“Network Clue” Design Document

Gaurav Gupta, Khalef Hosany, Sampada Sonalkar, Thomas Mwakibinga

CS 4840 Embedded System Design

This is the design document for the Network Clue game.

The design consists of a hardware module for displaying the game board and user interface on the monitor and a software module for controlling the play and exchanging information among players.

Hardware design

The 640x480 pixel monitor display consists of a 480x480 pixel game board and a user interface area. The graphics is handled by the FPGA and the unit communicates with the NIOS processor through the Avalon Bus. As part of the display is static (the game board itself) and part is dynamic (the pieces, weapons etc..), the system will first of all display a static image on the screen and then overwrite certain parts with sprite information representing moving pieces on the screen.

In order to display colors, several color palettes will be used. These will be selected by a 4-bit input. The color palettes will be 30 bits wide, corresponding to 10bit R, G, and B values that are sent to the DAC and displayed as a certain colored pixel. There will be 16 palettes which will themselves have 16 memory addresses. Each address, except for 1111, will be used for a certain color. Thus eache palette holds 15 colors. A pixel having 1111 within a Sprite will be treated as clear and the board will be displayed instead. This allows us to include transparency in the sprites; leading to much crisper graphics. Using such a combination of palettes allows us to display a large number of colors using relatively less memory space, again leading to a more enjoyable visual display.

In order to further optimize memory usage, we will be using blocks of 8*8 pixels to represent the display information. As the display resolution is 640*480, we will have 80 such blocks horizontally and 60 vertically. Thus the total number of blocks needed to represent the whole display is 4800. This defines the number of memory addresses in the board RAM.

The sprite RAM contains the current location of the starting point of a sprite block on the screen. It will thus have as many address lines as sprites available in the final design and is accessed by the NIOS in order to move sprites on the screen.

The Font RAM is 36bits wide and contains pixel information for both the sprites and the board. The first 4 bits are used to select the color palette. The remaining 32 bits are indexes into the selected color palette to activate a certain color. As 4 bits are required per pixel, we will be able to save information for 8 pixels(representing one line of the block defined above) on one single Font RAM address line.

The sequence of operation is as follows:

- During horizontal blanking, the address of all the sprites are loaded from the Sprite RAM to registers. The index into the font RAM from the board RAM is also loaded into a separate register.
- As the line is being displayed, the current horizontal and vertical position is used to determine whether a sprite or the board needs to be displayed for the next pixel block
- If a sprite is found, the corresponding address is passed to the font RAM through the multiplexer. Otherwise the output of the multiplexer is ignored.
- The font ram is accessed by the board register and the sprite register to obtain the respective pixel information. This 36 bit data is broken down into 2 parts: 4 bits sent to a register for the color palette and 32 bits sent to a shift register.
- 4bits are shifted out at a time from the shift register (corresponding to one pixel). The 4bits coming out of the sprite shift register are examined. If they display 1111, we have a transparency at this point and the board pixel is sent to the palette RAM. Otherwise the sprite information is sent. In both cases, the corresponding color palette (stored in the register) is also sent.
- The palette RAM thus receives 8 bits, selects a certain color palette, chooses a color in that palette and sends it to the DAC to be displayed.
- This is repeated for the 7 remaining pixels in the block, the block position is updated, the new register values are filled in and the process repeats.
- The proper operation of the unit is handled by the controller, which calculates offsets, compares addresses, control the multiplexer select lines, chooses the sprite or board pixel display, generates HSYNC, VSYNC, BLANK etc..

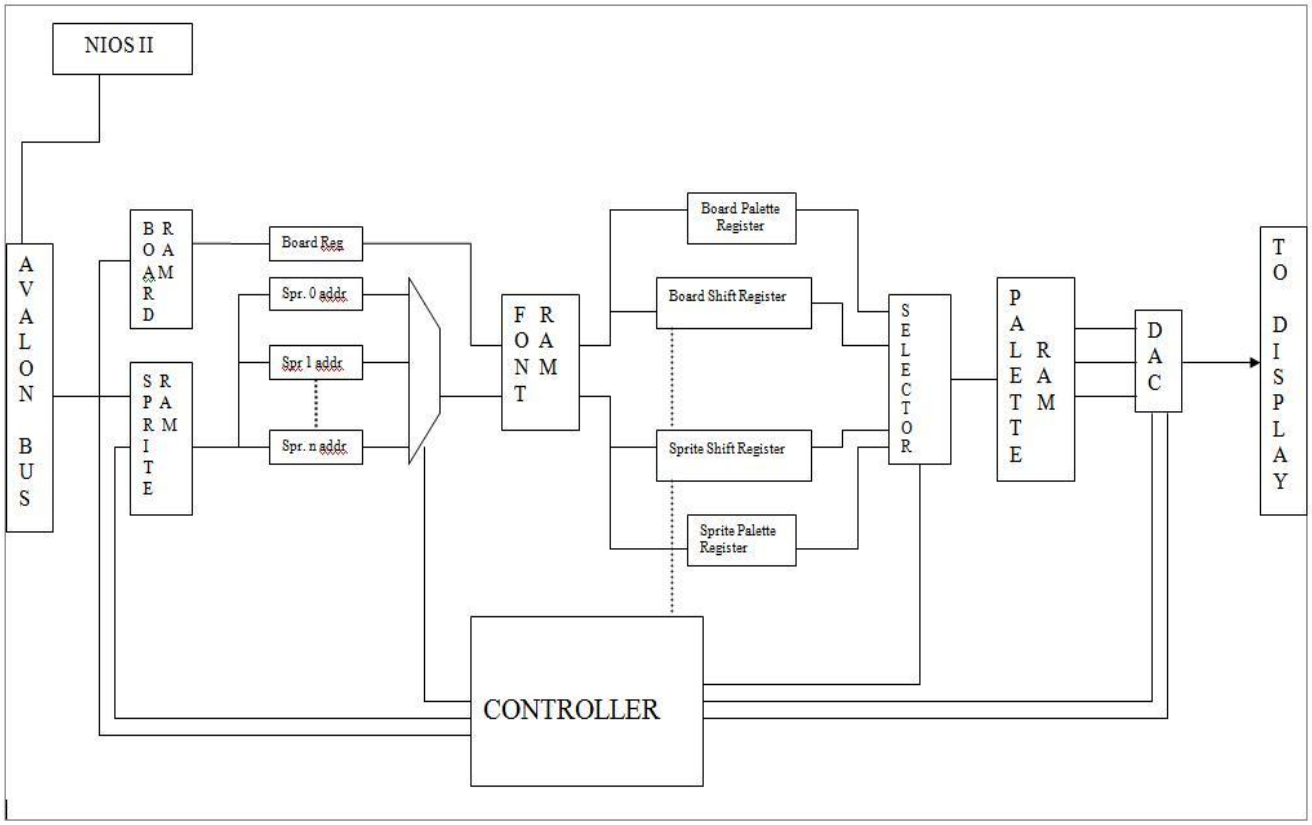

Figure 1 Video System

Software design

The organization of the game logic is as shown in Figure 2. The game logic controls the game play and interfaces the keyboard, video display and the networking components.

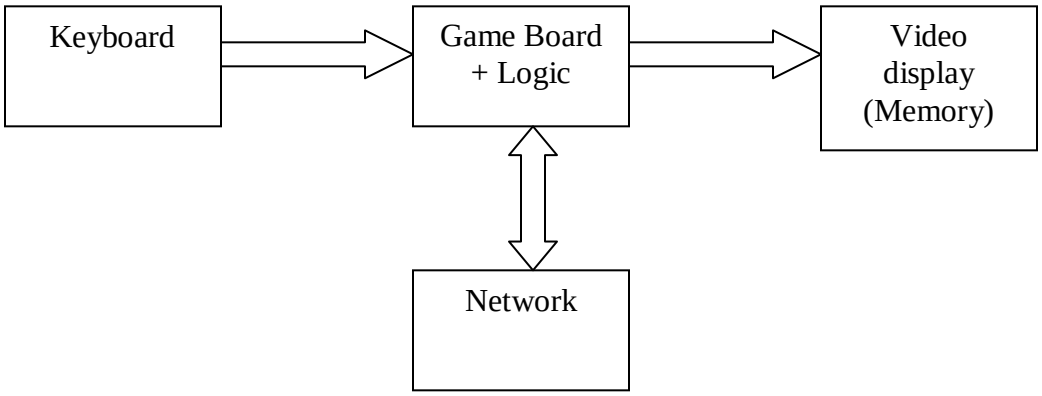

Figure 2 Software Organization

The game logic needs to perform the following tasks

1. Initial setup:
    a. The game board is a 20*20 tile board with 9 rooms connected by passages.
    b. The game logic displays the sign-in screen and allows the player to select a character for play. It also sets up the turns for players.
    c. It randomly picks a suspect card, weapon card, and room card and puts it away in the confidential case file.
    d. The remaining cards are evenly distributed among players
    e. The weapons are placed in different rooms and characters are positioned in the start position.
2. Game play: The game logic enforces the following rules of play:
    a. The players take turns to roll the dice. The game logic determines who plays next.
    b. When the dice is rolled a random number between 2-12 is generated by the system.
    c. The player may move the number of squares using horizontal, vertical, forward or backward moves; diagonal moves are not allowed. The player is not allowed to enter the same square twice on the same turn. The player is not allowed to enter on a square occupied by another character pawn.
    d. The player is allowed to enter a room only through an open door.
    e. The player cannot enter a room if its doorway is blocked by another character pawn. Also, the player cannot leave a room if the doorway is blocked by another pawn.
    f. The player cannot re-enter a room on the same turn.

    g. On entering a room, the player can make a suggestion by moving a Suspect and Weapon into that room.
    h. After the suggestion is made, the opponents try to disprove the suggestion by showing a card from their hand.
    i. In the next round the player may choose to roll the dice and exit the room or to take the secret passageway to another room.
    j. If a player's character pawn was moved to a room while making a suggestion, the player could choose to make a suggestion from that room or roll the dice and exit the room.
3. Making an Accusation: A player can make an accusation on his turn for a Suspect, Weapon and Room without going to that room. The system will show the confidential cards to the player. If the accusation is true, the game ends and the player wins. If the accusation is false, the player cannot play any more but can participate in disproving suggestions.

We have designed the following data structures for implementing the rules of game play. The data structures might have some additions as the details get added.

- Game Board: A 20X20 matrix of tiles
- Struct Tile: Information about a tile in the board
    - Type: Room, Door, Passage, Start position
    - OccupiedBy: Pawn or NULL
- Struct Suspect[6]: Information about a pawn on the board
    - Name
    - Position
    - Room last visited
- Struct Weapon[8]: Information about a weapon on the board
    - Name
    - Position
    - Room
- Struct Room[9]: Information about the room
    - Name
    - Door position
    - Secret passage
    - List of suspects in the room: Index into pawn array
    - List of weapons in the room: Index into weapon array
- AllCards: Array of all available cards
- ConfidentialFile: 3 cards kept separately
- Struct Player[6]:
    - Pawn
    - List of cards
- Struct Card:
    - Type: Suspect/Weapon/Room
    - Name
- Struct Position:
    - X, Y position in the Board matrix

User Interface Design

The user interface area is a 160x480 pixel area on the right side of the game board in the display. The interaction is with the help of the Tab, Enter, and arrow keys on the keyboard. The layout of the UI is as shown in Figure 1.

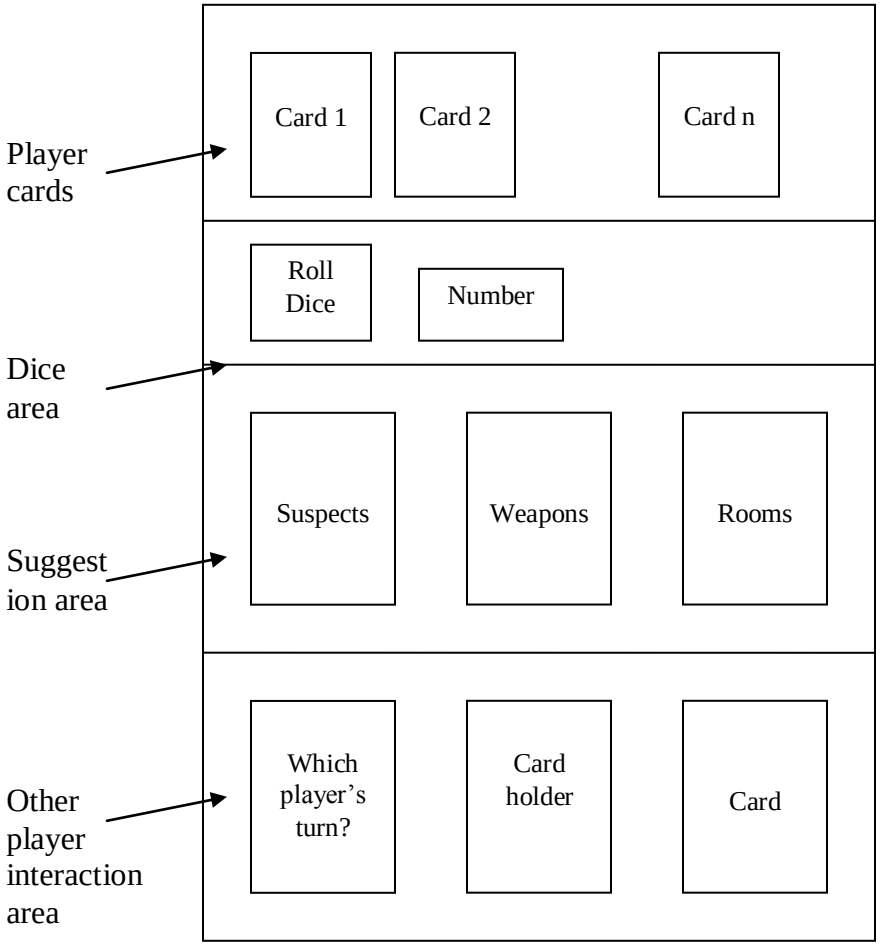

Figure 1 User interface layout