CSEE 4840 Spring 2016
DGEsc: Dungeon Escape
Game Design

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

I. Design Overview:

This game is nothing like anything on the market. As Columbia EE students, we have taken classes on communication circuits, and RF design. The legendary yet tragic story of one of our outstanding alumni Edwin Howard Armstrong, the inventor of FM radio, inspired us to develop this game. You will further explore the details and plots of this Game Plot section.

This is an third person 2D RPG themed game. The map layout and character is designed to be in similar manner as the well-known RPG game “Legend of Zelda”[1], or “Final Fantasy”[2].

Figure 1: Legend of Zelda Game Scene (on the left), Final Fantasy Game Scene (on the right).

We use sprites to tile up the map, as well as represent characters, objects and monsters. On the hardware perspective, Verilog HDL are used to program, store and display sprites with VGA interface on FPGA board, as well as receive keyboard control commands and processing sounds. On the software perspective, we have write our program in C language for map and object generation, character movements, and other application of game rules.

II. Game Plot

Our game story happens on the 12th floor Mudd building, Columbia University, in embedded system lab. Eric, our main character is doing his homework on this local online chatting room in this lab room. He finishes at 1 a.m. at night, and there is nobody else in this room. At some point, he went out for the bathroom on the 13th floor - we allow the player to explore the building (12 and 13th floor) a little bit. Once he sent “Hello” in his console to this room, and see all the other computers light up, he was happy, and ready to go back home. However, after several seconds, someone from IP 195.4.1.31 replied “Hello Eric!”. Confused and scared because nobody else is in the room to send this message, Eric typed in “?”. And suddenly, the electricity goes off in this building. And his journey begins. Here is our main story line:

1. No service on his phone, we was locked out of the lab after he left. Stairwell to the 11th floor locked.
2. He explored in dark, and found that there is some weird liquid on Armstrong’s bust on 13th floor. Found a key and some scripts (circuits schematic hand drawn by Armstrong) inside the bust.
3. Collect components on 12th floor and 13th floor: 1. Audio Stage Amplifier - from Professor Yannis’s mailbox. 2. Radio Frequency Generator from circuit lab on 12th floor. 3. Radio Frequency power amplifier - from Professor

[1] The Legend of Zelda (Japanese: ゼルダの伝説 Hepburn: Zeruda no Densetsu?) is a high-fantasy action-adventure video game series created by Japanese game designers Shigeru Miyamoto and Takashi Tezuka. It is primarily developed and published by Nintendo, although some portable installments have been outsourced to Capcom, Vanpool, and Grezzo. The series' gameplay incorporates elements of action, adventure, and puzzle-solving games. The Legend of Zelda is one of Nintendo's most prominent franchises.

[2] Final Fantasy (ファイナルファンタジー Fainaru Fantajī?) is a science fiction and fantasy media franchise created by Hironobu Sakaguchi, and developed and owned by Square Enix (formerly Square). The franchise centers on a series of fantasy and science fantasy role-playing video games (RPGs). The eponymous first game in the series, published in 1987, was conceived by Sakaguchi as his last-ditch effort in the game industry; the title was a success and spawned sequels. The video game series has since branched into other genres such as tactical role-playing, action role-playing, massively multiplayer online role-playing, racing, third-person shooter, fighting, rhythm and anime.

Vallancourt's personal collection. Final component missing - mysterious modulator. 4. Speaker from a subwoofer on 12th floor.

4. Insert the key into pupin's bust, we find the mysterious modulator. (But start to hear foot steps). We are missing a power supply.
5. Go to maker's space to collect battery, and build this circuit. Meanwhile, the timer starts, if you don't build things quick enough, someone in black is going to approach you and kill you.
6. If built in time, someone starts talking through the receiver, giving you instructions to go to the basement of Mudd building through Vallancout's office secret tunnel.
7. End of Episode One.

III. Hardware Components

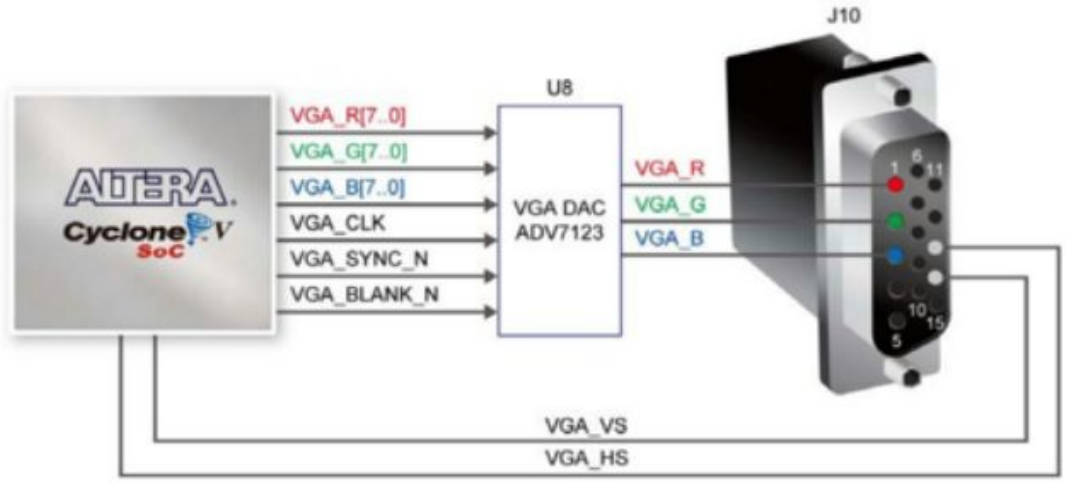
1. Keyboard Controllers

Keyboard is used to control the total game process; all the functions of the game should be realized with the use of keyboard. These functions include systematic functions such as pause, resume, start and exit; as well as gaming functions such as going forward, going back, making turns and fire. The functions of the keyboard are designed both in Verilog HDL and C language. In this design, we use following keys on keyboard and their functions are also listed below:

Key	Function
↑ , ↓ ,← , →	Up, Down, Left, Right movements of character, or selection
Esc	Menu
Return	Confirm
Space	Examine
Microphone	Blow out dust/other interactive event

2. VGA Block

SOCkitboard includes a 15-pin D-SUB connector for VGA output. The VGA synchro nization signals are provided directly from the Cyclone V SoC FPGA, and the Analog Devices ADV7123 triple 10-bit high-speed video DAC (only the higher 8-bits are use d) is used to produce the analog data signals (red, green, and blue). The following figure gives the associated schematic.



VGA is hardware designed in Verilog HDL to correctly connect the FPGA board with functions loaded to the monitor, as well as to display everything in the gaming window. VGA display would interact with memory which consists of RAM and decoder to generate display at the current moment with respect to the display stored as the previous game window. In VGA display,

there are kinds of elements to be shown:

Main character Eric:

Figure 2: Main Character Eric sprite.[3]

Wall (different choices of wall tiles):

Figure 3: Wall sprite.[4]

Floor (wood floor example):

Figure 4: Floor sprite.[5]

---

[3] http://www.deviantart.com/tag/charaset

[4]https://grandmadebslittlebits.wordpress.com/2015/06/13/rpg-maker-vx-ace-modern-interior-computer-server-wall-and-cabinet-tiles/

[5]http://www.deviantart.com/morelikethis/122524193

Objects example:

Figure 5: Scroll sprite.[6]

Dialogue Box Mask (Transparent):

Figure 6: Dialogue Box Mask.[7]

3. Audio Interface in FPGA

The Sockit Board we are using here Cyclone V provides high quality 24 bit audio via SSM2603 Audio CODEC chip. SSM2603 uses I2C protocol for configuration. .It supports three ports: Mic-in, Line-in(audio in), Line-out(audio out) and with various sampling rate from 8Khz to 96kHz. In this project, we are using Mic-In for player to interact with the game ambience to solve the puzzle. (For example, if there is a dusty document, the player can blow out dust via the microphone. This is little bit like the Japan version of Legend of Zelda in which you can shout at the microphone to kill the Pols Voice.)

[6]http://muhagames.com/sesja/items/
[7]http://forums.rpgmakerweb.com/index.php?/topic/3829-importing-to-ace-message-system/

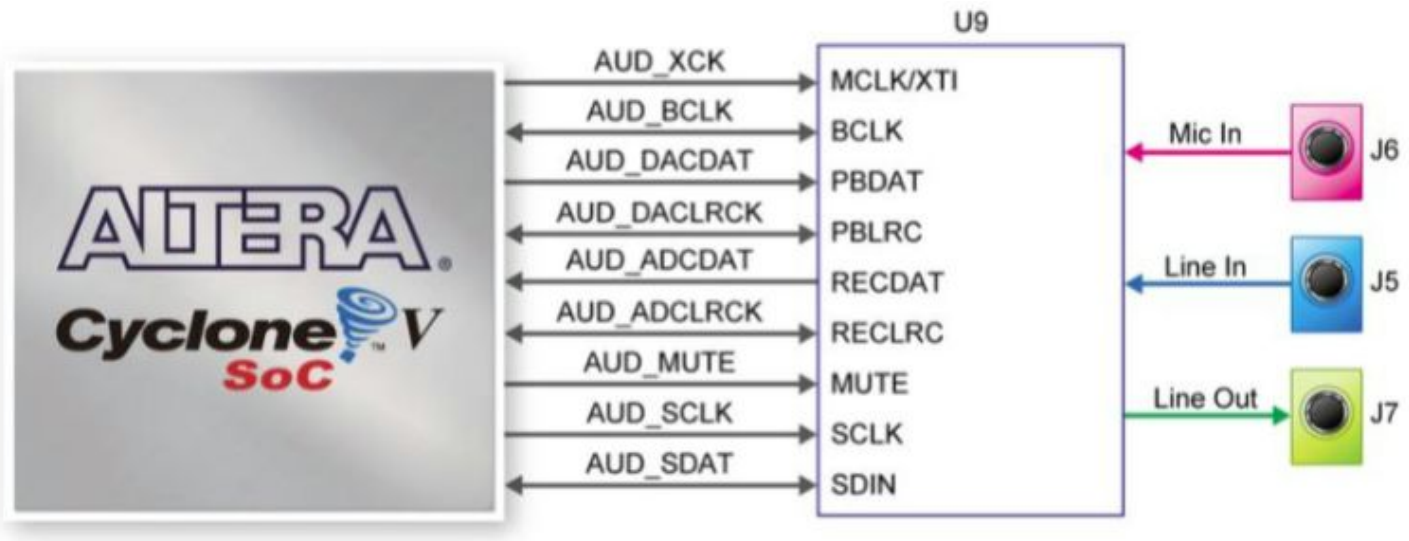


Figure. Connection between Cyclone V Sockit board and SSM2603 Audio CODEC[8]

Signal Name	FPGA Pin No.	Description	I/O Standard
AUD_ADCLRCK	PIN_AG30	Audio CODEC ADC LR Clock	3.3V
AUD_ADCDAT	PIN_AC27	Audio CODEC ADC Data	3.3V
AUD_DACLRCK	PIN_AH4	Audio CODEC DAC LR Clock	3.3V
AUD_DACDAT	PIN_AG3	Audio CODEC DAC Data	3.3V
AUD_XCK	PIN_AC9	Audio CODEC Chip Clock	3.3V
AUD_BCLK	PIN_AE7	Audio CODEC Bit-Stream Clock	3.3V
AUD_I2C_SCLK	PIN_AH30	I2C Clock	3.3V
AUD_I2C_SDAT	PIN_AF30	I2C Data	3.3V
AUD_MUTE	PIN_AD26	DAC Output Mute, Active Low	3.3V

Figure. Pin Assignment for SSM2603 Audio CODEC[9]

In our project, we will collect the information from game event and generator audio data through the subtractive synthesizer. The real-time audio will be put in the music buffer via HPS so that the SSM2603 codec can retrieve the audio file and play it through the line out.

IV. Software Components

1. Sprite Control

The sprite stored in SRAM with mif format can be displayed on screen on given X and Y coordinates. This is done by the sprite controller. It takes X and Y coordinates from game logic controller and access the sprite data on the SRAM with given address, then the sprite controller can generate RGB values for the VGA controller.

[8] http://www.terasic.com.tw/cgi-bin/page/archive_download.pl?Language=English&No=816&FID=55132755c0462d5ad3f226ab804d0cf0

[9] http://www.terasic.com.tw/cgi-bin/page/archive_download.pl?Language=English&No=816&FID=55132755c0462d5ad3f226ab804d0cf0

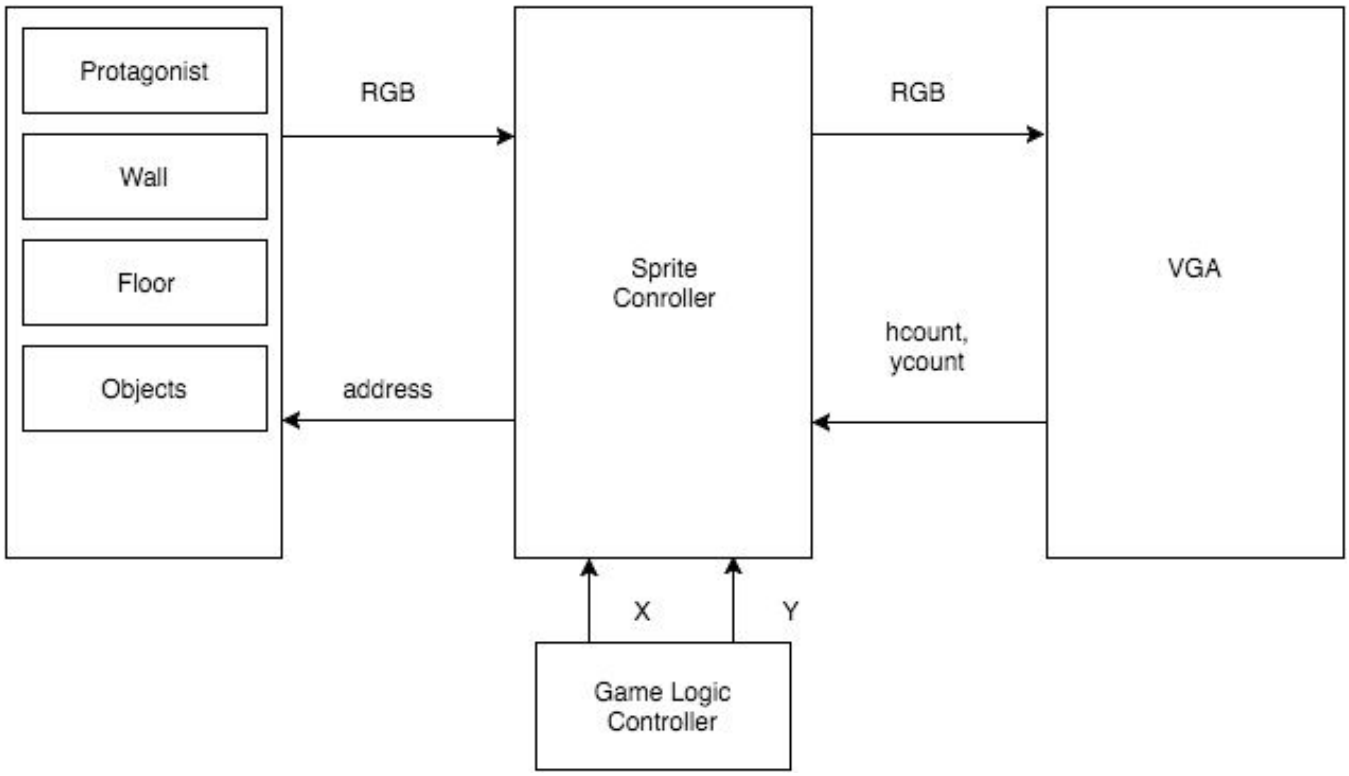


Figure. Sprite Controller Block Diagram

2. Audio

Although the DDR3 1Gb SRAM in SocKit board should be capable of around 2 hours of cd quality audio data, in order to save the memory for all the image files, we need limited the memory allocation for audio data. The solution of not reduce the audio quality and limited the memory size of audio data is to generate the audio real-time during the game play. Generating real real-time audio can not only reduce the memory size of audio data, but also dramatize the gameplay experience and create a cinematic feeling.

Currently we have two approach for this solution. First is implement a subtractive synthesizer for FPGA. Analog synthesizers in 1960s and 1970s commonly use a technique called “subtractive synthesis”. It is a method to synthesize audio signal by attenuating the original audio signal. The attenuation can be done with applying the original signal to filter, envelope generator and low frequency generator. Those analog synthesizers back to 1960s and 1970s fed the simple waveforms like sine waves, sawtooth waves, square waves and pulse waves to filters and envelope generator to synthesize different kinds of sound. With different event the player triggered, different setting will be input to the subtractive synthesizer and generate different sound. This will make game experience more interesting instead of one boring, unrealistic sample.

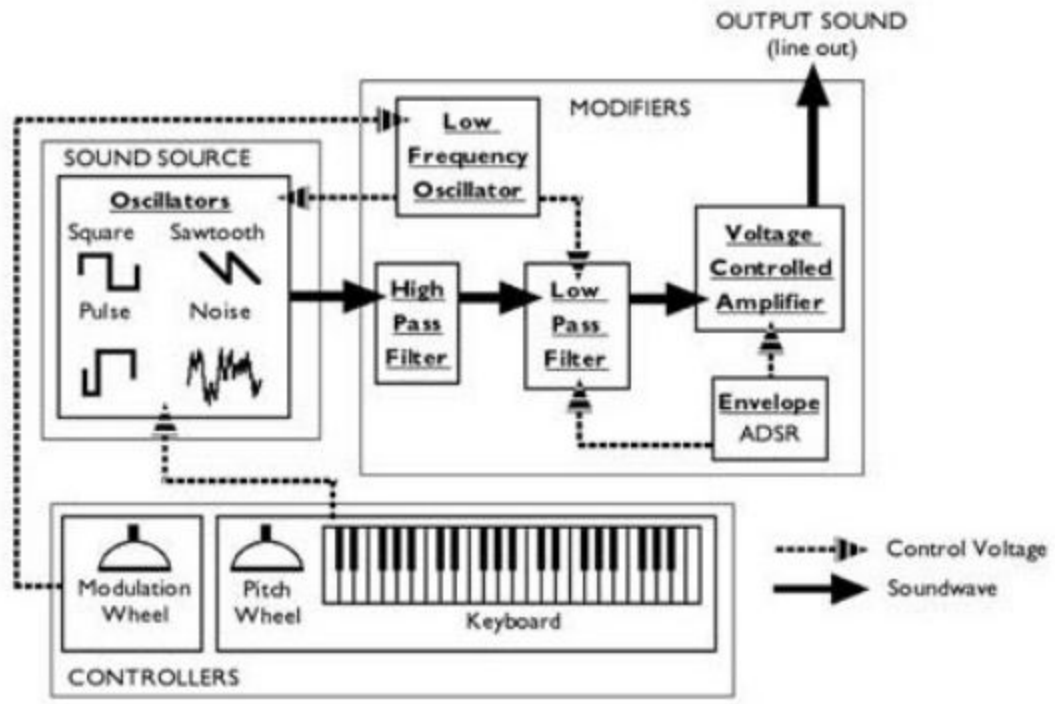


Figure. Signal Flow of a Typical Subtractive Synthesizer[10]

Although subtractive synthesizer with ADSR envelope generator provides a solution of good quality instrument simulation, it might occupied a lot CPU computation resources. Subtractive Synthesizer might only suitable for sound effects. Therefore, a CPU resources saving approach should be considered for game background music playback.

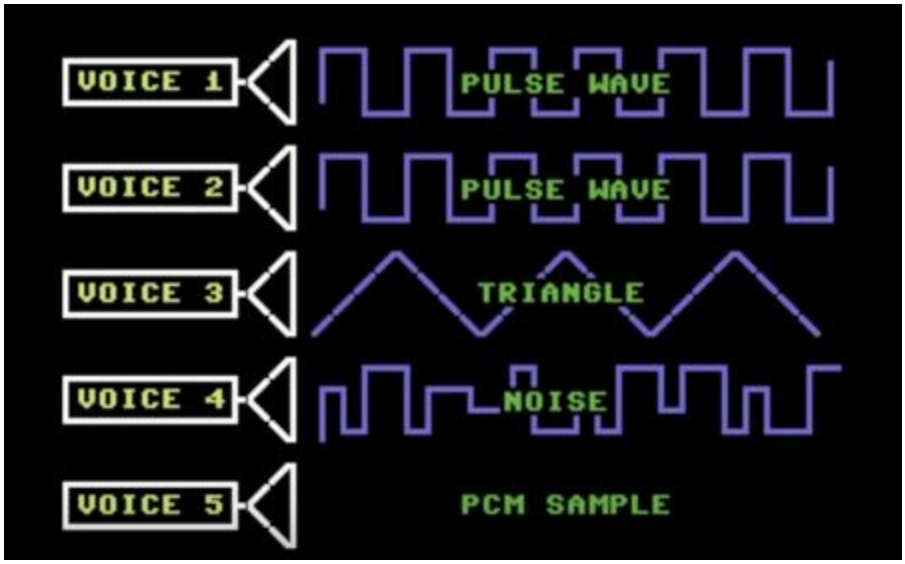


Figure. NES (famicon) Audio Generator[11]

We will use the idea from the NES, using the signal generator from the subtractive synthesizer to compose/play background music. The original NES has five channel. Two pulse wave generator, one triangle wave generator, one noise generator and one channel for PCM sample. In our project, we will use two pulse wave generator, one triangle wave generator and one noise generator to playback the background music.

V. Milestones

Thu Mar 24	Design Ready.

[10] http://www.planetoftunes.com/synthesis/subtractive-synthesis.htm.

[11] https://www.youtube.com/watch?v=q_3d1x2VPxk

Tue Mar 29	Milestone prepare: Convert JPEG tiles into mif file. Get sprites shown on our screen.
Thu Mar 31	Milestone 1: Background map generation. Using software to control the positions of the tiles, and hardware to display the tiles. Generate a dialogue mask over the screen to pass message to players.
Tue Apr 5	Configure the movement of the character, controlled by our keyboard. Configure the interaction between the character and objects.
Thu Apr 7	Test character movements, interaction, and dialogue inside the map, and debug.
Tue Apr 12	Milestone 2: Insert story into our software. Have the software framework ready for the first episode.
Thu Apr 14	Debug.
Tue Apr 19	Integrating sound effect, and possibly sound control into the game.
Thu Apr 21	Debug.
Tue Apr 26	Milestone 3: We should have a working bugless prototype of the first episode of our game.
Tue Apr 28	Presentation, demonstration, and final report ready.