = 0.125s
- Associate ***timerISR()***

```
void timerISR() {
    // Update player's position
    movePlayer();
}
```

- Call movePlayer() each time timer goes off

# Exercise 4
# Character's right movement

- Rear tiles: (X1, Y1), (X1, Y2), Front tiles: (X2, Y1), (X2, Y2)

```
if (movingDirection == RIGHT) {
    // Clear the previous rear tiles using t
    BG_MAP_RAM(1)[playerY1*32+playerX1] = 0;
    BG_MAP_RAM(1)[playerY2*32+playerX1] = 0;
    // Update position variables
    playerX1++;
    playerX2++;
    // Handle edge case
    if (playerX2 == 32) {
        // Clear also front
        BG_MAP_RAM(1)[playerY1*32+31] = 0;
        BG_MAP_RAM(1)[playerY2*32+31] = 0;
        // Reset to left edge
        playerX1 = 0;
        playerX2 = 1;
    }
    // Redraw tiles at updated position
    BG_MAP_RAM(1)[playerY1*32+playerX1] = 1;
    BG_MAP_RAM(1)[playerY2*32+playerX1] = 1;
    BG_MAP_RAM(1)[playerY1*32+playerX2] = 2;
    BG_MAP_RAM(1)[playerY2*32+playerX2] = 3;
}
```

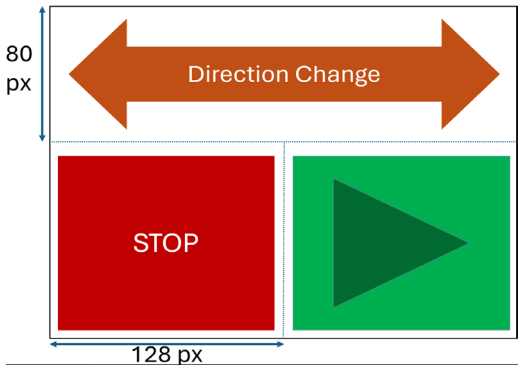
- Clear previous rear tiles
- 1 tile displacement to the right

Explicit handling for right-edge case
- Clear also previous front tiles
- Reset coordinates to left edge

- Draw tiles at correct positions:
  - (x1, y1): rear
  - (x1, y2): rear
  - (x2, y1): frontUp
  - (x2, y2): frontDown

# Exercise 5
# Use touchscreen to stop/play/change direction

- Handle each region touched accordingly
  - Use touchscreen polling coordinates to decide action
  - Stop: disable timer0 → no movement trigerred
  - Play: (re-)enable timer0
  - Change direction: Change direction + redraw character




```
touchRead(&touch);
if (touch.px != 0 && touch.py != 0) {
    // Direction change case
    if ((touch.py <= 80)) {
        if (movingDirection == RIGHT)
            movingDirection = LEFT;
        else
            movingDirection = RIGHT;
    }
    // Stop movement
    else if (touch.px < 128)
        irqDisable(IRQ_TIMER0);
    // Resume/Continue movement
    else
        irqEnable(IRQ_TIMER0);
}
```

- Read and check the touch coordinates

- If y ≤ 80 pixels:
  - Change direction left ←→ right

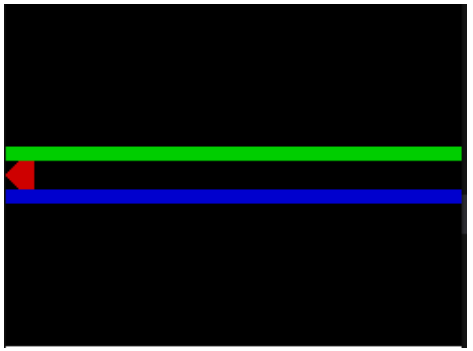
Else (y > 80 pixels):

- If x < 128 pixels:
  - Disable timer0
- Else:
  - Enable timer0

# Exercise 5
# Left movement and tile re-arrangement/flip

- Front tiles: X1, Rear tiles: X2

```
if (movingDirection == LEFT) {
    // Clear the previous rear tiles using the empty til
    BG_MAP_RAM(1)[playerY1*32+playerX2] = 0;
    BG_MAP_RAM(1)[playerY2*32+playerX2] = 0;
    // Update position variables and check corner cases
    playerX1--;
    playerX2--;
    if (playerX1 == -1) {
        // Clear also front
        BG_MAP_RAM(1)[playerY1*32] = 0;
        BG_MAP_RAM(1)[playerY2*32] = 0;
        // Reset to right edge
        playerX1 = 30;
        playerX2 = 31;
    }

    // Redraw tiles at updated position + Re-orientate p
    BG_MAP_RAM(1)[playerY1*32+playerX1] = 2 | (1 << 10);
    BG_MAP_RAM(1)[playerY2*32+playerX1] = 3 | (1 << 10);
    BG_MAP_RAM(1)[playerY1*32+playerX2] = 1;
    BG_MAP_RAM(1)[playerY2*32+playerX2] = 1;
}
```



- Clear “new” rear
- 1 tile displacement to the left
- New edge case: left edge
- Clear leftovers
- Reset to the right edge
- Re-arrange tiles:
  - X1: front
  - X2: rear
- Use 11th bit to horizontally mirror front tiles

# Questions?