SPACE SHOOTER

Philip Ljungkvist (et07jj5)
Jonas Johannesson (et07pl3)
Syed Zaki Uddin (sx08zs2)

Introduction

- Space Shoot'em up Game
- Two Dimensional Environment
- Space ships equipped with on board cannon
- Display on a VGA

specification

- **Output graphics to a VGA monitor with a resolution of 320x240 pixels @60Hz**
- **Update the player position by using Rotary encoder**
- **Detects collisions**
- **Independent background and foreground**

Partitioning

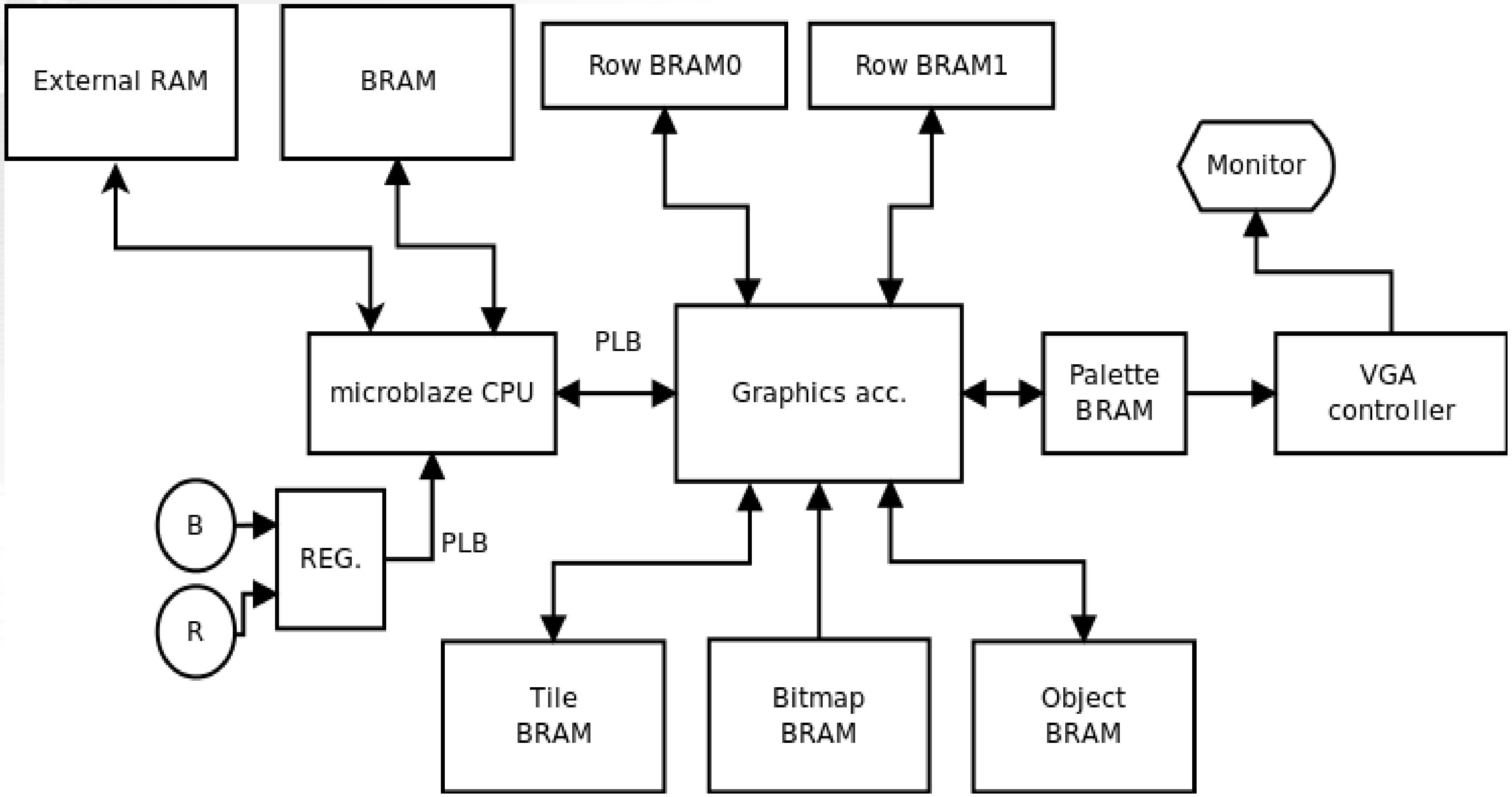
Hardware

- **Graphic Accelerator**
- **Rotary Encoder**
- **VGA Controller**

Software

- **Update positions for spaceships**
- **Handle collisions**
- **Background update**
- **Interrupt routines**

Architecture and Design



Memories

Five different Memories

- ROW RAM's
- BITMAP RAM
- TILE MAP RAM
- OBJECT RAM
- PALETTE RAM

TYPE OF MEMORY	SIZE in bits	FUNCTION
ROW RAM's	4160 x 2	Row Buffer
BITMAP RAM	49152	Stores tiles
TILE MAP RAM	8928	Stores tile's addresses
OBJECT RAM	6912	Stores position for objects
PALETTE RAM	256	Store colors
Total Memory = 73568 bits		

Graphics Accelerator

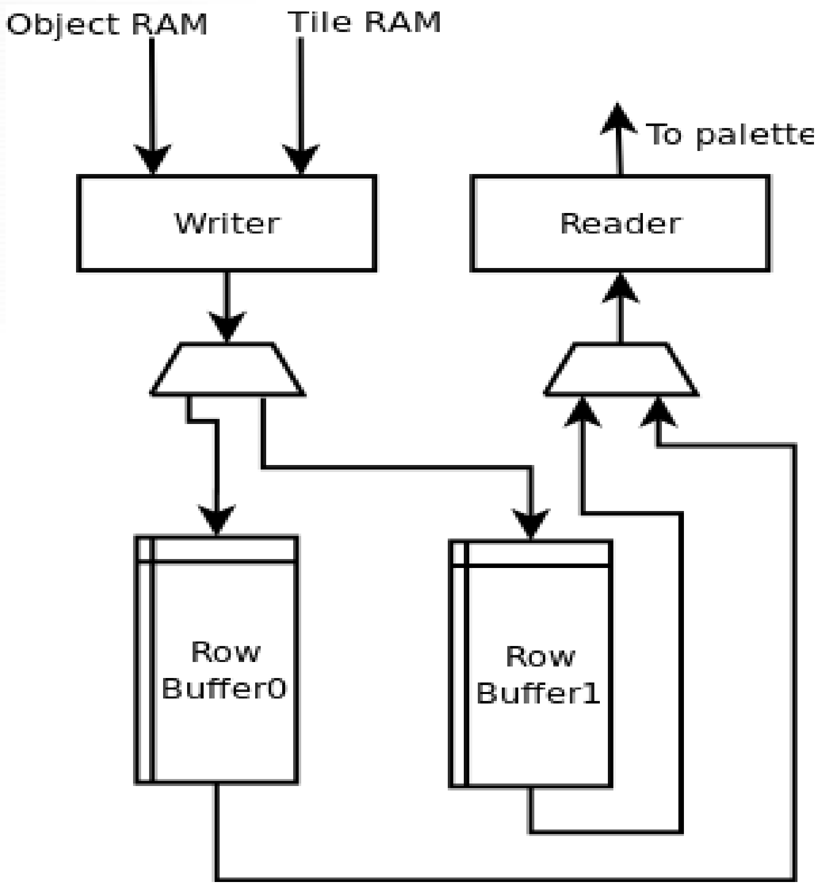
Two different kind of accelerators

1. Foreground renderer
- Used to draw space ships and bullets

2. Background renderer
- Used to draw background

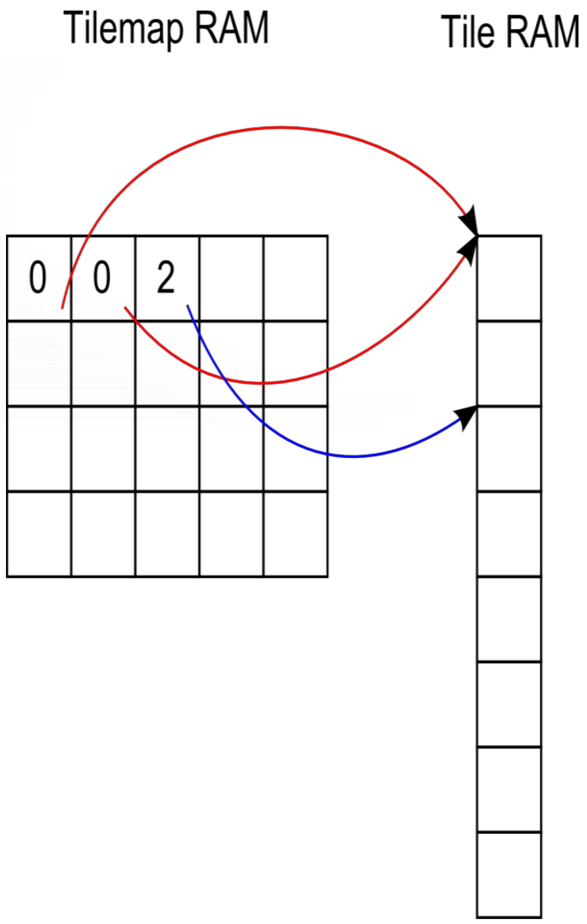
Foreground Renderer

- Uses a double row-buffer
- Reads position and tile number from object RAM
- Writes one buffer while other buffer is reading
- Switch buffer after each line

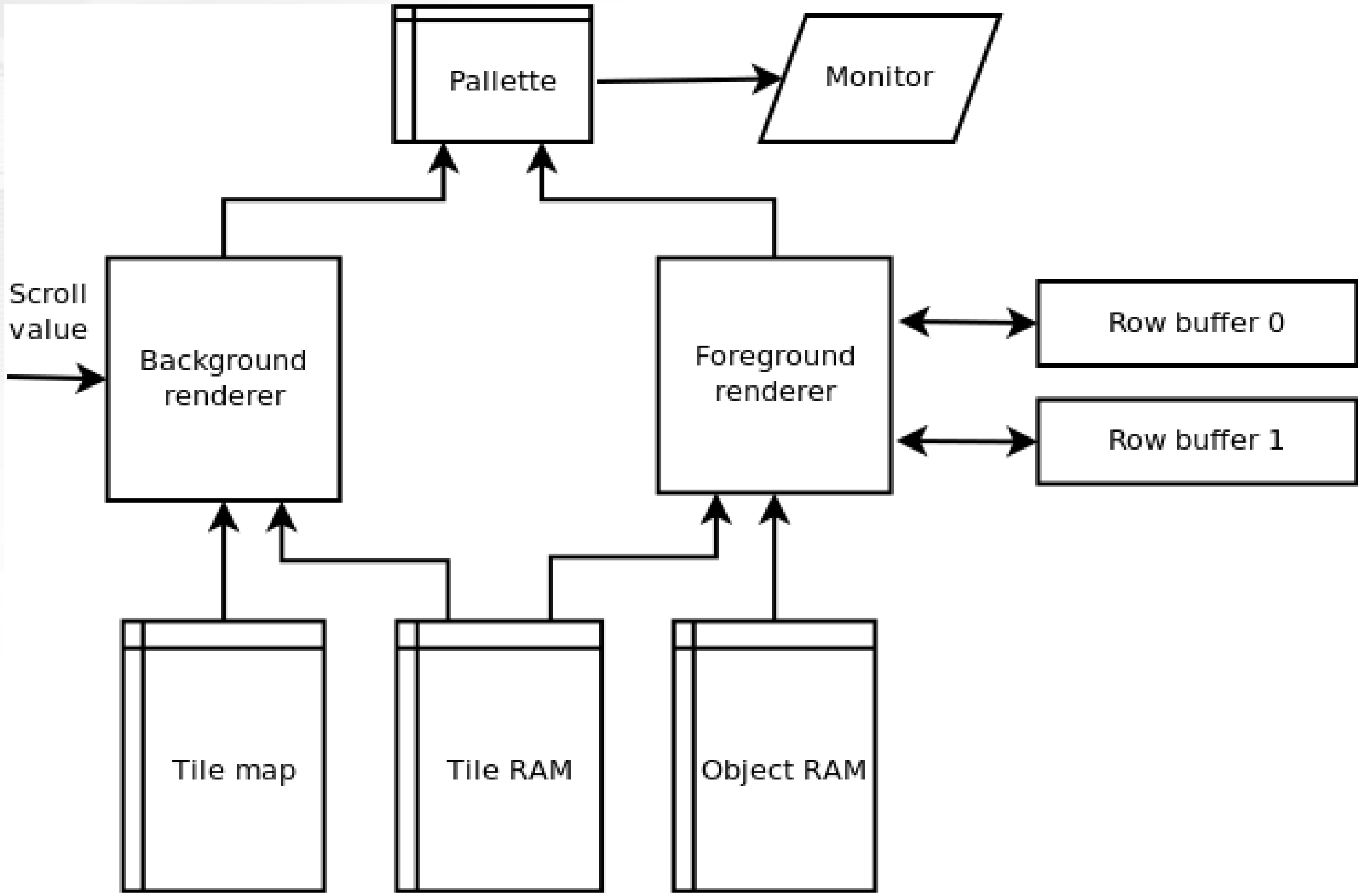


Background Renderer

- Background is stored in a grid of 32x15 tiles in the tile map RAM
- Each tile is 16x16 pixels
- Every word in the tile map RAM stores an address to the bitmap RAM and a palette number
- Only 20x15 tiles is visible at one time (320x240 pixels)



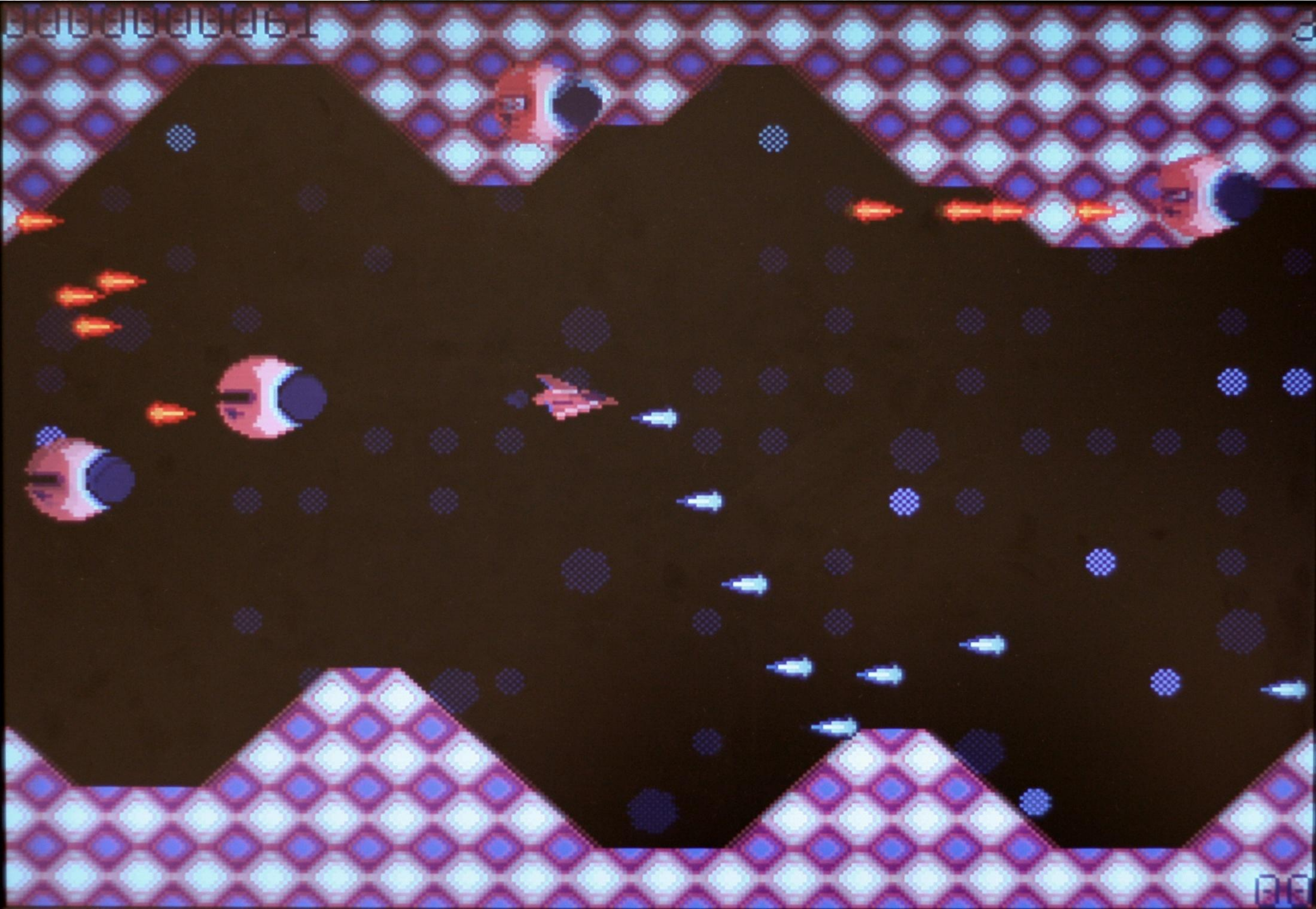
Graphics Accelerator



Software Implementation

- **Generate new random background column, and write tile map RAM**
- **Update object RAM during vertical blank**
- **Read inputs from controller**
- **Update player position**
- **Read time-stamped events and create new enemies**
- **Update enemy positions**
- **Handle collisions**

SPACE SHOOTER



Conclusion

- 2754 slices (31 percent)
- 25 BRAM's (89 percent)
- Estimated maximum Clock frequency is 162 MHz
- 24854 bytes of instructions

Problems and solutions

- The program was too large to fit in bRAM

  This was solved by putting the stack and the heap in external RAM.

- Getting address calculations correct turned out to be problematic

  Draw good images and diagrams helps a bit

Questions