Yi Wu (yw2707)
Yuxin Yang (yy2586)
Shengjia Zhang (sz2547)
Xiaoqing Yong (xy2246)

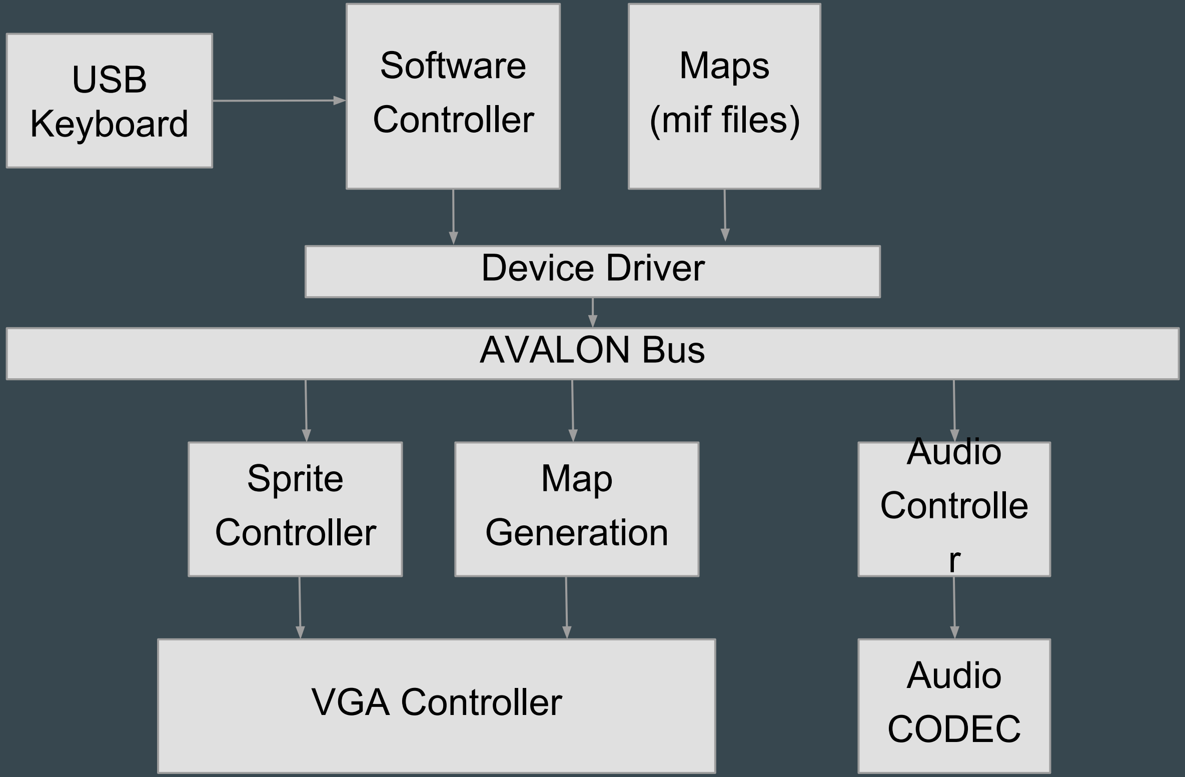
DGEsc: Dungeon Escape Game

CSEE 4840 Embedded System Design
Spring 2016

Overview

In this project we implemented a third person 2D RPG themed game on the SoCkit board using VGA monitor to display and keyboard to control. Player will control the character in the game to explore the maps, trigger plots, and finally beat the enemies. Following is an example of the game scene and our main character.

Overall Design



VGA Module

Background:

The dimension of all the maps are 640 × 480. We converted all the maps into MIF files and stored in software. Every time the character enters a scene, we transmit the corresponding MIF file to the hardware and display on VGA

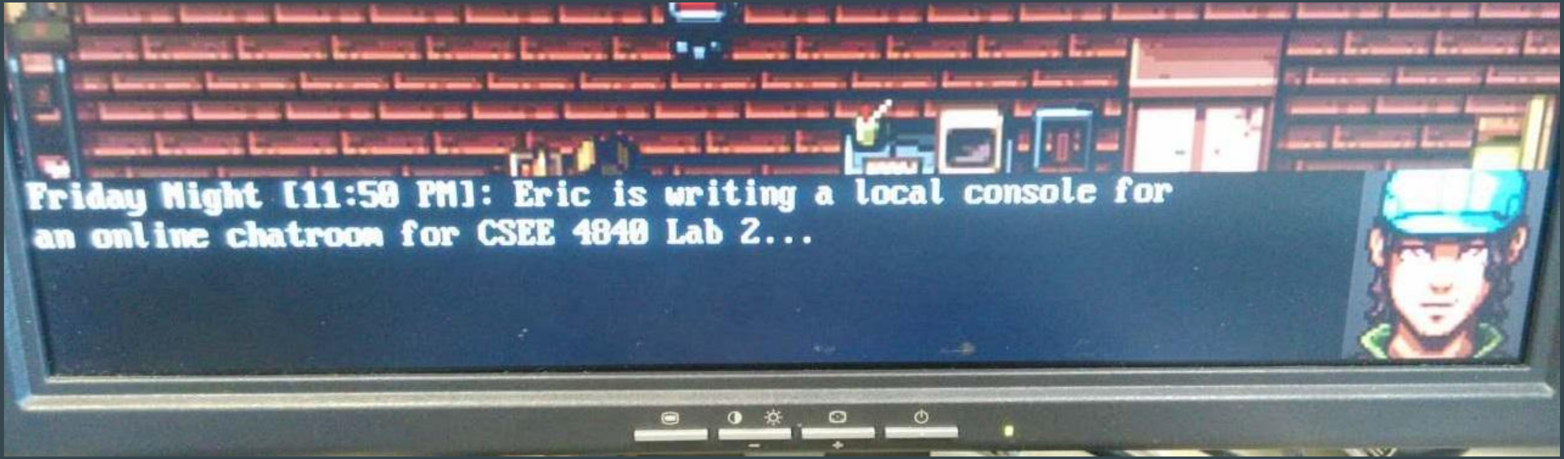
Sprite:

We use sprite to display our main character and some other items. Eric has four facing directions. When the character moves, we designed two extra transition statuses in between to simulate continuous movement. So there are 12 images for Eric.

VGA Module

Dialog:

We implemented the dialog using frame buffer. When a dialog is triggered in software, it’s written to the framebuffer through device driver, and displayed at the bottom.

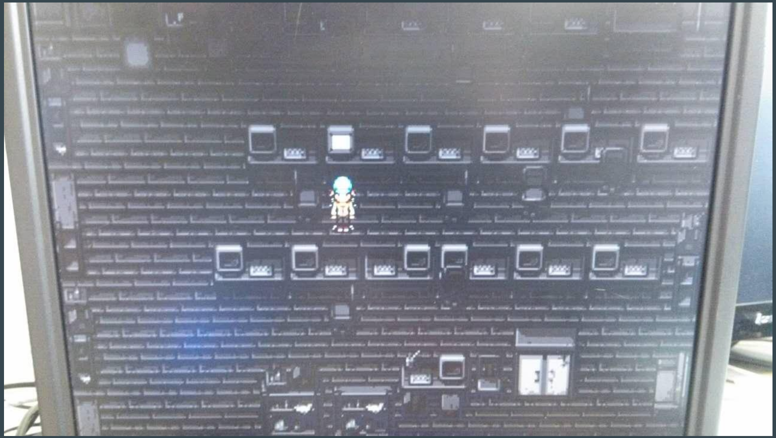


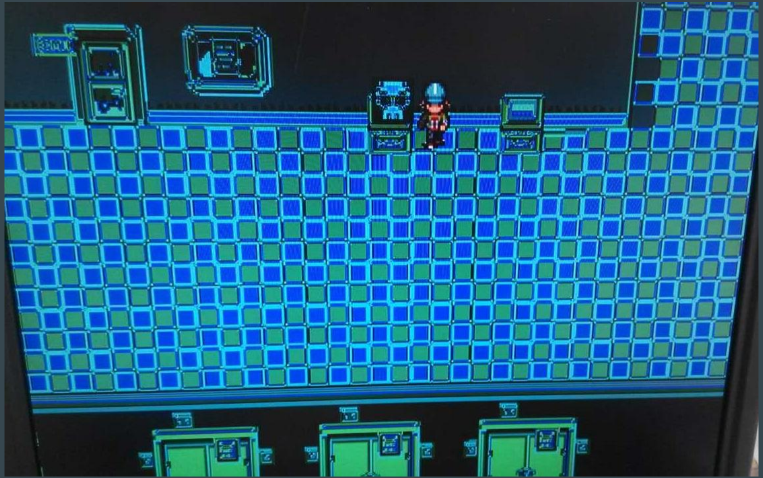
Friday Night [11:50 PM]: Eric is writing a local console for an online chatroom for CSEE 4840 Lab 2...

VGA Module

Visual Effects:

We can achieve many visual effects by manipulating R, G, B values of each background pixel. We designed effects of blackout and light flashing and entering alternate world. For blackout on a certain map, we set the R, G and B value of each pixel on background to be identical (e.g. all set to the R's value). For flashing, we achieve this by turning on and off the blackout effect. To enter alternate world, we simply set all R values to zero.





Audio Module

Configure codec (sampling rate, sampling width, etc.):

- 24 bit i2c_data:
  - 7 bit slave addr + 1 bit r/w + 7 bit register addr + 9 bit contents.
- I2C - serial data and serial clock.

Send samples to DAC (FPGA drives all of the clocks):

- MCLK - 11.28MHz
- LRclk - 11kHz (1/0 → L/R channel)
- BCLK - ¼*MCLK

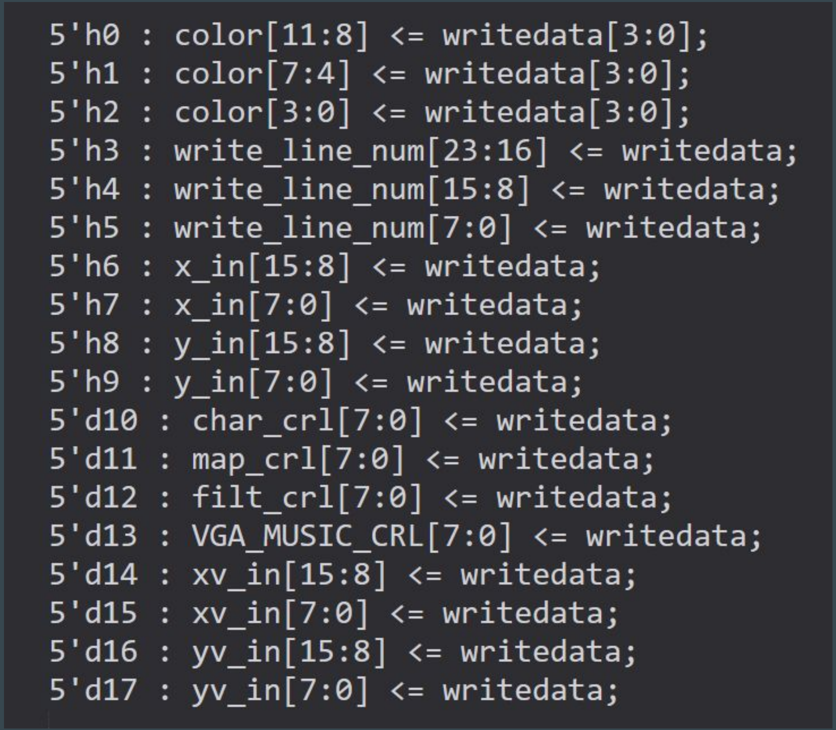
Audio Module

Control and play:

- 3 pieces of music stored in ROM.
- Music starts to play at the negative edge of beginning signal.
- Former music stops immediately whenever a new negedge occurs.
- Each piece of music plays for only once as per each negedge.

Hardware - Software Interface

Communication:



```
5'h0 : color[11:8] <= writedata[3:0];
5'h1 : color[7:4] <= writedata[3:0];
5'h2 : color[3:0] <= writedata[3:0];
5'h3 : write_line_num[23:16] <= writedata;
5'h4 : write_line_num[15:8] <= writedata;
5'h5 : write_line_num[7:0] <= writedata;
5'h6 : x_in[15:8] <= writedata;
5'h7 : x_in[7:0] <= writedata;
5'h8 : y_in[15:8] <= writedata;
5'h9 : y_in[7:0] <= writedata;
5'd10 : char_crl[7:0] <= writedata;
5'd11 : map_crl[7:0] <= writedata;
5'd12 : filt_crl[7:0] <= writedata;
5'd13 : VGA_MUSIC_CRL[7:0] <= writedata;
5'd14 : xv_in[15:8] <= writedata;
5'd15 : xv_in[7:0] <= writedata;
5'd16 : yv_in[15:8] <= writedata;
5'd17 : yv_in[7:0] <= writedata;
```

Software Design

Keyboard Controller:

We only use four keyboard keys throughout the whole game, which are ↑ , ↓ ,← , →, corresponding to moving Up, Down, Left, Right. Player can also check items and advance the plot using these keys

Maps in software:

Each of our map are represented by a matrix with 26 rows and 34 columns. Reachable areas are indicated by 0 and unreachables by -1. For areas where scene change happens, we marked them by the map number to be displayed. Everytime we the character is moved, we will query the destination coordinate in the matrix and act accordingly.

Game Logic

Triggering plots and dialogs:

Check character’s face direction and positions in the map to decide whether to trigger plots and dialogs.

Issues In Implementation

- Insufficient Memory
- Map Generation
- Write-enable
- Audio