Bomberman

Yichun Deng, Hanyi Du, Murui Li, Wantong Li
May 10th, 2016

Overview

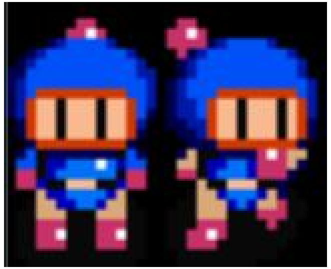
- Originally developed by Hudson Soft and first published in 1983
- Strategic, maze-based game
- A two-player version, where each player’s goal is to defeat the other player through placing bombs on the map

Image & audio Processing

- Generate a memory initialization file (MIF) for each image and sound
- Single-port ROM memory blocks
- To save memory space

For image : the three LSB-bits of image are truncated

For audio : edit audio files for length and sampling rate

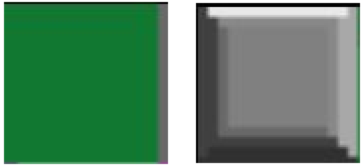


Sprite

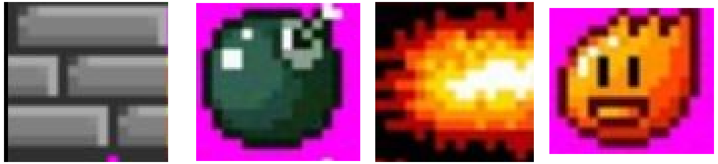
- Why use sprite?
    - We have lots of data to control and display
    - It is easy for us to add and delete sprite (i.e. 32*32 pixel for item, 32*64 pixel for character)
    - Code different small sprite instead of whole screen display

Sprite (cont.)

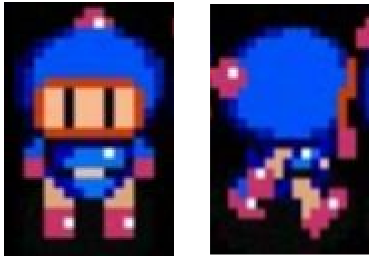
- Background
  - Total size: 480*608 pixel (15 rows*19 columns)
  - Mif: grass and stone



- Map
  - Total size: 416*544 pixel (13 rows*17 columns)
  - Mif: wall(1), bomb(1), flames(7), items(6)



- character
  - Total size: 32*64 pixel
  - Mif: red character(20), blue character(20)



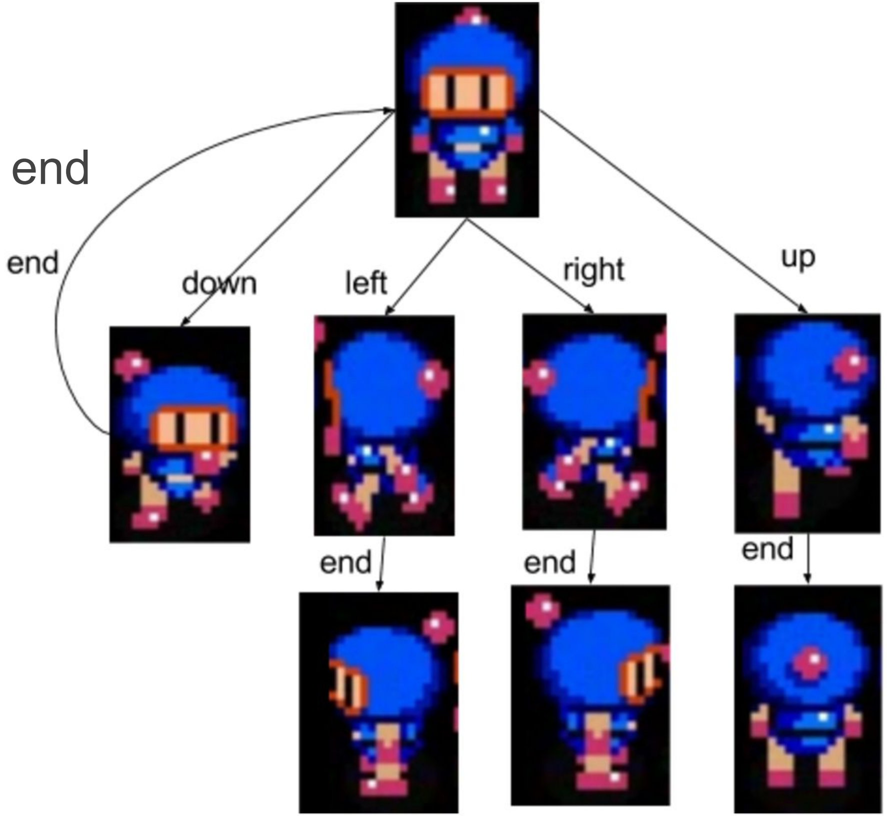
Memory (map)

- Propose: Hardware has the whole map information
  - Easy for VGA to display
  - Software can only send the changed information
- Memory size: 256 5-bit data (items)
  - Each address represent 32*32 pixel space
- Address: 8-bit read address for VGA, 8-bit write address for controller
  - No read and write contention

Character FSM

- Software control signal:

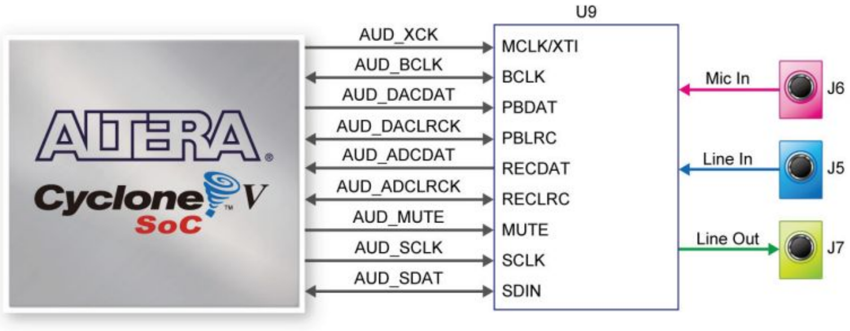
  Left, right, up, down, end



Audio

I2C bus for configuration:

- Sampling rate: 22050 Hz
- Quantization bits: 16 bits



Two kinds of audio

- Background music (in the right channel)
- Sound effect (in the left channel)

Reference: Howards Mao (http://zhehaomao.com/blog/fpga/2014/01/15/sockit-8.html)

Hardware Debugging

- Debug method: use system console to test the hardware performance
- Solved bugs:
  - Sprite display: clock synchronization for VGA and memory
  - Character movement:
    - replace control signal with FSM
    - Different address counter: use 4-bit MSB of hcount and vcount to control
  - Reset signal, multiple driver problem

Gamepad Control

- Logitech Gamepad F310
  - USB connection
  - We use the five of its digital keys: four for directions and 1 for placing bombs
  - Modified the usbkeyboard.c file to search for two devices with bDeviceClass == 255
  - After pairing, we decoded the signals that represent the pressing of each key

Software Modules 1

- Initialization
  - Game map, bomb map, characters, status
  - Start movement threads, bomb thread, and status thread
- I/O Control to Hardware
  - Send information regarding position, man/item, direction/item type
  - Use mutex to protect iowrite

Music (3 bits)	Position (8 bits)	Man or Item (1 bit)	Man: direction (5 bits) or Item: item type (5 bits)

- Movement Control
  - One thread for each player
  - Inputs from USB gamepads
  - Obstacle detection

Software Modules 2

- Bomb Control and Timing
  - Bomb thread with a timer to count down explosion for every bomb
  - Responsible for the flame animation
- Gifts Creation and Acquisition
  - Possibility of creating random gifts after destruction of each soft brick
  - Update character status after acquiring a gift
  - Gifts: ultra bomb, immunity, halt, reverse control, constipation
- End of Game
  - Three results: man1 wins, man2 wins, or draw
  - Stop taking inputs from gamepads but leave threads running
- Reset
  - Initialize game map, bomb map, characters, status

Software Debugging

- Print game map periodically to check game logic
  - To check if the software map == the map that the hardware displays
- Difficulties encountered:
  - Mapping
  - Synchronization
  - What to send
  - When to send
  - different explosion time for bombs

Lessons Learned

- Team work
- Hardware/software interface
- Debugging skills

Demo Time