Ethernet Battleship

Marc Howard, Dan Aprahamian, Apoorva Gade, Shihab Hamati

INTRODUCTION

Our project was implementing an Ethernet Battleship game that communicated between a computer and an Altera DE2 board. We implemented drivers and manipulators for several pieces of hardware for this project, as well nearly 2000 lines of game logic code. The hardware we used was VGA, Ethernet, and a PS2 Keyboard. We will discuss each of these in turn.

VGA

Our initial design was as follows: We would use tiles to display the graphics of the game, and an on-board RAM would store tile numbers for each location on screen. The RAM could be written to from NIOS. Meanwhile, a VGA counter would operate, sweeping and drawing each individual pixel. The data value of each pixel would be generated by first looking into the RAM to find the corresponding tile for that location. It would then use this tile number to look up the actual pixel values for that tile, which would be stored in a ROM. The hardware would use that value to drive the VGA signal.

Split Screen

We initially had debate over whether to use 16x16 or 32x32 tiles. On one hand, we wanted to be able to display both small and large versions of the gaming grid. At the same time, constructing large images from 16x16 tiles would be complicated, and would make rotating the image difficult. We eventually decided on a mixed solution: The left side of the screen consisted of a

12x15 grid of 32x32 tile slots. The right side consisted of a 16x30 grid of 16x16 tiles. Using this format, we were able to have a large display on the left side of the screen, with a smaller display on the right side (to accommodate having to look at both your own ships and where you had fired at the enemy. The tiles were stored in memory using the following equations:

- For 32x32 tiles, (row * 12 + column)
- For 16x16 tiles, (row * 16 + column + 180)
    - The +180 accounts for the locations storing 32x32 tiles in memory.

This address conversion was performed in a module present between the VGA counter and the RAM unit.

Tile numbers were represented in memory as follows:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Not used (future development)						HL	FL	RT	Tile #						

Rotate, Flip, and Invert

Instead of duplicating tiles (which would take up memory), we decided to designate bits 7 and 8 of the tile address data to represent rotate (RT) and flip (FL) respectively. Specific hardware was built in that checked these bits. If FL was set, the X and Y pixel coordinates were rotated by 180 degrees. If the RT flag was set, they were rotated an additional 90 degrees. This allowed 4 total rotation angles. Rotation was done in between reading the tile address from the RAM and reading the pixel value from the ROM.

Additionally, we included an additional bit HL, for Highlight/Invert. If this bit was set, then the color values of the pixel data for that tile would be inverted. Initially, this was meant to be a different module after the Tile ROM, but we eventually incorporated it into the Tile ROM to avoid synchronization issues. This inverting effect was used to indicate the location of our cursor.

Tile Data

We initially hoped to store the tile data in a ROM on chip. However, we soon found this to be impossible: we had nearly 250kb of Tile data, and less than 50kb of on-chip memory. Instead of either reducing our graphic detail or implementing complex size-reduction schemes, we decided to instead store all of the tile data in the DE2 SRAM. This necessitated storing the actual NIOS program in the SDRAM (see the SDRAM section). Because we are now storing the data in off-chip SRAM, read times for the data are longer. Because the SRAM is not on our processor clock, this makes it very difficult to ensure that values were properly read and written from it properly. Thus, we incorporated an SRAM interface into our VGA driver. The SRAM interface would take in the tile number and pixel coordinates, convert them into an SRAM address, and send them to the SRAM as a read. In the next clock cycle (in addition to sending over the next address), it would read in the 8 bit color value the SRAM returned, and output it. If the color were meant to be inverted, it would invert the color at this time.

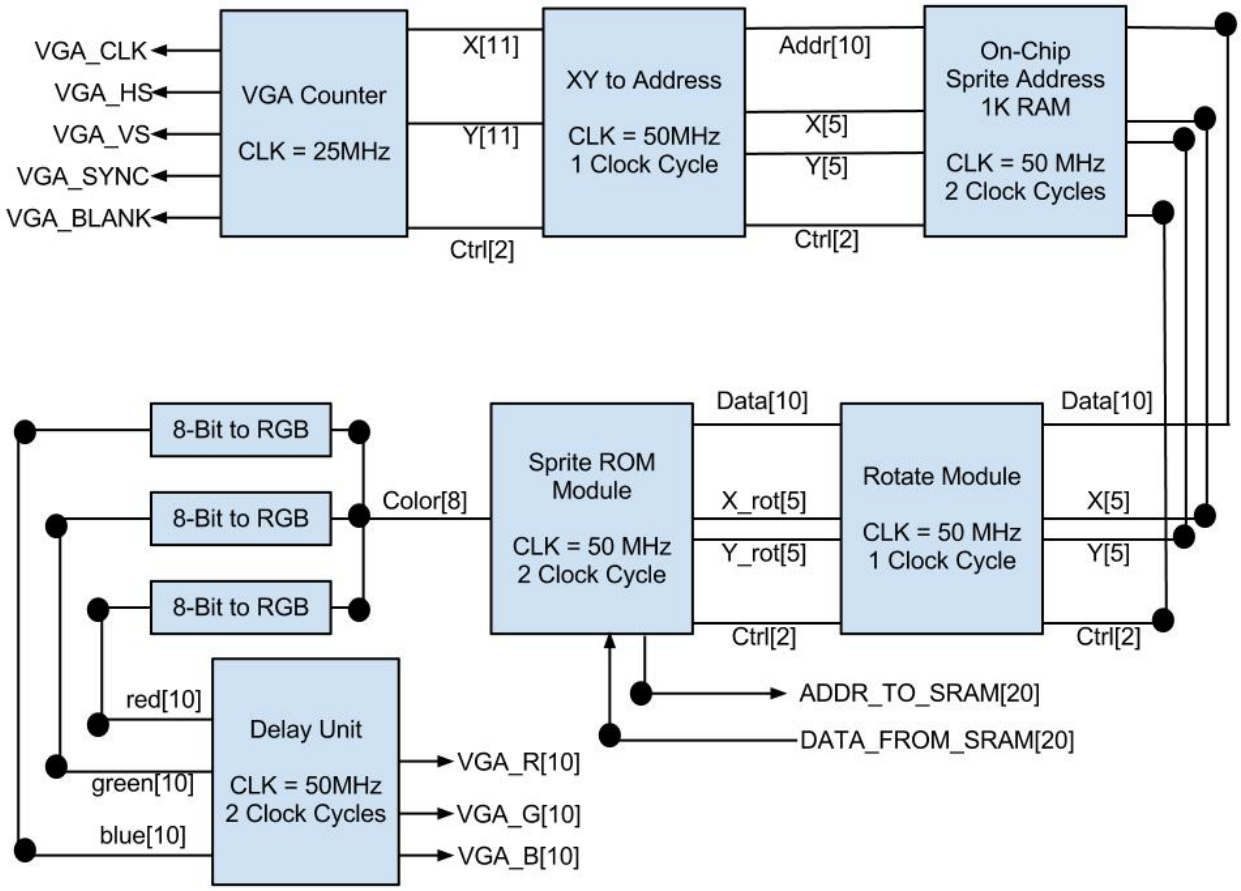
Color Converters and Delays

After the 8-bit color value is determined, it is broken into its color components, and then passed through a combinational logic converter that converts it into a 10-bit value that can then be converted to the VGA_R, VGA_G, and VGA_B signals. However, in order to make sure that values are output at the correct times, two shift registers are added to capture the values and sustain them for the two clock cycles required for VGA.

SDRAM and PLL

Because we needed to use the SDRAM to hold the software program, we needed to develop an SDRAM driver. This driver automatically generated using the SOPC builder. However, we then used a pre-generated Phase Lock Loop to make sure that the SDRAM clock is 3ns ahead of the system clock. This PLL generates 3 signals: a DRAM clock at 50MHz with a phase shift of -3ns, a system clock at 50MHz with a phase shift of 0ns that drives the system clock, and a 25MHz clock with a phase shift of 0ns to drive the Ethernet clock.

Final Design



Note that all synchronization signals are passed through all blocks to ensure full system synchonization.

PS2 KEYBOARD

The PS2 keyboard interfaces with the VGA. It is used to select the target tile for attacking. The Letter A-J are used for addressing one dimension and 0-9 for the other dimension of the 2X2 array. The arrow keys are used to navigate around the array. The hardware used is that given in lab3. C code has been implemented to return strings to the game logic based on the key-code read from the hardware.

ETHERNET

In our project, the DE2 Board runs a version with the game while the MacBook runs another. The two players on these two devices are to communicate via Ethernet cable. This method was chosen because of its ubiquitous use in global networking, and our previous experience with it in lab 2. The protocols used for communicating game messages are IP and the unreliable but simple UDP packets. Although UDP delivery is unreliable compared with TCP, it is good enough when only 2 machines are connected via a dedicated switch in the lab. We first set on developing the Ethernet C code to test and allow communication between the DE2 Board and the MacBook, running the Nios II code on top of the compiled VHDL code provided in lab 2, which includes the Ethernet. We hard wired the MacBook's MAC and IP addresses into our code, and thus we were able to send UDP/IP messages to the MacBook. But the MacBook which tried to communicate through Python refused to send the packets without first sending ARP packets and waiting for an appropriate response. We researched what ARP is – Address Resolution Protocol – and designed an appropriate response packet that will assist the MacBook to map our board's IP address to its physical MAC address.

<table>
<tr><td colspan="2">Hardware type</td><td>Protocol type</td></tr>
<tr><td>HW addr lth</td><td>P addr lth</td><td>Opcode</td></tr>
<tr><td colspan="3">Source hardware address</td></tr>
<tr><td colspan="3">Source protocol address</td></tr>
<tr><td colspan="3">Destination hardware address</td></tr>
<tr><td colspan="3">Destination protocol address</td></tr>
</table>

ARP message

But even establishing the ARP protocol, we noticed that the first few UDP messages were not sent by our board. This was resolved by allowing the DM9000A some delay time to initialize its configuration before we asked it to send a packet. Note that in our game code, every sent packet starts with a keyword, and the code will continuously attempt to receive packets until it gets one starting with the correct keyword for that state in the program. If a packet is received with an error according to the checksum, it is ignored and the game keeps waiting for a proper packet. Since our packets had a consistent amount of garbling, which caused the game to wait forever for a proper packet. When this check was removed, the game did work somewhat, but the message

was occasionally garbled, causing the same result. This could be corrected by implementing a full TCP stack which would ensure delivery of well-formed error-free messages, but that was beyond the scope of our project.

After establishing proper communication through our software for Milestone 2, we then integrated the VHDL component for the on-board DM9000A Ethernet controller with our other components. We imported the component into the SOPC builder and connected the appropriate pins to the Avalon bus. An interrupt pin from the controller was crucial to indicate to the Nios II that the controller received a packet. However, this task was rather troublesome. We ran into a few problems, some of which we solved, and some which sadly remain buggy. The first problem we noticed was that the board was not responsive. This was solved by allowing some time before we set the "reset_n" signal in the top level VHDL code to high. Another problem was that our Ethernet packet was sometimes garbled at a bit level. This was solved by using a replacing our combinational logic which introduced a delay into a phase locked loop to generate a 25 MHz clock from the 50 MHz clock on board. Nevertheless, the messages received still have some bits which are not received correctly and cause problems in receiving the proper packets sent from the MacBook to the DE2 Board.

GAME CODE

The game logic code was implemented on the computer in python and then converted to C for the DE2 board. The only difference between the two sets of code is a flag that determines which goes first (DE2 board goes first and computer goes second) and some basic syntax. The only major difference is a few functions that interact with hardware. These were implemented differently on the computer than on the DE2 board because of the different hardware on each system. The wrapper functions for interacting with these hardware elements are the same for both programs, and are included in the appendices following this report.

On the computer the game code written in Python, using the built in libraries for keyboard, the socket library for Ethernet, the Tkinter library for display and the PIL library for image manipulation. The version of Python used was 2.7. Some Java was also used for turning the tile images into a text file of a format readable by the SRAM on the DE2.

The basic game logic is as follows:

1. Get name from user (PS2 Input)
2. Have user place ships
3. Exchange names with opponent (Ethernet)
4. Take a turn – Select a square and fire a shot (Ethernet)
5. Wait for a shot from opponent and respond (Ethernet)
6. Repeat 4 and 5 until one player reaches 17 Hits
7. Ask if player wants to play again. If so go to 2, if not go to 1.

This code also works between two computers running Python if the global turn variable is set to 0 on one computer and 1 on the other computer (to determine who goes first). There may be Ethernet bugs running the programs on varying network types, as this program does rely on UDP, which can be unreliable. Likewise, two DE2 boards can also play together using this same method, as long as Ethernet problems are also kept in mind.

It would not be logical or practical to delve too deep into the logic of the game code here since there are almost 2000 lines of it, but it is included in the appendices after this report. It is worth noting, however, that the game code does require the folder called ”Tiles” (which is in our project tarball) to be in the same directory as Battleship.py when it is run in order to give it access to the images. It is also worth noting that the python libraries that have been mentioned must be installed on any computers running the game.

OVERVIEW OF PERSONAL RESPONSIBILITIES

In this project, there were several key components that needed to be completed in order to make our game work. We needed the VGA driver, Image manipulation, Ethernet, PS2 Keyboard, and game logic for board and computer. The breakdown of work for this project turned out to be slightly uneven due to people’s skill sets and available time, and this was reflected in both our ultimate result and in our team spirit.

Dan developed the VGA driver and Image manipulation hardware and was our general VHDL problem solver. When someone else could not come up with a solution to a problem, or figure

out exactly what was not working, he was always eager and able to help. He was a key member of the group, and critical to the success of the project.

Apoorva was in charge of the PS2 Keyboard. She used the VHDL from lab 3, and wrote a small amount of C code. Her code was an implementation of the hardware wrapper function she was given by Marc (wait for an input and then return a string containing the data from a single keystroke). She usually came to meetings and always cheerful, and occasionally helped to debug other people's parts of the project.

Shihab was in charge of the Ethernet. He used the Ethernet VHDL from lab 2 and edited a small amount of lab 2 C code in order to make the Ethernet function in accordance with the hardware wrapper function he was given by Marc. He successfully debugged several problems the Ethernet had, but did not give it much attention after Milestone 2 was working until the final stages of the project. The Ethernet ended up being our weakest element, and the only real reason our game was not entirely successful. Shihab had good communication while working with the other team members to debug the Ethernet, and was very friendly.

Marc was the team leader and managed the entire project. He had a large part in the design and implementation of every element, and especially in making sure that all of the elements worked together properly. In addition to helping design and debug the components, he designed all of the graphics and designed and wrote all of the game code for both computer and DE2 board. He also produced most of the literature surrounding this project, as well as making sure that the team met regularly and that all deadlines were met both in accordance with the syllabus and to the professor's satisfaction.

CONCLUSION

Our game was a great success given the group dynamics and the experience that we all had entering this class. It works well between any combinations of the 2 platforms and displays good working functionality of all of the hardware elements that we used, with only slight problems in the Ethernet. We all learned a lot and had a great time making this game a reality.

APPENDICIES: Code and other important files

Hardware Function Headers:

```
//Graphics functions

//Sets the value of that square on the screen to the data provided
void changeTile32(int row, int col, int tileNumber, int rotate, int flip, int invert)

//Sets the value of that square on the screen to the data provided
void changeTile16(int row, int col, int tileNumber, int rotate, int flip, int invert)

//Reads the data from that square on the screen, and writes it back inverted.
void invert32(int row, int col)

//Reads the data from that square on the screen, and writes it back inverted.
void invert16(int row, int col)


//PS2 Functions

//Get a character from the keyboard.  Return the character.  If arrow key, backspace, enter, or
escape, return the string of that name.  (ie.  "Left")
char* getInput( )


//Ethernet Functions

//Send a packet with a message contained in a null terminated char* input
void sendPacket(char* message)

char* receivePacket()
```

Game code (python):

```
#Ethernet Battleship
#Written by Marc Howard - Spring 2012

import numpy as np
import Tkinter as tk
import socket
from PIL import Image, ImageTk
import PIL.ImageOps

display32 = np.zeros((15,12,4), int)
display16 = np.zeros((30,16,4), int)
playerScore = 0
opponentScore = 0
turn = 1
cursorX = 0
cursorY = 0
displaySetting = 0 #0 = player ships big, opponent ships small;  1 = opponent ships big, player ships small
exitGame = 0  #0 = keep going, 1 = exit game
exitPlayer = 1 #0 = Keep same player, 1 = New player
exitRound = 0  #0 = keep going, 1 = exit round
playerName = "Your Name"
opponentName = "Their Name"
sockSend = socket.socket( socket.AF_INET, # Internet
                  socket.SOCK_DGRAM ) # UDP

sockReceive = socket.socket( socket.AF_INET, # Internet
                                                socket.SOCK_DGRAM ) # UDP

root = tk.Tk()
canvas = tk.Canvas(root, width = 640, height = 480)
canvas.pack()
tkimages = []
tiles32 = []
tiles16 = []
inputKey = ""
patrolboatleft = 2
submarineleft = 3
destroyerleft = 3
battleshipleft = 4
carrierleft = 5


def main():
        global exitPlayer
```

```
	global exitGame

	root.bind_all('<Key>', key)
	print( "I have bound keys")
	initializeNetworking("192.168.1.1", "192.168.1.2", 5005)
	print("I have initialized networking")


	sendPacket("Test")
	#receivePacket()
	#receivePacket()

	initializeDisplayData()
	print("I have imported data")
	while exitGame == 0:
		print("I have determined that the game is not over")
		if exitPlayer == 1:
			print("I am setting up a game for new players")
			setupGame()
		print "I am going to play a new round now"
		playGame()
		print "I am going to check if the player wants to play again"
		checkPlayAgain()
	print "I am going to quit the game now"
	quitGame()

	return

#Setup a game
def setupGame():
	global playerName, exitGame, exitRound, tkimages
	tkimages = []
	print "I am initializing the display now"
	initializeDisplay()
	exitGame = 0
	exitRound = 0
	playerName = getPlayerName()
	return

#Play a game.  Initialize the round, then play a round.
def playGame():
	global exitPlayer
	global exitRound
	global turn

	initializeRound() #This sets up the ships and waits until both players are ready
```

```
	while exitRound == 0:
		#Play the game.  exitRound should be set to 1 if a player wins, 2 if a player quits the game using the quit condition (implement), 3 if the game goes out of sync. or 4 if another error occurs
		if turn == 0: #If players turn, enact that.
			playerTurn()
			turn = 1; #Set turn to opponent.
		elif turn == 1: #If opponent turn, enact that.
			opponentTurn()
			turn = 0 #Set turn to player.
		else: #Error condition.  Should never occur.
			exitRound = 4
		if playerScore >= 17 or opponentScore >= 17:
			exitRound = 1
	#End of game loop

	#If game ended in victory
	if exitRound == 1:
		if playerScore == 17: #Player win condition.  Enact player victory.
			playerVictory()
		elif opponentScore == 17: #Opponent win condition.  Enact opponent victory.
			opponentVictory()

	#If player chose to exit game.
	elif exitRound == 2:
		print("Player quit game.  Goodbye!")
		sendPacket("Player Quit")

	#If game falls out of sync.
	elif exitRound == 3:
		print("Game out of sync.  Game ending.  Sorry!")
		sendPacket("Out of Sync Quit")

	#If other error condition occurs
	else:
		print ("Game error.  Game ending.  Sorry!")
		sendPacket("Game Error Quit")
	return

#Exit the game, closing the socket and game window
def quitGame():
	sockSend.close()
	sockReceive.close()
	#Close game window.
	return
```

```
#Set up the basic screen display, with a blank ocean and player name for both screens.
def initializeDisplay():
	#Display the 32 side
	#Display border on the 32 side
	for c in range (0,5):
		changeTile32(0,c,8,0,0,0)
	for r in range (0,15):
		changeTile32(r,0,8,0,0,0)
	for c in range (0,12):
		changeTile32(14,c,8,0,0,0)

	#Display the Battleship Title on the 32 side
	for i in range (0,7):
		changeTile32(0,5+i,109+i,0,0,0)

	#Display the battleship title on the 16 side
	for c in range (0,3):
		changeTile16(0,0 + (2*c),112 + (4*c),0,0,0)
		changeTile16(1,0 + (2*c),114 + (4*c),0,0,0)
		changeTile16(0,1 + (2*c),113 + (4*c),0,0,0)
		changeTile16(1,1 + (2*c),115 + (4*c),0,0,0)

	for c in range (3,8):
		changeTile16(0,0 + (2*c),8,0,0,0)
		changeTile16(1,0 + (2*c),10,0,0,0)
		changeTile16(0,1 + (2*c),9,0,0,0)
		changeTile16(1,1 + (2*c),11,0,0,0)

	#Display the blank text squares on the 32 side
	for r in range (1,3):
		for c in range (1,12):
			changeTile32(r,c,11,0,0,0)
	changeTile32(3,1,11,0,0,0)

	#Display the numbers along the top of the battle screen on the 32 side
	for c in range (0,10):
		changeTile32(3,2+c,65+c,0,0,0)

	#Display the letters along the side of the battle screen on the 32 side
	for r in range (0,10):
		changeTile32(4+r,1,12+r,0,0,0)

	#Display the ocean on the 32 side
	for r in range (0,10):
		for c in range (0,10):
```

```
                changeTile32(r+4,c+2,5,0,0,0)

        #Display the 16 side
        #Display the border on the 16 side.  Fill the entire side with border.  This will be
overwritten as necessary.
        for r in range (1,15):
                for c in range (0,8):
                        changeTile16(r*2+0,0 + (2*c),8,0,0,0)
                        changeTile16(r*2+1,0 + (2*c),10,0,0,0)
                        changeTile16(r*2+0,1 + (2*c),9,0,0,0)
                        changeTile16(r*2+1,1 + (2*c),11,0,0,0)

        #Display the blank text squares on the 16 side
        for r in range (0,2):
                for c in range (0,11):
                        changeTile16(r+3, c+2,14,0,0,0)
                        changeTile16(r+19, c+2,14,0,0,0)
                        changeTile16(r+22, c+2,14,0,0,0)
                        changeTile16(r+25, c+2,14,0,0,0)
        changeTile16(6, 2,14,0,0,0)

        #Display the numbers along the top of the battle screen on the 16 side
        for c in range (0,10):
                changeTile16(6,3+c,68+c,0,0,0)

        #Display the letters along the side of the battle screen on the 16 side
        for r in range (0,10):
                changeTile16(7+r,2,15+r,0,0,0)

        #Display the ocean on the 16 side
        for r in range (0,10):
                for c in range (0,10):
                        changeTile16(r+7,c+3,5,0,0,0)

        return

#Sets the value of that square on the screen to the data provided
def changeTile32(row, col, tileNumber, rotate, flip, invert):
        display32[row,col,0] = tileNumber
        display32[row,col,1] = rotate
        display32[row,col,2] = flip
        display32[row,col,3] = invert
        if row > 14 or col > 11 or row < 0 or col < 0:
                print "Invalid row and column:"
                print row
                print col
```

```
            return
        if tileNumber > 117 or tileNumber < 0:
            print "Invalid Tile Number:"
            print tileNumber
        image = tiles32[tileNumber]
        if rotate and flip:
            image = image.rotate(270)
        elif flip:
            image = image.rotate(180)
        elif rotate:
            image = image.rotate(90)
        if invert:
            image = PIL.ImageOps.invert(image)
        tkimage = ImageTk.PhotoImage(image)
        tkimages.append(tkimage)
        item = canvas.create_image(col*32, row*32, image = tkimage, anchor = 'nw')
        root.update()
        return

#Sets the value of that square on the screen to the data provided
def changeTile16(row, col, tileNumber, rotate, flip, invert):
        display16[row,col,0] = tileNumber
        display16[row,col,1] = rotate
        display16[row,col,2] = flip
        display16[row,col,3] = invert
        if row > 29 or col > 15 or row < 0 or col < 0:
            print "Invalid row and column:"
            print row
            print col
            return
        if tileNumber > 125 or tileNumber < 0:
            print "Invalid Tile Number:"
            print tileNumber
        image = tiles16[tileNumber]
        if rotate and flip:
            image = image.rotate(270)
        elif flip:
            image = image.rotate(180)
        elif rotate:
            image = image.rotate(90)
        if invert:
            image = PIL.ImageOps.invert(image)
        tkimage = ImageTk.PhotoImage(image)
        tkimages.append(tkimage)
        item = canvas.create_image(12*32 + col*16, row*16, image = tkimage, anchor = 'nw')
        root.update()
```

```
        return

#Invert a square on the 32 side
def invert32(row, col):
        if display32[row, col, 3] == 0:
                display32[row, col, 3] = 1
        elif display32[row, col, 3] == 1:
                display32[row, col, 3] = 0
        changeTile32(row, col, display32[row,col,0], display32[row,col,1], display32[row,col,2],
display32[row,col,3])
        return


#Invert a square on the 16 side
def invert16(row, col):
        if display16[row, col, 3] == 0:
                display16[row, col, 3] = 1
        elif display16[row, col, 3] == 1:
                display16[row, col, 3] = 0
        changeTile16(row, col, display16[row,col,0], display16[row,col,1], display16[row,col,2],
display16[row,col,3])
        return


#Display a string on the 32 side
def displayString32(string, row, col):
        l = len(string)
        for i in range (0, l):
                displayChar32(string[i], row, col + i)
        return

#Display a string on the 16 side
def displayString16(string, row, col):
        l = len(string)
        for i in range (0, l):
                displayChar16(string[i], row, col + i)
        return

# Setup the Network interface
def initializeNetworking(IP_ADDRESS_OTHER, IP_ADDRESS_SELF, PORT):
        sockSend.connect((IP_ADDRESS_OTHER, PORT))
        sockReceive.bind((IP_ADDRESS_SELF, PORT))
        return

#Send a packet with a string message.
def sendPacket(message):
```

```
	sockSend.send(message)
	return

#Receive a packet with a string message
def receivePacket():
	data, addr = sockReceive.recvfrom( 1024 ) # buffer size is 1024 bytes
	print "received message:", data
	return data

#Check if the player wants to play again.
def checkPlayAgain():
	global exitPlayer, cursorX, cursorY
	displayString32("Play Again?", 1, 1)
	displayString32("    Y N    ", 2, 1)
	cursorX = 5
	cursorY = 2
	invert32(cursorY, cursorX)
	exitPlayer = -1
	while  exitPlayer == -1:
		input = getInput()
		if input == "Left" and cursorX == 7:
			invert32(cursorY, cursorX)
			cursorX = 5
			invert32(cursorY, cursorX)
		elif input == "Right" and cursorX == 5:
			invert32(cursorY, cursorX)
			cursorX = 7
			invert32(cursorY, cursorX)
		elif input == "Return":
			invert32(cursorY, cursorX)
			if cursorX == 5:
				exitPlayer = 0
			else:
				exitPlayer = 1
	return

#Display a character at a coordinate on the 32 side
def displayChar32(string, row, col):
	char = ""
	if string == "A":
		char = string
		changeTile32(row, col, 12, 0, 0, 0)
	elif string == "B":
		char = string
		changeTile32(row, col, 13, 0, 0, 0)
	elif string == "C":
```

```
        char = string
        changeTile32(row, col, 14, 0, 0, 0)
elif string == "D":
        char = string
        changeTile32(row, col, 15, 0, 0, 0)
elif string == "E":
        char = string
        changeTile32(row, col, 16, 0, 0, 0)
elif string == "F":
        char = string
        changeTile32(row, col, 17, 0, 0, 0)
elif string == "G":
        char = string
        changeTile32(row, col, 18, 0, 0, 0)
elif string == "H":
        char = string
        changeTile32(row, col, 19, 0, 0, 0)
elif string == "I":
        char = string
        changeTile32(row, col, 20, 0, 0, 0)
elif string == "J":
        char = string
        changeTile32(row, col, 21, 0, 0, 0)
elif string == "K":
        char = string
        changeTile32(row, col, 22, 0, 0, 0)
elif string == "L":
        char = string
        changeTile32(row, col, 23, 0, 0, 0)
elif string == "M":
        char = string
        changeTile32(row, col, 24, 0, 0, 0)
elif string == "N":
        char = string
        changeTile32(row, col, 25, 0, 0, 0)
elif string == "O":
        char = string
        changeTile32(row, col, 26, 0, 0, 0)
elif string == "P":
        char = string
        changeTile32(row, col, 27, 0, 0, 0)
elif string == "Q":
        char = string
        changeTile32(row, col, 28, 0, 0, 0)
elif string == "R":
        char = string
```

```
        changeTile32(row, col, 29, 0, 0, 0)
elif string == "S":
        char = string
        changeTile32(row, col, 30, 0, 0, 0)
elif string == "T":
        char = string
        changeTile32(row, col, 31, 0, 0, 0)
elif string == "U":
        char = string
        changeTile32(row, col, 32, 0, 0, 0)
elif string == "V":
        char = string
        changeTile32(row, col, 33, 0, 0, 0)
elif string == "W":
        char = string
        changeTile32(row, col, 34, 0, 0, 0)
elif string == "X":
        char = string
        changeTile32(row, col, 35, 0, 0, 0)
elif string == "Y":
        char = string
        changeTile32(row, col, 36, 0, 0, 0)
elif string == "Z":
        char = string
        changeTile32(row, col, 37, 0, 0, 0)
elif string == "a":
        char = string
        changeTile32(row, col, 38, 0, 0, 0)
elif string == "b":
        char = string
        changeTile32(row, col, 39, 0, 0, 0)
elif string == "c":
        char = string
        changeTile32(row, col, 40, 0, 0, 0)
elif string == "d":
        char = string
        changeTile32(row, col, 41, 0, 0, 0)
elif string == "e":
        char = string
        changeTile32(row, col, 42, 0, 0, 0)
elif string == "f":
        char = string
        changeTile32(row, col, 43, 0, 0, 0)
elif string == "g":
        char = string
        changeTile32(row, col, 44, 0, 0, 0)
```

```
elif string == "h":
        char = string
        changeTile32(row, col, 45, 0, 0, 0)
elif string == "i":
        char = string
        changeTile32(row, col, 46, 0, 0, 0)
elif string == "j":
        char = string
        changeTile32(row, col, 47, 0, 0, 0)
elif string == "k":
        char = string
        changeTile32(row, col, 48, 0, 0, 0)
elif string == "l":
        char = string
        changeTile32(row, col, 49, 0, 0, 0)
elif string == "m":
        char = string
        changeTile32(row, col, 50, 0, 0, 0)
elif string == "n":
        char = string
        changeTile32(row, col, 51, 0, 0, 0)
elif string == "o":
        char = string
        changeTile32(row, col, 52, 0, 0, 0)
elif string == "p":
        char = string
        changeTile32(row, col, 53, 0, 0, 0)
elif string == "q":
        char = string
        changeTile32(row, col, 54, 0, 0, 0)
elif string == "r":
        char = string
        changeTile32(row, col, 55, 0, 0, 0)
elif string == "s":
        char = string
        changeTile32(row, col, 56, 0, 0, 0)
elif string == "t":
        char = string
        changeTile32(row, col, 57, 0, 0, 0)
elif string == "u":
        char = string
        changeTile32(row, col, 58, 0, 0, 0)
elif string == "v":
        char = string
        changeTile32(row, col, 59, 0, 0, 0)
elif string == "w":
```

```
        char = string
        changeTile32(row, col, 60, 0, 0, 0)
elif string == "x":
        char = string
        changeTile32(row, col, 61, 0, 0, 0)
elif string == "y":
        char = string
        changeTile32(row, col, 62, 0, 0, 0)
elif string == "z":
        char = string
        changeTile32(row, col, 63, 0, 0, 0)
elif string == "0":
        char = string
        changeTile32(row, col, 64, 0, 0, 0)
elif string == "1":
        char = string
        changeTile32(row, col, 65, 0, 0, 0)
elif string == "2":
        char = string
        changeTile32(row, col, 66, 0, 0, 0)
elif string == "3":
        char = string
        changeTile32(row, col, 67, 0, 0, 0)
elif string == "4":
        char = string
        changeTile32(row, col, 68, 0, 0, 0)
elif string == "5":
        char = string
        changeTile32(row, col, 69, 0, 0, 0)
elif string == "6":
        char = string
        changeTile32(row, col, 70, 0, 0, 0)
elif string == "7":
        char = string
        changeTile32(row, col, 71, 0, 0, 0)
elif string == "8":
        char = string
        changeTile32(row, col, 72, 0, 0, 0)
elif string == "9":
        char = string
        changeTile32(row, col, 73, 0, 0, 0)
elif string == "10":
        char = string
        changeTile32(row, col, 74, 0, 0, 0)
elif string == ":":
        char = string
```

```
                changeTile32(row, col, 116, 0, 0, 0)
        elif string == "?":
                char = string
                changeTile32(row, col, 117, 0, 0, 0)
        elif string == " ":
                char = string
                changeTile32(row, col, 11, 0, 0, 0)
        else:
                changeTile32(row, col, 11, 0, 0, 0)
        return char

#Display a character at a coordinate on the 16 side
def displayChar16(string, row, col):
        char = ""
        if string == "A":
                char = string
                changeTile16(row, col, 3+12, 0, 0, 0)
        elif string == "B":
                char = string
                changeTile16(row, col, 3+13, 0, 0, 0)
        elif string == "C":
                char = string
                changeTile16(row, col, 3+14, 0, 0, 0)
        elif string == "D":
                char = string
                changeTile16(row, col, 3+15, 0, 0, 0)
        elif string == "E":
                char = string
                changeTile16(row, col, 3+16, 0, 0, 0)
        elif string == "F":
                char = string
                changeTile16(row, col, 3+17, 0, 0, 0)
        elif string == "G":
                char = string
                changeTile16(row, col, 3+18, 0, 0, 0)
        elif string == "H":
                char = string
                changeTile16(row, col, 3+19, 0, 0, 0)
        elif string == "I":
                char = string
                changeTile16(row, col, 3+20, 0, 0, 0)
        elif string == "J":
                char = string
                changeTile16(row, col, 3+21, 0, 0, 0)
        elif string == "K":
                char = string
```

```
        changeTile16(row, col, 3+22, 0, 0, 0)
elif string == "L":
        char = string
        changeTile16(row, col, 3+23, 0, 0, 0)
elif string == "M":
        char = string
        changeTile16(row, col, 3+24, 0, 0, 0)
elif string == "N":
        char = string
        changeTile16(row, col, 3+25, 0, 0, 0)
elif string == "O":
        char = string
        changeTile16(row, col, 3+26, 0, 0, 0)
elif string == "P":
        char = string
        changeTile16(row, col, 3+27, 0, 0, 0)
elif string == "Q":
        char = string
        changeTile16(row, col, 3+28, 0, 0, 0)
elif string == "R":
        char = string
        changeTile16(row, col, 3+29, 0, 0, 0)
elif string == "S":
        char = string
        changeTile16(row, col, 3+30, 0, 0, 0)
elif string == "T":
        char = string
        changeTile16(row, col, 3+31, 0, 0, 0)
elif string == "U":
        char = string
        changeTile16(row, col, 3+32, 0, 0, 0)
elif string == "V":
        char = string
        changeTile16(row, col, 3+33, 0, 0, 0)
elif string == "W":
        char = string
        changeTile16(row, col, 3+34, 0, 0, 0)
elif string == "X":
        char = string
        changeTile16(row, col, 3+35, 0, 0, 0)
elif string == "Y":
        char = string
        changeTile16(row, col, 3+36, 0, 0, 0)
elif string == "Z":
        char = string
        changeTile16(row, col, 3+37, 0, 0, 0)
```

```
elif string == "a":
        char = string
        changeTile16(row, col, 3+38, 0, 0, 0)
elif string == "b":
        char = string
        changeTile16(row, col, 3+39, 0, 0, 0)
elif string == "c":
        char = string
        changeTile16(row, col, 3+40, 0, 0, 0)
elif string == "d":
        char = string
        changeTile16(row, col, 3+41, 0, 0, 0)
elif string == "e":
        char = string
        changeTile16(row, col, 3+42, 0, 0, 0)
elif string == "f":
        char = string
        changeTile16(row, col, 3+43, 0, 0, 0)
elif string == "g":
        char = string
        changeTile16(row, col, 3+44, 0, 0, 0)
elif string == "h":
        char = string
        changeTile16(row, col, 3+45, 0, 0, 0)
elif string == "i":
        char = string
        changeTile16(row, col, 3+46, 0, 0, 0)
elif string == "j":
        char = string
        changeTile16(row, col, 3+47, 0, 0, 0)
elif string == "k":
        char = string
        changeTile16(row, col, 3+48, 0, 0, 0)
elif string == "l":
        char = string
        changeTile16(row, col, 3+49, 0, 0, 0)
elif string == "m":
        char = string
        changeTile16(row, col, 3+50, 0, 0, 0)
elif string == "n":
        char = string
        changeTile16(row, col, 3+51, 0, 0, 0)
elif string == "o":
        char = string
        changeTile16(row, col, 3+52, 0, 0, 0)
elif string == "p":
```

```
        char = string
        changeTile16(row, col, 3+53, 0, 0, 0)
elif string == "q":
        char = string
        changeTile16(row, col, 3+54, 0, 0, 0)
elif string == "r":
        char = string
        changeTile16(row, col, 3+55, 0, 0, 0)
elif string == "s":
        char = string
        changeTile16(row, col, 3+56, 0, 0, 0)
elif string == "t":
        char = string
        changeTile16(row, col, 3+57, 0, 0, 0)
elif string == "u":
        char = string
        changeTile16(row, col, 3+58, 0, 0, 0)
elif string == "v":
        char = string
        changeTile16(row, col, 3+59, 0, 0, 0)
elif string == "w":
        char = string
        changeTile16(row, col, 3+60, 0, 0, 0)
elif string == "x":
        char = string
        changeTile16(row, col, 3+61, 0, 0, 0)
elif string == "y":
        char = string
        changeTile16(row, col, 3+62, 0, 0, 0)
elif string == "z":
        char = string
        changeTile16(row, col, 3+63, 0, 0, 0)
elif string == "0":
        char = string
        changeTile16(row, col, 3+64, 0, 0, 0)
elif string == "1":
        char = string
        changeTile16(row, col, 3+65, 0, 0, 0)
elif string == "2":
        char = string
        changeTile16(row, col, 3+66, 0, 0, 0)
elif string == "3":
        char = string
        changeTile16(row, col, 3+67, 0, 0, 0)
elif string == "4":
        char = string
```

```
                changeTile16(row, col, 3+68, 0, 0, 0)
        elif string == "5":
                char = string
                changeTile16(row, col, 3+69, 0, 0, 0)
        elif string == "6":
                char = string
                changeTile16(row, col, 3+70, 0, 0, 0)
        elif string == "7":
                char = string
                changeTile16(row, col, 3+71, 0, 0, 0)
        elif string == "8":
                char = string
                changeTile16(row, col, 3+72, 0, 0, 0)
        elif string == "9":
                char = string
                changeTile16(row, col, 3+73, 0, 0, 0)
        elif string == "10":
                char = string
                changeTile16(row, col, 3+74, 0, 0, 0)
        elif string == ":":
                char = string
                changeTile16(row, col, 124, 0, 0, 0)
        elif string == "?":
                char = string
                changeTile16(row, col, 125, 0, 0, 0)
        elif string == " ":
                char = string
                changeTile16(row, col, 14, 0, 0, 0)
        else:
                changeTile16(row, col, 14, 0, 0, 0)
        return char

#Get the players name and display it
def getPlayerName():
        #Set the cursor to the beginning of the ENTERNAME box
        cursorX = 1
        cursorY = 2
        invert32(cursorY, cursorX)
        name = ""
        displayString32("Enter Name:", 1, 1)
        root.update()
        input = ""
        input = getInput()
        while input != "Return":
                print(input)
                if input == "BackSpace" and cursorX > 1:
```

```
                        if (cursorX < 12):
                                invert32(cursorY, cursorX)
                        cursorX = cursorX - 1
                        changeTile32(cursorY, cursorX, 11, 0, 0, 0)
                        invert32(cursorY, cursorX)
                        name = name[0:cursorX-1]
                elif cursorX <= 11 and input != "BackSpace" and input != "Left" and input !=
"Right" and input != "Up" and input != "Down" and input != "Escape":
                        name = name + displayChar32(input, cursorY, cursorX)
                        cursorX = cursorX + 1
                        if (cursorX < 12):
                                invert32(cursorY, cursorX)
                input = getInput()
        print "Name:", name
        return name

#Every time a key is pressed it gets set to inputKey
def key(event):
        global inputKey
        inputKey = ""
        #print("I have noticed a key event!")
        #shows key or tk code for the key
        #print(event.keysym)
        if event.char == event.keysym:
        # normal number and letter characters
                inputKey = event.char
        if event.keysym == 'Escape' or event.keysym == 'Left' or event.keysym == 'Right' or
event.keysym == 'Up' or event.keysym == 'Down' or event.keysym == 'Return' or event.keysym
== 'BackSpace' or event.keysym == 'space':
                inputKey = event.keysym
        #print("inputKey has been changed to:")
        #print(inputKeyGLOBAL[0])
        return

#Get a key from the inputList queue
def getInput():
        global inputKey
        #print("I am attempting to get a key from the user")
        while inputKey == "":
                root.update()
        input = inputKey
        if inputKey == "space":
                input == " "
        inputKey = ""
        #print("I have successfully gotten a keystroke")
        #print(input)
```

```
        return input

#Load the tiles from the disk
def initializeDisplayData():
        tiles32.append(Image.open("./Tiles/32x32 Tiles/Black 32.bmp"))
        tiles32.append(Image.open("./Tiles/32x32 Tiles/Red 32.bmp"))
        tiles32.append(Image.open("./Tiles/32x32 Tiles/Green 32.bmp"))
        tiles32.append(Image.open("./Tiles/32x32 Tiles/Blue 32.bmp"))
        tiles32.append(Image.open("./Tiles/32x32 Tiles/White 32.bmp"))
        tiles32.append(Image.open("./Tiles/32x32 Tiles/Basic Ocean 32.bmp"))
        tiles32.append(Image.open("./Tiles/32x32 Tiles/Hit Beacon 32.bmp"))
        tiles32.append(Image.open("./Tiles/32x32 Tiles/Miss Beacon 32.bmp"))
        tiles32.append(Image.open("./Tiles/32x32 Tiles/Border 32.bmp"))
        tiles32.append(Image.open("./Tiles/32x32 Tiles/Explosion 32.bmp"))
        tiles32.append(Image.open("./Tiles/32x32 Tiles/Splash Ocean 32.bmp"))
        tiles32.append(Image.open("./Tiles/32x32 Tiles/Space 32.bmp"))
        tiles32.append(Image.open("./Tiles/32x32 Tiles/A U 32.bmp"))
        tiles32.append(Image.open("./Tiles/32x32 Tiles/B U 32.bmp"))
        tiles32.append(Image.open("./Tiles/32x32 Tiles/C U 32.bmp"))
        tiles32.append(Image.open("./Tiles/32x32 Tiles/D U 32.bmp"))
        tiles32.append(Image.open("./Tiles/32x32 Tiles/E U 32.bmp"))
        tiles32.append(Image.open("./Tiles/32x32 Tiles/F U 32.bmp"))
        tiles32.append(Image.open("./Tiles/32x32 Tiles/G U 32.bmp"))
        tiles32.append(Image.open("./Tiles/32x32 Tiles/H U 32.bmp"))
        tiles32.append(Image.open("./Tiles/32x32 Tiles/I U 32.bmp"))
        tiles32.append(Image.open("./Tiles/32x32 Tiles/J U 32.bmp"))
        tiles32.append(Image.open("./Tiles/32x32 Tiles/K U 32.bmp"))
        tiles32.append(Image.open("./Tiles/32x32 Tiles/L U 32.bmp"))
        tiles32.append(Image.open("./Tiles/32x32 Tiles/M U 32.bmp"))
        tiles32.append(Image.open("./Tiles/32x32 Tiles/N U 32.bmp"))
        tiles32.append(Image.open("./Tiles/32x32 Tiles/O U 32.bmp"))
        tiles32.append(Image.open("./Tiles/32x32 Tiles/P U 32.bmp"))
        tiles32.append(Image.open("./Tiles/32x32 Tiles/Q U 32.bmp"))
        tiles32.append(Image.open("./Tiles/32x32 Tiles/R U 32.bmp"))
        tiles32.append(Image.open("./Tiles/32x32 Tiles/S U 32.bmp"))
        tiles32.append(Image.open("./Tiles/32x32 Tiles/T U 32.bmp"))
        tiles32.append(Image.open("./Tiles/32x32 Tiles/U U 32.bmp"))
        tiles32.append(Image.open("./Tiles/32x32 Tiles/V U 32.bmp"))
        tiles32.append(Image.open("./Tiles/32x32 Tiles/W U 32.bmp"))
        tiles32.append(Image.open("./Tiles/32x32 Tiles/X U 32.bmp"))
        tiles32.append(Image.open("./Tiles/32x32 Tiles/Y U 32.bmp"))
        tiles32.append(Image.open("./Tiles/32x32 Tiles/Z U 32.bmp"))
        tiles32.append(Image.open("./Tiles/32x32 Tiles/A L 32.bmp"))
        tiles32.append(Image.open("./Tiles/32x32 Tiles/B L 32.bmp"))
        tiles32.append(Image.open("./Tiles/32x32 Tiles/C L 32.bmp"))
        tiles32.append(Image.open("./Tiles/32x32 Tiles/D L 32.bmp"))
```

```
tiles32.append(Image.open("./Tiles/32x32 Tiles/E L 32.bmp"))
tiles32.append(Image.open("./Tiles/32x32 Tiles/F L 32.bmp"))
tiles32.append(Image.open("./Tiles/32x32 Tiles/G L 32.bmp"))
tiles32.append(Image.open("./Tiles/32x32 Tiles/H L 32.bmp"))
tiles32.append(Image.open("./Tiles/32x32 Tiles/I L 32.bmp"))
tiles32.append(Image.open("./Tiles/32x32 Tiles/J L 32.bmp"))
tiles32.append(Image.open("./Tiles/32x32 Tiles/K L 32.bmp"))
tiles32.append(Image.open("./Tiles/32x32 Tiles/L L 32.bmp"))
tiles32.append(Image.open("./Tiles/32x32 Tiles/M L 32.bmp"))
tiles32.append(Image.open("./Tiles/32x32 Tiles/N L 32.bmp"))
tiles32.append(Image.open("./Tiles/32x32 Tiles/O L 32.bmp"))
tiles32.append(Image.open("./Tiles/32x32 Tiles/P L 32.bmp"))
tiles32.append(Image.open("./Tiles/32x32 Tiles/Q L 32.bmp"))
tiles32.append(Image.open("./Tiles/32x32 Tiles/R L 32.bmp"))
tiles32.append(Image.open("./Tiles/32x32 Tiles/S L 32.bmp"))
tiles32.append(Image.open("./Tiles/32x32 Tiles/T L 32.bmp"))
tiles32.append(Image.open("./Tiles/32x32 Tiles/U L 32.bmp"))
tiles32.append(Image.open("./Tiles/32x32 Tiles/V L 32.bmp"))
tiles32.append(Image.open("./Tiles/32x32 Tiles/W L 32.bmp"))
tiles32.append(Image.open("./Tiles/32x32 Tiles/X L 32.bmp"))
tiles32.append(Image.open("./Tiles/32x32 Tiles/Y L 32.bmp"))
tiles32.append(Image.open("./Tiles/32x32 Tiles/Z L 32.bmp"))
tiles32.append(Image.open("./Tiles/32x32 Tiles/0 32.bmp"))
tiles32.append(Image.open("./Tiles/32x32 Tiles/1 32.bmp"))
tiles32.append(Image.open("./Tiles/32x32 Tiles/2 32.bmp"))
tiles32.append(Image.open("./Tiles/32x32 Tiles/3 32.bmp"))
tiles32.append(Image.open("./Tiles/32x32 Tiles/4 32.bmp"))
tiles32.append(Image.open("./Tiles/32x32 Tiles/5 32.bmp"))
tiles32.append(Image.open("./Tiles/32x32 Tiles/6 32.bmp"))
tiles32.append(Image.open("./Tiles/32x32 Tiles/7 32.bmp"))
tiles32.append(Image.open("./Tiles/32x32 Tiles/8 32.bmp"))
tiles32.append(Image.open("./Tiles/32x32 Tiles/9 32.bmp"))
tiles32.append(Image.open("./Tiles/32x32 Tiles/10 32.bmp"))
tiles32.append(Image.open("./Tiles/32x32 Tiles/Aircraft Carrier 1 32.bmp"))
tiles32.append(Image.open("./Tiles/32x32 Tiles/Aircraft Carrier 2 32.bmp"))
tiles32.append(Image.open("./Tiles/32x32 Tiles/Aircraft Carrier 3 32.bmp"))
tiles32.append(Image.open("./Tiles/32x32 Tiles/Aircraft Carrier 4 32.bmp"))
tiles32.append(Image.open("./Tiles/32x32 Tiles/Aircraft Carrier 5 32.bmp"))
tiles32.append(Image.open("./Tiles/32x32 Tiles/Battleship 1 32.bmp"))
tiles32.append(Image.open("./Tiles/32x32 Tiles/Battleship 2 32.bmp"))
tiles32.append(Image.open("./Tiles/32x32 Tiles/Battleship 3 32.bmp"))
tiles32.append(Image.open("./Tiles/32x32 Tiles/Battleship 4 32.bmp"))
tiles32.append(Image.open("./Tiles/32x32 Tiles/Destroyer 1 32.bmp"))
tiles32.append(Image.open("./Tiles/32x32 Tiles/Destroyer 2 32.bmp"))
tiles32.append(Image.open("./Tiles/32x32 Tiles/Destroyer 3 32.bmp"))
tiles32.append(Image.open("./Tiles/32x32 Tiles/Submarine 1 32.bmp"))
```

```
tiles32.append(Image.open("./Tiles/32x32 Tiles/Submarine 2 32.bmp"))
tiles32.append(Image.open("./Tiles/32x32 Tiles/Submarine 3 32.bmp"))
tiles32.append(Image.open("./Tiles/32x32 Tiles/Patrol Boat 1 32.bmp"))
tiles32.append(Image.open("./Tiles/32x32 Tiles/Patrol Boat 2 32.bmp"))
tiles32.append(Image.open("./Tiles/32x32 Tiles/Aircraft Carrier Hit 1 32.bmp"))
tiles32.append(Image.open("./Tiles/32x32 Tiles/Aircraft Carrier Hit 2 32.bmp"))
tiles32.append(Image.open("./Tiles/32x32 Tiles/Aircraft Carrier Hit 3 32.bmp"))
tiles32.append(Image.open("./Tiles/32x32 Tiles/Aircraft Carrier Hit 4 32.bmp"))
tiles32.append(Image.open("./Tiles/32x32 Tiles/Aircraft Carrier Hit 5 32.bmp"))
tiles32.append(Image.open("./Tiles/32x32 Tiles/Battleship Hit 1 32.bmp"))
tiles32.append(Image.open("./Tiles/32x32 Tiles/Battleship Hit 2 32.bmp"))
tiles32.append(Image.open("./Tiles/32x32 Tiles/Battleship Hit 3 32.bmp"))
tiles32.append(Image.open("./Tiles/32x32 Tiles/Battleship Hit 4 32.bmp"))
tiles32.append(Image.open("./Tiles/32x32 Tiles/Destroyer Hit 1 32.bmp"))
tiles32.append(Image.open("./Tiles/32x32 Tiles/Destroyer Hit 2 32.bmp"))
tiles32.append(Image.open("./Tiles/32x32 Tiles/Destroyer Hit 3 32.bmp"))
tiles32.append(Image.open("./Tiles/32x32 Tiles/Submarine Hit 1 32.bmp"))
tiles32.append(Image.open("./Tiles/32x32 Tiles/Submarine Hit 2 32.bmp"))
tiles32.append(Image.open("./Tiles/32x32 Tiles/Submarine Hit 3 32.bmp"))
tiles32.append(Image.open("./Tiles/32x32 Tiles/Patrol Boat Hit 1 32.bmp"))
tiles32.append(Image.open("./Tiles/32x32 Tiles/Patrol Boat Hit 2 32.bmp"))
tiles32.append(Image.open("./Tiles/32x32 Tiles/Battleship Title 1 32.bmp"))
tiles32.append(Image.open("./Tiles/32x32 Tiles/Battleship Title 2 32.bmp"))
tiles32.append(Image.open("./Tiles/32x32 Tiles/Battleship Title 3 32.bmp"))
tiles32.append(Image.open("./Tiles/32x32 Tiles/Battleship Title 4 32.bmp"))
tiles32.append(Image.open("./Tiles/32x32 Tiles/Battleship Title 5 32.bmp"))
tiles32.append(Image.open("./Tiles/32x32 Tiles/Battleship Title 6 32.bmp"))
tiles32.append(Image.open("./Tiles/32x32 Tiles/Battleship Title 7 32.bmp"))
tiles32.append(Image.open("./Tiles/32x32 Tiles/Colon 32.bmp"))
tiles32.append(Image.open("./Tiles/32x32 Tiles/Question 32.bmp"))
tiles16.append(Image.open("./Tiles/16x16 Tiles/Black 16.bmp"))
tiles16.append(Image.open("./Tiles/16x16 Tiles/Red 16.bmp"))
tiles16.append(Image.open("./Tiles/16x16 Tiles/Green 16.bmp"))
tiles16.append(Image.open("./Tiles/16x16 Tiles/Blue 16.bmp"))
tiles16.append(Image.open("./Tiles/16x16 Tiles/White 16.bmp"))
tiles16.append(Image.open("./Tiles/16x16 Tiles/Basic Ocean 16.bmp"))
tiles16.append(Image.open("./Tiles/16x16 Tiles/Hit Beacon 16.bmp"))
tiles16.append(Image.open("./Tiles/16x16 Tiles/Miss Beacon 16.bmp"))
tiles16.append(Image.open("./Tiles/16x16 Tiles/Border Top Left 16.bmp"))
tiles16.append(Image.open("./Tiles/16x16 Tiles/Border Top Right 16.bmp"))
tiles16.append(Image.open("./Tiles/16x16 Tiles/Border Bottom Left 16.bmp"))
tiles16.append(Image.open("./Tiles/16x16 Tiles/Border Bottom Right 16.bmp"))
tiles16.append(Image.open("./Tiles/16x16 Tiles/Explosion 16.bmp"))
tiles16.append(Image.open("./Tiles/16x16 Tiles/Splash Ocean 16.bmp"))
tiles16.append(Image.open("./Tiles/16x16 Tiles/Space 16.bmp"))
tiles16.append(Image.open("./Tiles/16x16 Tiles/A U 16.bmp"))
```

```
tiles16.append(Image.open("./Tiles/16x16 Tiles/B U 16.bmp"))
tiles16.append(Image.open("./Tiles/16x16 Tiles/C U 16.bmp"))
tiles16.append(Image.open("./Tiles/16x16 Tiles/D U 16.bmp"))
tiles16.append(Image.open("./Tiles/16x16 Tiles/E U 16.bmp"))
tiles16.append(Image.open("./Tiles/16x16 Tiles/F U 16.bmp"))
tiles16.append(Image.open("./Tiles/16x16 Tiles/G U 16.bmp"))
tiles16.append(Image.open("./Tiles/16x16 Tiles/H U 16.bmp"))
tiles16.append(Image.open("./Tiles/16x16 Tiles/I U 16.bmp"))
tiles16.append(Image.open("./Tiles/16x16 Tiles/J U 16.bmp"))
tiles16.append(Image.open("./Tiles/16x16 Tiles/K U 16.bmp"))
tiles16.append(Image.open("./Tiles/16x16 Tiles/L U 16.bmp"))
tiles16.append(Image.open("./Tiles/16x16 Tiles/M U 16.bmp"))
tiles16.append(Image.open("./Tiles/16x16 Tiles/N U 16.bmp"))
tiles16.append(Image.open("./Tiles/16x16 Tiles/O U 16.bmp"))
tiles16.append(Image.open("./Tiles/16x16 Tiles/P U 16.bmp"))
tiles16.append(Image.open("./Tiles/16x16 Tiles/Q U 16.bmp"))
tiles16.append(Image.open("./Tiles/16x16 Tiles/R U 16.bmp"))
tiles16.append(Image.open("./Tiles/16x16 Tiles/S U 16.bmp"))
tiles16.append(Image.open("./Tiles/16x16 Tiles/T U 16.bmp"))
tiles16.append(Image.open("./Tiles/16x16 Tiles/U U 16.bmp"))
tiles16.append(Image.open("./Tiles/16x16 Tiles/V U 16.bmp"))
tiles16.append(Image.open("./Tiles/16x16 Tiles/W U 16.bmp"))
tiles16.append(Image.open("./Tiles/16x16 Tiles/X U 16.bmp"))
tiles16.append(Image.open("./Tiles/16x16 Tiles/Y U 16.bmp"))
tiles16.append(Image.open("./Tiles/16x16 Tiles/Z U 16.bmp"))
tiles16.append(Image.open("./Tiles/16x16 Tiles/A L 16.bmp"))
tiles16.append(Image.open("./Tiles/16x16 Tiles/B L 16.bmp"))
tiles16.append(Image.open("./Tiles/16x16 Tiles/C L 16.bmp"))
tiles16.append(Image.open("./Tiles/16x16 Tiles/D L 16.bmp"))
tiles16.append(Image.open("./Tiles/16x16 Tiles/E L 16.bmp"))
tiles16.append(Image.open("./Tiles/16x16 Tiles/F L 16.bmp"))
tiles16.append(Image.open("./Tiles/16x16 Tiles/G L 16.bmp"))
tiles16.append(Image.open("./Tiles/16x16 Tiles/H L 16.bmp"))
tiles16.append(Image.open("./Tiles/16x16 Tiles/I L 16.bmp"))
tiles16.append(Image.open("./Tiles/16x16 Tiles/J L 16.bmp"))
tiles16.append(Image.open("./Tiles/16x16 Tiles/K L 16.bmp"))
tiles16.append(Image.open("./Tiles/16x16 Tiles/L L 16.bmp"))
tiles16.append(Image.open("./Tiles/16x16 Tiles/M L 16.bmp"))
tiles16.append(Image.open("./Tiles/16x16 Tiles/N L 16.bmp"))
tiles16.append(Image.open("./Tiles/16x16 Tiles/O L 16.bmp"))
tiles16.append(Image.open("./Tiles/16x16 Tiles/P L 16.bmp"))
tiles16.append(Image.open("./Tiles/16x16 Tiles/Q L 16.bmp"))
tiles16.append(Image.open("./Tiles/16x16 Tiles/R L 16.bmp"))
tiles16.append(Image.open("./Tiles/16x16 Tiles/S L 16.bmp"))
tiles16.append(Image.open("./Tiles/16x16 Tiles/T L 16.bmp"))
tiles16.append(Image.open("./Tiles/16x16 Tiles/U L 16.bmp"))
```

```
tiles16.append(Image.open("./Tiles/16x16 Tiles/V L 16.bmp"))
tiles16.append(Image.open("./Tiles/16x16 Tiles/W L 16.bmp"))
tiles16.append(Image.open("./Tiles/16x16 Tiles/X L 16.bmp"))
tiles16.append(Image.open("./Tiles/16x16 Tiles/Y L 16.bmp"))
tiles16.append(Image.open("./Tiles/16x16 Tiles/Z L 16.bmp"))
tiles16.append(Image.open("./Tiles/16x16 Tiles/0 16.bmp"))
tiles16.append(Image.open("./Tiles/16x16 Tiles/1 16.bmp"))
tiles16.append(Image.open("./Tiles/16x16 Tiles/2 16.bmp"))
tiles16.append(Image.open("./Tiles/16x16 Tiles/3 16.bmp"))
tiles16.append(Image.open("./Tiles/16x16 Tiles/4 16.bmp"))
tiles16.append(Image.open("./Tiles/16x16 Tiles/5 16.bmp"))
tiles16.append(Image.open("./Tiles/16x16 Tiles/6 16.bmp"))
tiles16.append(Image.open("./Tiles/16x16 Tiles/7 16.bmp"))
tiles16.append(Image.open("./Tiles/16x16 Tiles/8 16.bmp"))
tiles16.append(Image.open("./Tiles/16x16 Tiles/9 16.bmp"))
tiles16.append(Image.open("./Tiles/16x16 Tiles/10 16.bmp"))
tiles16.append(Image.open("./Tiles/16x16 Tiles/Aircraft Carrier 1 16.bmp"))
tiles16.append(Image.open("./Tiles/16x16 Tiles/Aircraft Carrier 2 16.bmp"))
tiles16.append(Image.open("./Tiles/16x16 Tiles/Aircraft Carrier 3 16.bmp"))
tiles16.append(Image.open("./Tiles/16x16 Tiles/Aircraft Carrier 4 16.bmp"))
tiles16.append(Image.open("./Tiles/16x16 Tiles/Aircraft Carrier 5 16.bmp"))
tiles16.append(Image.open("./Tiles/16x16 Tiles/Battleship 1 16.bmp"))
tiles16.append(Image.open("./Tiles/16x16 Tiles/Battleship 2 16.bmp"))
tiles16.append(Image.open("./Tiles/16x16 Tiles/Battleship 3 16.bmp"))
tiles16.append(Image.open("./Tiles/16x16 Tiles/Battleship 4 16.bmp"))
tiles16.append(Image.open("./Tiles/16x16 Tiles/Destroyer 1 16.bmp"))
tiles16.append(Image.open("./Tiles/16x16 Tiles/Destroyer 2 16.bmp"))
tiles16.append(Image.open("./Tiles/16x16 Tiles/Destroyer 3 16.bmp"))
tiles16.append(Image.open("./Tiles/16x16 Tiles/Submarine 1 16.bmp"))
tiles16.append(Image.open("./Tiles/16x16 Tiles/Submarine 2 16.bmp"))
tiles16.append(Image.open("./Tiles/16x16 Tiles/Submarine 3 16.bmp"))
tiles16.append(Image.open("./Tiles/16x16 Tiles/Patrol Boat 1 16.bmp"))
tiles16.append(Image.open("./Tiles/16x16 Tiles/Patrol Boat 2 16.bmp"))
tiles16.append(Image.open("./Tiles/16x16 Tiles/Aircraft Carrier Hit 1 16.bmp"))
tiles16.append(Image.open("./Tiles/16x16 Tiles/Aircraft Carrier Hit 2 16.bmp"))
tiles16.append(Image.open("./Tiles/16x16 Tiles/Aircraft Carrier Hit 3 16.bmp"))
tiles16.append(Image.open("./Tiles/16x16 Tiles/Aircraft Carrier Hit 4 16.bmp"))
tiles16.append(Image.open("./Tiles/16x16 Tiles/Aircraft Carrier Hit 5 16.bmp"))
tiles16.append(Image.open("./Tiles/16x16 Tiles/Battleship Hit 1 16.bmp"))
tiles16.append(Image.open("./Tiles/16x16 Tiles/Battleship Hit 2 16.bmp"))
tiles16.append(Image.open("./Tiles/16x16 Tiles/Battleship Hit 3 16.bmp"))
tiles16.append(Image.open("./Tiles/16x16 Tiles/Battleship Hit 4 16.bmp"))
tiles16.append(Image.open("./Tiles/16x16 Tiles/Destroyer Hit 1 16.bmp"))
tiles16.append(Image.open("./Tiles/16x16 Tiles/Destroyer Hit 2 16.bmp"))
tiles16.append(Image.open("./Tiles/16x16 Tiles/Destroyer Hit 3 16.bmp"))
tiles16.append(Image.open("./Tiles/16x16 Tiles/Submarine Hit 1 16.bmp"))
```

```
        tiles16.append(Image.open("./Tiles/16x16 Tiles/Submarine Hit 2 16.bmp"))
        tiles16.append(Image.open("./Tiles/16x16 Tiles/Submarine Hit 3 16.bmp"))
        tiles16.append(Image.open("./Tiles/16x16 Tiles/Patrol Boat Hit 1 16.bmp"))
        tiles16.append(Image.open("./Tiles/16x16 Tiles/Patrol Boat Hit 2 16.bmp"))
        tiles16.append(Image.open("./Tiles/16x16 Tiles/Battleship Title 8 Upper Left 16.bmp"))
        tiles16.append(Image.open("./Tiles/16x16 Tiles/Battleship Title 8 Upper Right 16.bmp"))
        tiles16.append(Image.open("./Tiles/16x16 Tiles/Battleship Title 8 Lower Left 16.bmp"))
        tiles16.append(Image.open("./Tiles/16x16 Tiles/Battleship Title 8 Lower Right
16.bmp"))
        tiles16.append(Image.open("./Tiles/16x16 Tiles/Battleship Title 9 Upper Left 16.bmp"))
        tiles16.append(Image.open("./Tiles/16x16 Tiles/Battleship Title 9 Upper Right 16.bmp"))
        tiles16.append(Image.open("./Tiles/16x16 Tiles/Battleship Title 9 Lower Left 16.bmp"))
        tiles16.append(Image.open("./Tiles/16x16 Tiles/Battleship Title 9 Lower Right
16.bmp"))
        tiles16.append(Image.open("./Tiles/16x16 Tiles/Battleship Title 10 Upper Left 16.bmp"))
        tiles16.append(Image.open("./Tiles/16x16 Tiles/Battleship Title 10 Upper Right
16.bmp"))
        tiles16.append(Image.open("./Tiles/16x16 Tiles/Battleship Title 10 Lower Left 16.bmp"))
        tiles16.append(Image.open("./Tiles/16x16 Tiles/Battleship Title 10 Lower Right
16.bmp"))
        tiles16.append(Image.open("./Tiles/16x16 Tiles/Colon 16.bmp"))
        tiles16.append(Image.open("./Tiles/16x16 Tiles/Question 16.bmp"))
        return


#Get the players ship positions and record them, and determine which player goes first.
def initializeRound():
        global playerName, opponentName, cursorX, cursorY, patrolboatleft, submarineleft,
destroyerleft, battleshipleft, carrierleft, playerScore, opponentScore

        #Initialize score and ship health

        patrolboatleft = 2
        submarineleft = 3
        destroyerleft = 3
        battleshipleft = 4
        carrierleft = 5
        playerScore = 0
        opponentScore = 0


        #Place the Carrier
        displayString32("Place 5 sq ", 1, 1)
        displayString32("Carrier    ", 2, 1)
        cursorY = 4
        cursorX = 2
```

```
	carrier = 0
	print ("I am now going to try to get a location for the carrier")
	while carrier == 0:
		getLocation32()
		print "I got a location: ", cursorX, ", ", cursorY
		if checkValidShip(cursorX, cursorY, 5, "Left") or checkValidShip(cursorX,
cursorY, 5, "Right") or checkValidShip(cursorX, cursorY, 5, "Up") or checkValidShip(cursorX,
cursorY, 5, "Down"):
			carrier = 1
		else:
			print "Invalid placement for carrier"
			invert32(cursorY, cursorX)

	print "Valid location for carrier"
	carrier = 0
	displayString32("Direction? ", 1, 1)
	displayString32("Length: 5  ", 2, 1)
	while carrier == 0:
		input = getInput()
		if input == "Left" and checkValidShip(cursorX, cursorY, 5, "Left"):
			placeCarrier(cursorX, cursorY, "Left")
			carrier = 1
		if input == "Right" and checkValidShip(cursorX, cursorY, 5, "Right"):
			placeCarrier(cursorX, cursorY, "Right")
			carrier = 1
		if input == "Up" and checkValidShip(cursorX, cursorY, 5, "Up"):
			placeCarrier(cursorX, cursorY, "Up")
			carrier = 1
		if input == "Down" and checkValidShip(cursorX, cursorY, 5, "Down"):
			placeCarrier(cursorX, cursorY, "Down")
			carrier = 1
	print "You placed a carrier!"

	#Place the Battleship
	displayString32("Place 4 sq ", 1, 1)
	displayString32("Battleship ", 2, 1)
	cursorY = 4
	cursorX = 2
	battleship = 0
	print ("I am now going to try to get a location for the battleship")
	while battleship == 0:
		getLocation32()
		print "I got a location: ", cursorX, ", ", cursorY
		if checkValidShip(cursorX, cursorY, 4, "Left") or checkValidShip(cursorX,
cursorY, 4, "Right") or checkValidShip(cursorX, cursorY, 4, "Up") or checkValidShip(cursorX,
cursorY, 4, "Down"):
```

```
                battleship = 1
            else:
                print "Invalid placement for battleship"
                invert32(cursorY, cursorX)

        print "Valid location for battleship"
        battleship = 0
        displayString32("Direction? ", 1, 1)
        displayString32("Length: 4  ", 2, 1)
        while battleship == 0:
            input = getInput()
            if input == "Left" and checkValidShip(cursorX, cursorY, 4, "Left"):
                placeBattleship(cursorX, cursorY, "Left")
                battleship = 1
            if input == "Right" and checkValidShip(cursorX, cursorY, 4, "Right"):
                placeBattleship(cursorX, cursorY, "Right")
                battleship = 1
            if input == "Up" and checkValidShip(cursorX, cursorY, 4, "Up"):
                placeBattleship(cursorX, cursorY, "Up")
                battleship = 1
            if input == "Down" and checkValidShip(cursorX, cursorY, 4, "Down"):
                placeBattleship(cursorX, cursorY, "Down")
                battleship = 1
        print "You placed a battleship!"

        #Place the Destroyer
        displayString32("Place 3 sq ", 1, 1)
        displayString32("Destroyer  ", 2, 1)
        cursorY = 4
        cursorX = 2
        destroyer = 0
        print ("I am now going to try to get a location for the destroyer")
        while destroyer == 0:
            getLocation32()
            print "I got a location: ", cursorX, ", ", cursorY
            if checkValidShip(cursorX, cursorY, 3, "Left") or checkValidShip(cursorX,
cursorY, 3, "Right") or checkValidShip(cursorX, cursorY, 3, "Up") or checkValidShip(cursorX,
cursorY, 3, "Down"):
                destroyer = 1
            else:
                print "Invalid placement for destroyer"
                invert32(cursorY, cursorX)

        print "Valid location for destroyer"
        destroyer = 0
        displayString32("Direction? ", 1, 1)
```

```
	displayString32("Length: 3  ", 2, 1)
	while destroyer == 0:
		input = getInput()
		if input == "Left" and checkValidShip(cursorX, cursorY, 3, "Left"):
			placeDestroyer(cursorX, cursorY, "Left")
			destroyer = 1
		if input == "Right" and checkValidShip(cursorX, cursorY, 3, "Right"):
			placeDestroyer(cursorX, cursorY, "Right")
			destroyer = 1
		if input == "Up" and checkValidShip(cursorX, cursorY, 3, "Up"):
			placeDestroyer(cursorX, cursorY, "Up")
			destroyer = 1
		if input == "Down" and checkValidShip(cursorX, cursorY, 3, "Down"):
			placeDestroyer(cursorX, cursorY, "Down")
			destroyer = 1
	print "You placed a destroyer!"

	#Place the Submarine
	displayString32("Place 3 sq ", 1, 1)
	displayString32("Submarine  ", 2, 1)
	cursorY = 4
	cursorX = 2
	submarine = 0
	print ("I am now going to try to get a location for the submarine")
	while submarine == 0:
		getLocation32()
		print "I got a location: ", cursorX, ", ", cursorY
		if checkValidShip(cursorX, cursorY, 3, "Left") or checkValidShip(cursorX,
cursorY, 3, "Right") or checkValidShip(cursorX, cursorY, 3, "Up") or checkValidShip(cursorX,
cursorY, 3, "Down"):
			submarine = 1
		else:
			print "Invalid placement for submarine"
			invert32(cursorY, cursorX)

	print "Valid location for submarine"
	submarine = 0
	displayString32("Direction? ", 1, 1)
	displayString32("Length: 3  ", 2, 1)
	while submarine == 0:
		input = getInput()
		if input == "Left" and checkValidShip(cursorX, cursorY, 3, "Left"):
			placeSubmarine(cursorX, cursorY, "Left")
			submarine = 1
		if input == "Right" and checkValidShip(cursorX, cursorY, 3, "Right"):
			placeSubmarine(cursorX, cursorY, "Right")
```

```
                submarine = 1
            if input == "Up" and checkValidShip(cursorX, cursorY, 3, "Up"):
                placeSubmarine(cursorX, cursorY, "Up")
                submarine = 1
            if input == "Down" and checkValidShip(cursorX, cursorY, 3, "Down"):
                placeSubmarine(cursorX, cursorY, "Down")
                submarine = 1
        print "You placed a submarine!"

        #Place the Patrol Boat
        displayString32("Place 2 sq ", 1, 1)
        displayString32("Patrol Boat", 2, 1)
        cursorY = 4
        cursorX = 2
        patrolboat = 0
        print ("I am now going to try to get a location for the patrol boat")
        while patrolboat == 0:
            getLocation32()
            print "I got a location: ", cursorX, ", ", cursorY
            if checkValidShip(cursorX, cursorY, 2, "Left") or checkValidShip(cursorX,
cursorY, 2, "Right") or checkValidShip(cursorX, cursorY, 2, "Up") or checkValidShip(cursorX,
cursorY, 2, "Down"):
                patrolboat = 1
            else:
                print "Invalid placement for patrol boat"
                invert32(cursorY, cursorX)

        print "Valid location for patrol boat"
        patrolboat = 0
        displayString32("Direction? ", 1, 1)
        displayString32("Length: 2  ", 2, 1)
        while patrolboat == 0:
            input = getInput()
            if input == "Left" and checkValidShip(cursorX, cursorY, 2, "Left"):
                placePatrolBoat(cursorX, cursorY, "Left")
                patrolboat = 1
            if input == "Right" and checkValidShip(cursorX, cursorY, 2, "Right"):
                placePatrolBoat(cursorX, cursorY, "Right")
                patrolboat = 1
            if input == "Up" and checkValidShip(cursorX, cursorY, 2, "Up"):
                placePatrolBoat(cursorX, cursorY, "Up")
                patrolboat = 1
            if input == "Down" and checkValidShip(cursorX, cursorY, 2, "Down"):
                placePatrolBoat(cursorX, cursorY, "Down")
                patrolboat = 1
        print "You placed a patrol boat!"
```

```
	displayString16(playerName, 19, 2)
	displayString16(opponentName, 22, 2)
	displayString16("00 Hits", 20, 4)
	displayString16("00 Hits", 23, 4)

	#Exchange Ethernet Handshake with other computer.  Send playerName and ready signal, and wait for both to come back.

	#sendPacket("Player Name " + playerName)
	#sendPacket("Ready")
	#message1 = receivePacket()
	#receivedName = 0
	#while receivedName == 0:
		#if len(message1) >= 12 and message1[0:12] != "Player Name":
			#opponentName = message1[12:]
			#receivedName = 1
		#else:
			#message1 = receivePacket()
	#message2 = receivePacket()
	#receivedReady = 0
	#while receivedReady == 0:
		#if message2 == "Ready":
			#receivedReady = 1
		#else:
			#message2 = receivePacket()

	#displayString16(playerName, 19, 2)
	#displayString16(opponentName, 22, 2)
	return

def checkValidShip(x, y, length, direction):
	valid = 1
	if direction == "Left":
		if x-length >= 1:
			for i in range (x-length, x):
				if display32[y, i+1, 0] != 5:
					valid = 0
		else:
			valid = 0

	elif direction == "Right":
		if x+length <= 12:
			for i in range (x, x+length):
				if display32[y, i, 0] != 5:
					valid = 0
```

```
                else:
                        valid = 0

        elif direction == "Up":
                if y - length >= 3:
                        for i in range (y-length, y):
                                if display32[i+1, x, 0] != 5:
                                        valid = 0
                else:
                        valid = 0

        elif direction == "Down":
                if y + length <= 14:
                        for i in range (y, y+length):
                                if display32[i, x, 0] != 5:
                                        valid = 0
                else:
                        valid = 0

        return valid

def placeCarrier(x, y, direction):
        if direction == "Left":
                changeTile32(y, x, 75, 1, 1, 0)
                changeTile32(y, x-1, 76, 1, 1, 0)
                changeTile32(y, x-2, 77, 1, 1, 0)
                changeTile32(y, x-3, 78, 1, 1, 0)
                changeTile32(y, x-4, 79, 1, 1, 0)

        elif direction == "Right":
                changeTile32(y, x, 75, 1, 0, 0)
                changeTile32(y, x+1, 76, 1, 0, 0)
                changeTile32(y, x+2, 77, 1, 0, 0)
                changeTile32(y, x+3, 78, 1, 0, 0)
                changeTile32(y, x+4, 79, 1, 0, 0)

        elif direction == "Up":
                changeTile32(y, x, 75, 0, 1, 0)
                changeTile32(y-1, x, 76, 0, 1, 0)
                changeTile32(y-2, x, 77, 0, 1, 0)
                changeTile32(y-3, x, 78, 0, 1, 0)
                changeTile32(y-4, x, 79, 0, 1, 0)

        elif direction == "Down":
                changeTile32(y, x, 75, 0, 0, 0)
                changeTile32(y+1, x, 76, 0, 0, 0)
```

```
                changeTile32(y+2, x, 77, 0, 0, 0)
                changeTile32(y+3, x, 78, 0, 0, 0)
                changeTile32(y+4, x, 79, 0, 0, 0)
        return

def placeBattleship(x, y, direction):
        if direction == "Left":
                changeTile32(y, x, 80, 1, 1, 0)
                changeTile32(y, x-1, 81, 1, 1, 0)
                changeTile32(y, x-2, 82, 1, 1, 0)
                changeTile32(y, x-3, 83, 1, 1, 0)

        elif direction == "Right":
                changeTile32(y, x, 80, 1, 0, 0)
                changeTile32(y, x+1, 81, 1, 0, 0)
                changeTile32(y, x+2, 82, 1, 0, 0)
                changeTile32(y, x+3, 83, 1, 0, 0)

        elif direction == "Up":
                changeTile32(y, x, 80, 0, 1, 0)
                changeTile32(y-1, x, 81, 0, 1, 0)
                changeTile32(y-2, x, 82, 0, 1, 0)
                changeTile32(y-3, x, 83, 0, 1, 0)

        elif direction == "Down":
                changeTile32(y, x, 80, 0, 0, 0)
                changeTile32(y+1, x, 81, 0, 0, 0)
                changeTile32(y+2, x, 82, 0, 0, 0)
                changeTile32(y+3, x, 83, 0, 0, 0)
        return

def placeDestroyer(x, y, direction):
        if direction == "Left":
                changeTile32(y, x, 84, 1, 1, 0)
                changeTile32(y, x-1, 85, 1, 1, 0)
                changeTile32(y, x-2, 86, 1, 1, 0)

        elif direction == "Right":
                changeTile32(y, x, 84, 1, 0, 0)
                changeTile32(y, x+1, 85, 1, 0, 0)
                changeTile32(y, x+2, 86, 1, 0, 0)

        elif direction == "Up":
                changeTile32(y, x, 84, 0, 1, 0)
                changeTile32(y-1, x, 85, 0, 1, 0)
                changeTile32(y-2, x, 86, 0, 1, 0)
```

```
        elif direction == "Down":
                changeTile32(y, x, 84, 0, 0, 0)
                changeTile32(y+1, x, 85, 0, 0, 0)
                changeTile32(y+2, x, 86, 0, 0, 0)
        return

def placeSubmarine(x, y, direction):
        if direction == "Left":
                changeTile32(y, x, 87, 1, 1, 0)
                changeTile32(y, x-1, 88, 1, 1, 0)
                changeTile32(y, x-2, 89, 1, 1, 0)

        elif direction == "Right":
                changeTile32(y, x, 87, 1, 0, 0)
                changeTile32(y, x+1, 88, 1, 0, 0)
                changeTile32(y, x+2, 89, 1, 0, 0)

        elif direction == "Up":
                changeTile32(y, x, 87, 0, 1, 0)
                changeTile32(y-1, x, 88, 0, 1, 0)
                changeTile32(y-2, x, 89, 0, 1, 0)

        elif direction == "Down":
                changeTile32(y, x, 87, 0, 0, 0)
                changeTile32(y+1, x, 88, 0, 0, 0)
                changeTile32(y+2, x, 89, 0, 0, 0)
        return

def placePatrolBoat(x, y, direction):
        if direction == "Left":
                changeTile32(y, x, 90, 1, 1, 0)
                changeTile32(y, x-1, 91, 1, 1, 0)

        elif direction == "Right":
                changeTile32(y, x, 90, 1, 0, 0)
                changeTile32(y, x+1, 91, 1, 0, 0)

        elif direction == "Up":
                changeTile32(y, x, 90, 0, 1, 0)
                changeTile32(y-1, x, 91, 0, 1, 0)

        elif direction == "Down":
                changeTile32(y, x, 90, 0, 0, 0)
                changeTile32(y+1, x, 91, 0, 0, 0)
        return
```

```
def getLocation32():
	input = ""
	global cursorY, cursorX
	invert32(cursorY, cursorX)
	while input != "Return":
		if cursorX > 11 or cursorX < 2 or cursorY < 4 or cursorY > 13:
			print "Invalid cursor position: ", cursorX, ", ", cursorY
			print "Resetting cursor"
			cursorX = 2
			cursorY = 4
		if input == "Left" and cursorX > 2:
			invert32(cursorY, cursorX)
			cursorX = cursorX - 1
			invert32(cursorY, cursorX)
		elif input == "Right" and cursorX < 11:
			invert32(cursorY, cursorX)
			cursorX = cursorX + 1
			invert32(cursorY, cursorX)
		elif input == "Up" and cursorY > 4:
			invert32(cursorY, cursorX)
			cursorY = cursorY - 1
			invert32(cursorY, cursorX)
		elif input == "Down" and cursorY < 13:
			invert32(cursorY, cursorX)
			cursorY = cursorY + 1
			invert32(cursorY, cursorX)

		elif input == "A" or input == "a":
			invert32(cursorY, cursorX)
			cursorY = 4
			invert32(cursorY, cursorX)
		elif input == "1":
			invert32(cursorY, cursorX)
			cursorX = 2
			invert32(cursorY, cursorX)

		elif input == "B" or input == "b":
			invert32(cursorY, cursorX)
			cursorY = 5
			invert32(cursorY, cursorX)
		elif input == "2":
			invert32(cursorY, cursorX)
			cursorX = 3
			invert32(cursorY, cursorX)
```

```
elif input == "C" or input == "c":
        invert32(cursorY, cursorX)
        cursorY = 6
        invert32(cursorY, cursorX)
elif input == "3":
        invert32(cursorY, cursorX)
        cursorX = 4
        invert32(cursorY, cursorX)

elif input == "D" or input == "d":
        invert32(cursorY, cursorX)
        cursorY = 7
        invert32(cursorY, cursorX)
elif input == "4":
        invert32(cursorY, cursorX)
        cursorX = 5
        invert32(cursorY, cursorX)

elif input == "E" or input == "e":
        invert32(cursorY, cursorX)
        cursorY = 8
        invert32(cursorY, cursorX)
elif input == "5":
        invert32(cursorY, cursorX)
        cursorX = 6
        invert32(cursorY, cursorX)

elif input == "F" or input == "f":
        invert32(cursorY, cursorX)
        cursorY = 9
        invert32(cursorY, cursorX)
elif input == "6":
        invert32(cursorY, cursorX)
        cursorX = 7
        invert32(cursorY, cursorX)

elif input == "G" or input == "g":
        invert32(cursorY, cursorX)
        cursorY = 10
        invert32(cursorY, cursorX)
elif input == "7":
        invert32(cursorY, cursorX)
        cursorX = 8
        invert32(cursorY, cursorX)

elif input == "H" or input == "h":
```

```
                invert32(cursorY, cursorX)
                cursorY = 11
                invert32(cursorY, cursorX)
        elif input == "8":
                invert32(cursorY, cursorX)
                cursorX = 9
                invert32(cursorY, cursorX)

        elif input == "I" or input == "i":
                invert32(cursorY, cursorX)
                cursorY = 12
                invert32(cursorY, cursorX)
        elif input == "9":
                invert32(cursorY, cursorX)
                cursorX = 10
                invert32(cursorY, cursorX)

        elif input == "J" or input == "j":
                invert32(cursorY, cursorX)
                cursorY = 13
                invert32(cursorY, cursorX)
        elif input == "0":
                invert32(cursorY, cursorX)
                cursorX = 11
                invert32(cursorY, cursorX)

        input = getInput()
    return

def getLocation16():
    input = ""
    global cursorY, cursorX
    invert16(cursorY, cursorX)
    while input != "Return":
        if cursorX > 12 or cursorX < 3 or cursorY < 7 or cursorY > 16:
                print "Invalid cursor position: ", cursorX, ", ", cursorY
                print "Resetting cursor"
                cursorX = 2
                cursorY = 4
        if input == "Left" and cursorX > 3:
                invert16(cursorY, cursorX)
                cursorX = cursorX - 1
                invert16(cursorY, cursorX)
        elif input == "Right" and cursorX < 12:
                invert16(cursorY, cursorX)
                cursorX = cursorX + 1
```

```
            invert16(cursorY, cursorX)
    elif input == "Up" and cursorY > 7:
            invert16(cursorY, cursorX)
            cursorY = cursorY - 1
            invert16(cursorY, cursorX)
    elif input == "Down" and cursorY < 16:
            invert16(cursorY, cursorX)
            cursorY = cursorY + 1
            invert16(cursorY, cursorX)

    elif input == "A" or input == "a":
            invert16(cursorY, cursorX)
            cursorY = 7
            invert16(cursorY, cursorX)
    elif input == "1":
            invert16(cursorY, cursorX)
            cursorX = 3
            invert16(cursorY, cursorX)

    elif input == "B" or input == "b":
            invert16(cursorY, cursorX)
            cursorY = 8
            invert16(cursorY, cursorX)
    elif input == "2":
            invert16(cursorY, cursorX)
            cursorX = 4
            invert16(cursorY, cursorX)

    elif input == "C" or input == "c":
            invert16(cursorY, cursorX)
            cursorY = 9
            invert16(cursorY, cursorX)
    elif input == "3":
            invert16(cursorY, cursorX)
            cursorX = 5
            invert16(cursorY, cursorX)

    elif input == "D" or input == "d":
            invert16(cursorY, cursorX)
            cursorY = 10
            invert16(cursorY, cursorX)
    elif input == "4":
            invert16(cursorY, cursorX)
            cursorX = 6
            invert16(cursorY, cursorX)
```

```
elif input == "E" or input == "e":
        invert16(cursorY, cursorX)
        cursorY = 11
        invert16(cursorY, cursorX)
elif input == "5":
        invert16(cursorY, cursorX)
        cursorX = 7
        invert16(cursorY, cursorX)

elif input == "F" or input == "f":
        invert16(cursorY, cursorX)
        cursorY = 12
        invert16(cursorY, cursorX)
elif input == "6":
        invert16(cursorY, cursorX)
        cursorX = 8
        invert16(cursorY, cursorX)

elif input == "G" or input == "g":
        invert16(cursorY, cursorX)
        cursorY = 13
        invert16(cursorY, cursorX)
elif input == "7":
        invert16(cursorY, cursorX)
        cursorX = 9
        invert16(cursorY, cursorX)

elif input == "H" or input == "h":
        invert16(cursorY, cursorX)
        cursorY = 14
        invert16(cursorY, cursorX)
elif input == "8":
        invert16(cursorY, cursorX)
        cursorX = 10
        invert16(cursorY, cursorX)

elif input == "I" or input == "i":
        invert16(cursorY, cursorX)
        cursorY = 15
        invert16(cursorY, cursorX)
elif input == "9":
        invert16(cursorY, cursorX)
        cursorX = 11
        invert16(cursorY, cursorX)

elif input == "J" or input == "j":
```

```
                invert16(cursorY, cursorX)
                cursorY = 16
                invert16(cursorY, cursorX)
        elif input == "0":
                invert16(cursorY, cursorX)
                cursorX = 12
                invert16(cursorY, cursorX)

        input = getInput()
    return

#Conduct the players turn
def playerTurn():
    global cursorX, cursorY, playerScore
    fired = 0
    cursorX = 3
    cursorY = 7
    displayString16("Select shot", 3, 2)
    displayString16("location:  ", 4, 2)
    while fired == 0:
        getLocation16()
        if display16[cursorY, cursorX, 0] == 5:
            fired = 1
    x = cursorX - 2
    if x == 10:
        x = 0
    y = cursorY - 6
    if y == 10:
        y = 0
    print "Shot: ", x, " ", y
    message = "Shot: " + str(x) + " " + str(y)
    sendPacket(message)
    response = 0
    while response == 0:
        message = receivePacket()
        if len(message) >= 3 and message[0:3] == "Hit":
            playerScore = playerScore + 1
            displayString16("Hit       ", 3, 2)
            changeTile16(cursorY, cursorX, 6, 0, 0, 0)
            response = 1
        elif len(message) >= 4 and message[0:4] == "Miss":
            displayString16("Miss      ", 3, 2)
            changeTile16(cursorY, cursorX, 7, 0, 0, 0)
            response = 1
        elif len(message) >= 4 and message[0:4] == "Sunk":
            ship = message[6:7]
```

```
                        changeTile16(cursorY, cursorX, 6, 0, 0, 0)
                        playerScore = playerScore + 1
                        displayString16("Sunk       ", 3, 2)
                        if ship == "B":
                                displayString16("Battleship ", 4, 2)

                        elif ship == "A":
                                displayString16("Carrier    ", 4, 2)

                        elif ship == "D":
                                displayString16("Destroyer  ", 4, 2)

                        elif ship == "S":
                                displayString16("Submarine  ", 4, 2)

                        elif ship == "P":
                                displayString16("Patrol Boat", 4, 2)
                        response = 1
        print "I have sent a shot and received a response!"
        return

#Conduct the opponents turn
def opponentTurn():
        global opponentScore, carrierleft, battleshipleft, destroyerleft, submarineleft,
patrolboatleft
        receivedShot = 0
        x = -1
        y = -1
        while receivedShot == 0:
                message = receivePacket()
                if len(message) >= 9 and message[0:5] == "Shot":
                        receivedShot = 1
                        x = message[6:7]
                        y = message[8:9]
                        if x == 0:
                                x = 10
                        if y == 0:
                                y = 10

        print "I have received a shot"

        if display32[y + 3, x + 1, 0] == 5:
                sendPacket("Miss")
                changeTile32(y + 3, x + 1, 7, 0, 0, 0) #Display Miss
        elif display32[y + 3, x + 1, 0] >= 75 and display32[y + 3, x + 1, 0] <= 91:  #If this is an
unexploded ship
```

```
                opponentScore = opponentScore + 1
                if display32[y + 3, x + 1, 0] >= 75 and display32[y + 3, x + 1, 0] <= 79: #Carrier
                        carrierleft = carrierleft - 1
                        if carrierleft == 0:
                                sendPacket("Sunk A")
                                print "Carrier sunk!"
                        else:
                                sendPacket("Hit")
                elif display32[y + 3, x + 1, 0] >= 80 and display32[y + 3, x + 1, 0] <= 83:
#Battleship
                        battleshipleft = battleshipleft - 1
                        if battleshipleft == 0:
                                sendPacket("Sunk B")
                                print "Battleship sunk!"
                        else:
                                sendPacket("Hit")
                elif display32[y + 3, x + 1, 0] >= 84 and display32[y + 3, x + 1, 0] <= 86:
#Destroyer
                        destroyerleft = destroyerleft - 1
                        if destroyerleft == 0:
                                sendPacket("Sunk D")
                                print "Destroyer sunk!"
                        else:
                                sendPacket("Hit")
                elif display32[y + 3, x + 1, 0] >= 87 and display32[y + 3, x + 1, 0] <= 89:
#Submarine
                        submarineleft = submarineleft - 1
                        if submarineleft == 0:
                                sendPacket("Sunk S")
                                print "Submarine sunk!"
                        else:
                                sendPacket("Hit")
                elif display32[y + 3, x + 1, 0] >= 90 and display32[y + 3, x + 1, 0] <= 91: #Patrol
Boat
                        patrolboatleft = patrolboatleft - 1
                        if patrolboatleft == 0:
                                sendPacket("Sunk P")
                                print "Patrol Boat sunk!"
                        else:
                                sendPacket("Hit")

                sendPacket("Hit")
                changeTile32(y + 3, x + 1, display[y+3, x+1, 0] + 17, 0, 0, 0) #Display exploded
ship part
        else:
                print "Invalid shot attempted"
```

```
                sendPacket("Invalid shot")
        return

#If the player wins
def playerVictory():
        displayString16("Victory    ", 25, 2)
        return

#If the opponent wins
def opponentVictory():
        displayString32("Defeat     ", 25, 2)
        return

#initializeDisplayData()
#initializeDisplay()
main()
print("I have collapsed to the mainloop.")
root.mainloop()
```

GAME CODE (C for DE2 Board):

```
#include "basic_io.h"
#include "DM9000A.h"
#include "ethernet.h"

#include <stdio.h>
#include <system.h>
#include <io.h>
#include <string.h>
//#include "sprite16_data.h"
//#include "sprite32_data.h"
#include "vga_if.h"
#include "ethernet.h"
#include "DM9000A.h"
//unsigned char mac_address[6] = { 0x01, 0x60, 0x6E, 0x11, 0x02, 0x0F  };
//unsigned int interrupt_number;
//extern void ethernet_interrupt_handler();
//extern unsigned short transmit_checksum;

unsigned char code;
unsigned char keyflag = 0;
int display32[15][12][4];
int display16[30][16][4];

int playerScore = 0;
int opponentScore = 0 ;
int turn = 0;
int cursorX = 0;
int cursorY = 0;
int displaySetting = 0; //0 = player ships big, opponent ships small;  1 = opponent ships big,
player ships small
int exitGame = 0;  //0 = keep going, 1 = exit game
int exitPlayer = 1; //0 = Keep same player, 1 = New player
int exitRound = 0;  //0 = keep going, 1 = exit round
char* playerName = "Your Name  ";
char* opponentName = "Their Name ";
int patrolboatleft = 2;
int submarineleft = 3;
int destroyerleft = 3;
int battleshipleft = 4;
int carrierleft = 5;

int receivedMessageFlag = 0;

#define MAX_MSG_LENGTH 128
```

```
#define IP_LENGTH_OFFSET 16
#define UDP_PACKET_PAYLOAD_OFFSET 42
#define UDP_PACKET_LENGTH_OFFSET 38
#define UDP_PACKET_PAYLOAD (transmit_buffer + UDP_PACKET_PAYLOAD_OFFSET)
#define IP_HEADER_CHECKSUM_OFFSET 24
#define IP_PACKET_ID_OFFSET 18
#define IP_DESTINATION_ADDRESS_OFFSET 30
int jjj = 1;

// Ethernet MAC address.  Choose the last three bytes yourself
unsigned char mac_address[6] = { 0x01, 0x60, 0x6E, 0x11, 0x02, 0x0F  };
unsigned int interrupt_number;
// MOVED TO MAIN()

unsigned char ip_address[4] = { 0xc0, 0xa8, 0x01, 0x02 };

unsigned int receive_buffer_length = 0;
unsigned char receive_buffer[1600];
char receivedMessage[1600];
unsigned short transmit_checksum;
unsigned short receive_checksum;

char* pointer_to_transmit_buffer;
char* pointer_to_receive_buffer;

unsigned short packet_id = 0x0000;

// CHECKSUM FUNCTION ----------------------------------------
static unsigned short udp_everyPacket_computeChecksum(char* buffer){
   int access;
   unsigned int checksum = 0x00000000;

   for (access = 14; access < 34; access++) { // ip header from byte 14 to 33 inclusive
      if(access != IP_HEADER_CHECKSUM_OFFSET) {
         checksum = checksum + ((((int)buffer[access]) << 8) & 0x0000ff00); //msb
        // printf("\n+%x", ((((int)buffer[access]) << 8) & 0x0000ff00));
         access++;
         checksum = checksum + (((int)buffer[access]) & 0x000000ff); //lsb
         //printf("\n+%x", (((int)buffer[access]) & 0x000000ff));
         /* this adds to check sum the following:
         assume buffer[22] = 0x80 and buffer[23] = 0x11
         then (buffer[22] << 8) = 0x8000 */
      } else{
         access++; // since checksum is 2 bytes long, this skips checksum_offset + 1 also :)
      }
      // ignore the check sum fields (24-25) :)
```

```
    }

    //printf("\nCheck: %x", checksum);
    checksum = (checksum & 0x0FFFF) + ((checksum & 0xF0000) >> 16); // 2's complement
    //printf("\n2comp: %x", checksum);
    checksum = ~checksum; // 1's complement
   // printf("\n1comp: %x", checksum);
    //printf("\nshort: %x", (short)checksum);

    return (short)checksum;
}

// TRANSMIT BUFFER ---------------------------------------- // from Lab 2
unsigned char arp_buffer[] = {
  // Ethernet MAC header
  0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, // Destination MAC address (must be upadated)
  0x01, 0x60, 0x6E, 0x11, 0x02, 0x0F, // Source MAC address

  0x08, 0x06, // ARP packet type

  0x00, 0x01, // Hardware Type = ethernet 0x0001
  0x08, 0x00, // Protocol Type = IP (0x0800)
  0x06, 0x04, // 6 bytes for MAC address, 4 bytes for IP address
  0x00, 0x02, // 0x0002 indicates an ARP reply
  0x01, 0x60, 0x6E, 0x11, 0x02, 0x0F, // my MAC (again!)
  0xc0,0xa8,0x01,0x01, // our IP
  0xC8, 0x2A, 0x14, 0x37, 0xCA, 0x8A, // destination, must be updated too
  0xc0,0xa8,0x01,0x02 // STATIC
};

unsigned char transmit_buffer[] = {
  // Ethernet MAC header
  0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, // Destination MAC address
  0x01, 0x60, 0x6E, 0x11, 0x02, 0x0F, // Source MAC address ----------------------
  // MUST WRITE FUCN TO ALLOW USER TO MANUALLY INPUT MAC/IP
ADDRESSES ------------
  // ------------------------------------------------------------------------
  // ------------------------------------------------------------------------
  0x08, 0x00,                       // Packet Type: 0x800 = IP

  // IP Header
  0x45,              // version (IPv4), header length = 20 bytes
  0x00,              // differentiated services field
  0x00,0x9C,           // total length: 20 bytes for IP header +
                    // 8 bytes for UDP header + bytes of payload
  0x00, 0x00,          // packet ID
```

```
  0x00,              // flags
  0x00,              // fragment offset
  0x80,              // time-to-live
  0x11,              // protocol: 11 = UDP
  0x00,0x00,             // header checksum // initially 0x0000
  0xc0,0xa8,0x01,0x01, // source IP address
  0xc0,0xa8,0x01,0x02, // destination IP address

  // UDP Header
  0x13,0x8D, // source port port (5005)
  0x13,0x8D, // destination port (5005)
  0x00,0x88, // length (136: 8 for UDP header + data)
  0x00,0x00, // checksum: 0 = none

  // UDP payload
  0x74, 0x65, 0x73, 0x74, 0x20, 0x6d, 0x73, 0x67,
  0x74, 0x65, 0x73, 0x74, 0x20, 0x6d, 0x73, 0x67,
  0x74, 0x65, 0x73, 0x74, 0x20, 0x6d, 0x73, 0x67,
  0x74, 0x65, 0x73, 0x74, 0x20, 0x6d, 0x73, 0x67,
  0x74, 0x65, 0x73, 0x74, 0x20, 0x6d, 0x73, 0x67,
  0x74, 0x65, 0x73, 0x74, 0x20, 0x6d, 0x73, 0x67,
  0x74, 0x65, 0x73, 0x74, 0x20, 0x6d, 0x73, 0x67,
  0x74, 0x65, 0x73, 0x74, 0x20, 0x6d, 0x73, 0x67,
  0x74, 0x65, 0x73, 0x74, 0x20, 0x6d, 0x73, 0x67,
  0x74, 0x65, 0x73, 0x74, 0x20, 0x6d, 0x73, 0x67,
  0x74, 0x65, 0x73, 0x74, 0x20, 0x6d, 0x73, 0x67,
  0x74, 0x65, 0x73, 0x74, 0x20, 0x6d, 0x73, 0x67,
  0x74, 0x65, 0x73, 0x74, 0x20, 0x6d, 0x73, 0x67,
  0x74, 0x65, 0x73, 0x74, 0x20, 0x6d, 0x73, 0x67,
  0x74, 0x65, 0x73, 0x74, 0x20, 0x6d, 0x73, 0x67,
  0x74, 0x65, 0x73, 0x74, 0x20, 0x6d, 0x73, 0x67
};

// SETTING PKT LENGTH ----------------------------------------
void udp_everyPacket_setPacketLength(unsigned int packet_length){
  // sets the length field of Ethernet appropriately
  transmit_buffer[IP_LENGTH_OFFSET] = (20 + 8 + packet_length) >> 8;
  transmit_buffer[IP_LENGTH_OFFSET + 1] = (20 + 8 + packet_length) & 0xff;

  // sets the length field of UDP appropriately
  transmit_buffer[UDP_PACKET_LENGTH_OFFSET] = (8 + packet_length) >> 8;
  transmit_buffer[UDP_PACKET_LENGTH_OFFSET + 1] = (8 + packet_length) & 0xff;
  return;
}

// SETTING PACKET ID ----------------------------------------
```

```
void udp_everyPacket_setPacketID(){
   // include packet id
   packet_id = (packet_id + 1) & 0xffff; // maintian it 2 bytes long
   transmit_buffer[IP_PACKET_ID_OFFSET] = packet_id >> 8;
   transmit_buffer[IP_PACKET_ID_OFFSET + 1] = packet_id & 0x00ff;
   return;
}

void udp_everyPacket_setChecksum(char* buffer, unsigned int checksum){
    // include checksum
    buffer[IP_HEADER_CHECKSUM_OFFSET] = (checksum & 0x0000ffff) >> 8; // msb
    buffer[IP_HEADER_CHECKSUM_OFFSET + 1] = checksum & 0x000000ff; // lsb
    return;
}

// FROMING ATTACK LAUNCH PKT ----------------------------------------
/*unsigned int udp_formPacket_myAttack_coordinates(unsigned short X, unsigned short Y){
  // UDP_PACKET_PAYLOAD[0] indicates purpose of packet

  transmit_buffer[UDP_PACKET_PAYLOAD_OFFSET + 0] = 'A'; // A stands for send my
attaaaack!!!

  transmit_buffer[UDP_PACKET_PAYLOAD_OFFSET + 1] = 'X';
  if(X == 0) { // int 0 terminates UDP packet!
    transmit_buffer[UDP_PACKET_PAYLOAD_OFFSET + 2] = 'Z'; // Z for Zero, or Zoro! :)
  } else {
    transmit_buffer[UDP_PACKET_PAYLOAD_OFFSET + 2] = 'e';
  }
  transmit_buffer[UDP_PACKET_PAYLOAD_OFFSET + 3] = 'Y';
  if(Y == 0) {
    transmit_buffer[UDP_PACKET_PAYLOAD_OFFSET + 4] = 'Z';
  } else {
    transmit_buffer[UDP_PACKET_PAYLOAD_OFFSET + 4] = 'f';
  }
  transmit_buffer[UDP_PACKET_PAYLOAD_OFFSET + 5] = 0; // termination character

  unsigned short packet_length;
  packet_length = 6;

 return packet_length;
}*/

//////////////////////
//////char* receivePacket(){ ////// NAME IS ALREADY USED CHAAAANGEEEE!!!!
//////    //while (receive_buffer_length == 0){
//////        //Wait for a packet
```

```
//////    //}
//////    //receive_buffer_length = 0;
//////    return receive_buffer;
//////}

// RECEIVING ETHERNET UDP MSGS ----------------------------------------
//static void ethernet_interrupt_handler() {
static void ethernet_interrupt_handler() {
  unsigned int receive_status;

//if(jjj == 1){
//    TransmitPacket(arp_buffer, 42); /////////////// remove later
//    jjj = 0;
//}

//while(1){
  receive_status = ReceivePacket(receive_buffer, &receive_buffer_length);
// TransmitPacket(arp_buffer, 42);
  if (receive_status == DMFE_SUCCESS) {
    //printf("receive_buffer_length : %x\n", receive_buffer_length);

    if (receive_buffer_length >= 14) {
      //  A real Ethernet packet
      if ((receive_buffer[12] == 8 && receive_buffer[13] == 0 &&
      receive_buffer_length >= 34)) {
    // An IP packet
    if ((receive_buffer[23] == 0x11)){
      // A UDP packet
      if (receive_buffer_length >= UDP_PACKET_PAYLOAD_OFFSET) {
        receivedMessageFlag = 1;
        int i; //printf("\n\nReceived:\n");
        for(i = 0; i < receive_buffer_length; i++){
            //printf("%x\t", receive_buffer[i]);
            if ((i+1) % 8 == 0){
             //printf("\n");
            }
        }
        printf("\n\nMessage Received: ");
        //char* message_received = "";

        char* message_received = receive_buffer + UDP_PACKET_PAYLOAD_OFFSET;

        //sprintf(message_received, "%s", packet_message);
        //printf("%x\n", message_received[0]);
        printf("%s", message_received); printf("\n");
```

```
        //printf("\nReceived: %x\n",
         //  receive_buffer + UDP_PACKET_PAYLOAD_OFFSET);
         // get checksum for ip header
          pointer_to_receive_buffer = &receive_buffer[0];
          receive_checksum = udp_everyPacket_computeChecksum(pointer_to_receive_buffer);
           // verify checksum to determine weather to print to screen or discard
          if( ((receive_checksum >> 8) == (receive_buffer[IP_HEADER_CHECKSUM_OFFSET]))
            && ((receive_checksum & 0x00ff) ==
(receive_buffer[IP_HEADER_CHECKSUM_OFFSET+1])) )
            {
               //printf("\nChecksum OK");
            }
            else {
               //printf("\nChecksum test failed. Message ignored.\n");
            }

      }
    } else {
      //printf("\nReceived non-UDP packet\n");
    }
      } else {
    //printf("\nReceived non-IP packet\n");
        TransmitPacket(arp_buffer, 42);
        printf("\nARP reply sent.\n");
      }
    } else {
      //printf("Malformed Ethernet packet\n");
    }

  } else {
    //printf("Error receiving packet\n");
   // break;
  }
//}

  /* Clear the DM9000A ISR: PRS, PTS, ROS, ROOS 4 bits, by RW/C1 */
  dm9000a_iow(ISR, 0x3F);

  /* Re-enable DM9000A interrupts */
  dm9000a_iow(IMR, INTR_set);
}


char* receivePacket(){
    int i;
    while(receivedMessageFlag == 0){
```

```
    }
    for(i = UDP_PACKET_PAYLOAD_OFFSET; i < receive_buffer_length; i++)
       receivedMessage[i - UDP_PACKET_PAYLOAD_OFFSET] = receive_buffer[i];
    printf("receivePacket function returns: %s\n", receivedMessage);
    receivedMessageFlag = 0;
    return receivedMessage;
}

void sendPacket(char* message){
//This sends a UDP packet with the string message as the payload to the recipient at
IPADDRESS

  unsigned int message_length = 0;
  unsigned int packet_checksum;

   transmit_buffer[IP_DESTINATION_ADDRESS_OFFSET + 0] = ip_address[0];
   transmit_buffer[IP_DESTINATION_ADDRESS_OFFSET + 1] = ip_address[1];
   transmit_buffer[IP_DESTINATION_ADDRESS_OFFSET + 2] = ip_address[2];
   transmit_buffer[IP_DESTINATION_ADDRESS_OFFSET + 3] = ip_address[3];

   while(message[message_length] != '\0'){
    transmit_buffer[UDP_PACKET_PAYLOAD_OFFSET + message_length] =
message[message_length];
    message_length++;
   }
   transmit_buffer[UDP_PACKET_PAYLOAD_OFFSET + message_length] = '\0';
   message_length++;

      // Set Packet ID
    udp_everyPacket_setPacketID();

    // Set Packet Length
    udp_everyPacket_setPacketLength(message_length);

    // get checksum for ip header
    pointer_to_transmit_buffer = &transmit_buffer[0];
    packet_checksum = udp_everyPacket_computeChecksum(pointer_to_transmit_buffer);
    udp_everyPacket_setChecksum(pointer_to_transmit_buffer, packet_checksum);
    ////////////////////////
     int j; printf("\n"); for (j = 0; j < 14 + 20 + 8 + message_length; j++){ // 14 ethernet, 20 ip, 8
udp headers
     //printf("%x\t",transmit_buffer[j]);
     if ((j+1) % 6 == 0){
     //printf("\n");
     }
```

```
    } //////////////////////////

    if (TransmitPacket(transmit_buffer, message_length + 14 + 20 + 8)==DMFE_SUCCESS) {
        printf("\nMessage sent successfully");
    } else {
        printf("\nMessage sending failed\n");
    }
}

char* getInput()
{
    //printf("Getting Keystroke");
    int valid = 0;
    char* input_string;
    while (valid == 0){
        while(!IORD_8DIRECT(PS2_BASE, 0));
            //printf("polled status is %x \n" , IORD_8DIRECT(PS2_BASE, 0));
        //printf("Got Keystroke!");
        code = IORD_8DIRECT(PS2_BASE,4);
        //printf("scan code is %x\n", code);


       //if (is_my_turn == 1)  {
        switch(code)
        {
            case 0x1C: // A
              if (keyflag == 0){
                valid = 1;
                input_string ="A";
              }
              keyflag = 0;
              break;
            case 0x32: // B
              if (keyflag == 0){
                valid = 1;
                input_string ="B";
              }
              keyflag = 0;
              break;
            case 0x21: // C
              if (keyflag == 0){
                valid = 1;
                input_string ="C";
              }
              keyflag = 0;
              break;
```

```
case 0x23: // D
  if (keyflag == 0){
    valid = 1;
    input_string ="D";
  }
  keyflag = 0;
  break;
case 0x24: // E
  if (keyflag == 0){
    valid = 1;
    input_string ="E";
  }
  keyflag = 0;
  break;
case 0x2B: // F
  if (keyflag == 0){
    valid = 1;
    input_string ="F";
  }
  keyflag = 0;
  break;
case 0x34: // G
  if (keyflag == 0){
    valid = 1;
    input_string ="G";
  }
  keyflag = 0;
  break;
case 0x33: // H
  if (keyflag == 0){
    valid = 1;
    input_string ="H";
  keyflag = 0;
  }
  break;
case 0x43: // I
  if (keyflag == 0){
    valid = 1;
    input_string ="I";
  }
  keyflag = 0;
  break;
case 0x3B: // J
  if (keyflag == 0){
    valid = 1;
    input_string ="J";
```

```
  }
  keyflag = 0;
  break;
case 0x42: // K
  if (keyflag == 0){
    valid = 1;
    input_string ="K";
  }
  keyflag = 0;
  break;
case 0x4B: // L
  if (keyflag == 0){
    valid = 1;
    input_string ="L";
  }
  keyflag = 0;
  break;
case 0x3A: // M
  if (keyflag == 0){
    valid = 1;
    input_string ="M";
  }
  keyflag = 0;
  break;
 case 0x31: // N
  if (keyflag == 0){
    valid = 1;
    input_string ="N";
  }
  keyflag = 0;
  break;
 case 0x44: // O
  if (keyflag == 0){
    valid = 1;
    input_string ="O";
  }
  keyflag = 0;
  break;
 case 0x4D: // P
  if (keyflag == 0){
    valid = 1;
    input_string ="P";
  }
  keyflag = 0;
  break;
  case 0x15: // Q
```

```
 if (keyflag == 0){
   valid = 1;
   input_string ="Q";
 }
 keyflag = 0;
 break;
case 0x2D: // R
 if (keyflag == 0){
   valid = 1;
   input_string ="R";
 }
 keyflag = 0;
 break;
 case 0x1B: // S
 if (keyflag == 0){
   valid = 1;
   input_string ="S";
 }
 keyflag = 0;
 break;
 case 0x2C: // T
 if (keyflag == 0){
   valid = 1;
   input_string ="T";
 }
 keyflag = 0;
 break;
 case 0x3C: // U
 if (keyflag == 0){
   valid = 1;
   input_string ="U";
 }
 keyflag = 0;
 break;
case 0x2A: // V
 if (keyflag == 0){
   valid = 1;
   input_string ="V";
 }
 keyflag = 0;
 break;
case 0x1D: // W
 if (keyflag == 0){
   valid = 1;
   input_string ="W";
 }
```

```
  keyflag = 0;
  break;
 case 0x22: // X
  if (keyflag == 0){
    valid = 1;
    input_string ="X";
  }
  keyflag = 0;
  break;
  case 0x35: // Y
  if (keyflag == 0){
    valid = 1;
    input_string ="Y";
  }
  keyflag = 0;
  break;
  case 0x1A: // Z
  if (keyflag == 0){
    valid = 1;
    input_string ="Z";
  }
  keyflag = 0;
  break;

case 0x45: // 0
  if (keyflag == 0){
    valid = 1;
    input_string ="0";
  }
  keyflag = 0;
  break;
case 0x16: // 1
  if (keyflag == 0){
    valid = 1;
    input_string ="1";
  }
  keyflag = 0;
  break;
case 0x1E: // 2
  if (keyflag == 0){
    valid = 1;
    input_string ="2";
  }
  keyflag = 0;
  break;
case 0x26: // 3
```

```
  if (keyflag == 0){
    valid = 1;
    input_string ="3";
  }
  keyflag = 0;
  break;
case 0x25: // 4
  if (keyflag == 0){
    valid = 1;
    input_string ="4";
  }
  keyflag = 0;
  break;
case 0x2E: // 5
  if (keyflag == 0){
    valid = 1;
    input_string ="5";
  }
  keyflag = 0;
  break;
case 0x36: // 6
  if (keyflag == 0){
    valid = 1;
    input_string ="6";
  }
  keyflag = 0;
  break;
case 0x3D: // 7
  if (keyflag == 0){
    valid = 1;
    input_string ="7";
  }
  keyflag = 0;
  break;
case 0x3E: // 8
  if (keyflag == 0){
    valid = 1;
    input_string ="8";
  }
  keyflag = 0;
  break;
case 0x46: // 9
  if (keyflag == 0){
    valid = 1;
    input_string ="9";
  }
```

```
  keyflag = 0;
  break;
  case 0x29: // Space
  if (keyflag == 0){
    valid = 1;
    input_string =" ";
  }
  keyflag = 0;
  break;
   case 0x66: // Back Space
  if (keyflag == 0){
    valid = 1;
    input_string ="BackSpace";
  }
  keyflag = 0;
  break;
case 0x6B: // LEFT Key
if (keyflag == 1){
    valid = 1;
    input_string ="Left";
  } // ELSE DO NOTHING
 keyflag = 0;
  break;
case 0x74: // RIGHT Key
if (keyflag == 1){
    valid = 1;
    input_string ="Right";
  } // ELSE DO NOTHING

keyflag = 0;
  break;
case 0x75: // UP Key
if (keyflag == 1){
    valid = 1;
    input_string ="Up";
}
keyflag = 0;
  break;
case 0x72: //DOWN
if (keyflag == 1){
    valid = 1;
    input_string ="Down";
}
keyflag = 0;
  break;
case 0x5A:
```

```
            if (keyflag == 0)
            {
               valid = 1;
               input_string ="Return";

            }
            keyflag = 0;
            break;
            case 0x76: //escape
               if (keyflag == 0){
                  valid = 1;
                  input_string ="Escape";
               }
               keyflag = 0;
            break;

            case 0xE0 :
              keyflag = 1;
              input_string = "";
            break;
            case 0xF0 :
              keyflag = 2;
              input_string = "";
            break;
           default:
               keyflag = 0;
              // ILLEGAL GAME KEY
              // DO NOT CHANGE TARGET
              break;
        }
     }
    //}
     return input_string;
}


struct sprite_addr_type
{
   unsigned short data     : 7;
   unsigned short set_rt   : 1;
   unsigned short set_fl   : 1;
   unsigned short set_hl   : 1;
   unsigned short pd       : 6;
};

#define sprite_addr \
```

```
   struct sprite_addr_type;
/*
#define WRITE_RAM(offset, data) \
   IOWR_16DIRECT(VGA_BASE, offset << 1, data)
#define READ_RAM(offset) \
   IORD_16DIRECT(VGA_BASE, offset << 1 )
*/



int load_sprites_32()
{
   int i;
   for(i = 0; i < SPRITE_32_LEN; i++)
   {
      IOWR_8DIRECT(SRAM_BASE, i, sprite_32_data[i]);
   }
   return 0;
}

int load_sprites_16()
{
   int i;
   for(i = 0; i < SPRITE_16_LEN; i++)
   {
      IOWR_8DIRECT(SRAM_BASE, i + 0x20000, sprite_16_data[i]);
   }
   return 0;
}

unsigned short changeTile32(int y, int x, int spriteno, int rot, int flip, int inv)
{
   display32[y][x][0] = spriteno;
   display32[y][x][1] = rot;
   display32[y][x][2] = flip;
   display32[y][x][3] = inv;
   struct sprite_data data;
   unsigned short wdata;
   unsigned int addr;

   addr = ((y % 15) * 12) + (x % 12);
   data.data = spriteno;
   data.set_rt = rot & 1;
   data.set_fl = flip & 1;
   data.set_hl = inv & 1;
   data.pd = 0;
```

```
    wdata = *((unsigned short *)(&data));
    WRITE_RAM(addr, wdata);
    return wdata;
}

unsigned short changeTile16(int y, int x, int spriteno, int rot, int flip, int inv)
{
    display16[y][x][0] = spriteno;
    display16[y][x][1] = rot;
    display16[y][x][2] = flip;
    display16[y][x][3] = inv;
    struct sprite_data data;
    unsigned short wdata;
    unsigned int addr;

    addr = ((y % 30) * 16) + (x % 16) + 180;
    data.data = spriteno;
    data.set_rt = rot & 1;
    data.set_fl = flip & 1;
    data.set_hl = inv & 1;
    data.pd = 0;

    wdata = *((unsigned short *)(&data));
    WRITE_RAM(addr, wdata);
    return wdata;
}

unsigned short read_sprite_32(int x, int y)
{
    unsigned int addr = ((y % 15) * 12) + (x % 12);
    return READ_RAM(addr);
}

unsigned short read_sprite_16(int x, int y)
{
    unsigned int addr = ((y % 30) * 16) + (x % 16) + 180;
    return READ_RAM(addr);
}

/*unsigned short invert32(int x, int y)
{
    struct sprite_data data;
    int hl_bit;
    unsigned short wdata;
    unsigned int addr = ((y % 15) * 12) + (x % 12);
```

```
    wdata = READ_RAM(addr);
    data = *((struct sprite_data *) &wdata);
    hl_bit = data.set_hl ^ 1;
    data.set_hl = hl_bit & 1;

    wdata = *((unsigned short *)(&data));
    WRITE_RAM(addr, wdata);
    return wdata;
}

unsigned short invert16(int x, int y)
{
    struct sprite_data data;
    int hl_bit;
    unsigned short wdata;
    unsigned int addr = ((y % 30) * 16) + (x % 16) + 180;
    wdata = READ_RAM(addr);
    data = *((struct sprite_data *) &wdata);
    hl_bit = data.set_hl ^ 1;
    data.set_hl = hl_bit & 1;

    wdata = *((unsigned short *)(&data));
    WRITE_RAM(addr, wdata);
    return wdata;
}
*/

int write_sprite_data_to_memory()
{
    printf("Running write_sprite_data_to_memory()\n");
    int i;
    for(i = 0; i < SPRITE_32_LEN; i++)
    {
        IOWR_8DIRECT(SRAM_BASE, i, sprite_32_data[i]);
    }
    for(i = 0; i < SPRITE_16_LEN; i++)
    {
        IOWR_8DIRECT(SRAM_BASE, i + 0x20000, sprite_16_data[i]);
    }
    return 0;
}

int run_sprites()
{
    printf("Running run_sprites()\n");
    unsigned int addr, data, rt, fl, hl;
```

```
    unsigned int x, y;
    struct sprite_addr_type in_data;
    unsigned short wdata;

    for(;;)
    {
     printf("Waiting for input:");


     scanf("%d %d %d %d %d %d", &x, &y, &data, &fl, &rt, &hl);
     if(x >= 100)
       addr = y * 16 + (x % 100) + 180;
     else
       addr = y * 12 + x;
     printf("Caught: (%02d, %02d)(0x%08x)=%d\n",x, y, addr, data);
     in_data.pd = 0;
     in_data.set_hl = hl & 1;
     in_data.set_fl = fl & 1;
     in_data.set_rt = rt & 1;
     in_data.data = data & 0x1FF;
     printf("%x %d %d %d\n", in_data.data, in_data.set_rt, in_data.set_fl, in_data.set_hl);

     wdata = *((unsigned short *)(&in_data));
     printf("Writing %x to addr %x\n", wdata, addr);
     WRITE_RAM(addr, wdata);
     printf("You wrote %x to addr %x\n", READ_RAM(addr), addr);
    }
    return 0;
}

int run_sprites_2()
{
    printf("Running run_sprites_2()\n");
    unsigned int data, rt, fl, hl;
    unsigned int x, y, tmpx;
    unsigned short wdata;

    for(;;)
    {
     printf("Waiting for input:");

     scanf("%d %d %d %d %d %d", &x, &y, &data, &fl, &rt, &hl);
     if(x >= 100)
     {
       tmpx = x % 100;
       changeTile16(tmpx, y, data, rt, fl, hl);
```

```
      wdata = read_sprite_32(tmpx, y);
      printf("You wrote %x to 32(%d, %d)\n", wdata, tmpx, y);
    }
    else
    {
      changeTile32(x, y, data, rt, fl, hl);
      wdata = read_sprite_32(x, y);
      printf("You wrote %x to 32(%d, %d)\n", wdata, x, y);
    }


  }
  return 0;
}

int run_sram()
{
  printf("Running run_sram()\n");
  unsigned int wr;
  unsigned int addr = 0;
  unsigned int data = 0;
  unsigned int temp = 18;
  printf("addr = %d, data = %d\n", addr, data);
  for(;;)
  {
   printf("ENTER COMMAND: \n");
   scanf("%d %d %d", &wr, &addr, &data);
   if(wr == 0)
   {
     printf("READING FROM %d...\n", addr);
     data = IORD_8DIRECT(SRAM_BASE, addr);
     printf("READ %d FROM ADDRESS %d\n", data, addr);
   }
   else if (wr == 1)
   {
     printf("WRITING %d TO %d...\n", data, addr);
     IOWR_8DIRECT(SRAM_BASE, addr, data);
     temp = IORD_8DIRECT(SRAM_BASE, addr);
     printf("WROTE %d TO ADDRESS %d\n", temp, addr);
   }
   else
   {
    printf("NOT A VALID COMMAND\n");
   }

  }
```

```
    return 0;
}

int run_keys()
{
    printf("Running run_keys()\n");
    int x = 0;
    int y = 0;
    int addr = 0;
    char * input;
    struct sprite_addr_type black_data, white_data;
    black_data.data = 0;
    black_data.set_rt = 0;
    black_data.set_fl = 0;
    black_data.set_hl = 0;
    white_data.data = 4;
    white_data.set_rt = 0;
    white_data.set_fl = 0;
    white_data.set_hl = 0;
    for(;;)
    {
        WRITE_RAM(addr, *((unsigned short *)(&white_data)));
        input = getInput();
        WRITE_RAM(addr, *((unsigned short *)(&black_data)));
        if(strcmp(input, "UP") == 0)
        {
          if (y == 0)
            y = 14;
          else
            y = y - 1;
        }
        else if (strcmp(input, "DOWN") == 0)
        {
           if (y == 14)
            y = 0;
          else
            y = y + 1;
        }
        else if (strcmp(input, "LEFT") == 0)
        {
           if (x == 0)
            x = 11;
          else
            x = x - 1;
        }
        else if (strcmp(input, "RIGHT") == 0)
```

```
        {
           if (x == 11)
             x = 0;
           else
             x = x + 1;
        }
        input = "";
        addr = y * 12 + x;
    }

    return 0;
}


//Invert a square on the 32 side
void invert32(int row, int col){
    if (display32[row][col][3] == 0)
        display32[row][col][3] = 1;
    else if (display32[row][col][3] == 1)
        display32[row][col][3] = 0;
    changeTile32(row, col, display32[row][col][0], display32[row][col][1],
display32[row][col][2], display32[row][col][3]);
    return;
}

//Invert a square on the 16 side
void invert16(int row, int col){
    if (display16[row][col][3] == 0)
        display16[row][col][3] = 1;
    else if (display16[row][col][3] == 1)
        display16[row][col][3] = 0;
    changeTile16(row, col, display16[row][col][0], display16[row][col][1],
display16[row][col][2], display16[row][col][3]);
    return;
}

//Display a character at a coordinate on the 32 side
char * displayChar32(char p_char, int row, int col){
    char * czChar = "\0\0";
    if (p_char == 'A'){
        czChar[0] = p_char;
        changeTile32(row, col, 12, 0, 0, 0);
    }
    else if (p_char == 'B'){
        czChar[0] = p_char;
        changeTile32(row, col, 13, 0, 0, 0);
```

```
}
else if (p_char == 'C'){
   czChar[0] = p_char;
   changeTile32(row, col, 14, 0, 0, 0);
}
else if (p_char == 'D'){
   czChar[0] = p_char;
   changeTile32(row, col, 15, 0, 0, 0);
}
else if (p_char == 'E'){
   czChar[0] = p_char;
   changeTile32(row, col, 16, 0, 0, 0);
}
else if (p_char == 'F'){
   czChar[0] = p_char;
   changeTile32(row, col, 17, 0, 0, 0);
}
else if (p_char == 'G'){
   czChar[0] = p_char;
   changeTile32(row, col, 18, 0, 0, 0);
}
else if (p_char == 'H'){
   czChar[0] = p_char;
   changeTile32(row, col, 19, 0, 0, 0);
}
else if (p_char == 'I'){
   czChar[0] = p_char;
   changeTile32(row, col, 20, 0, 0, 0);
}
else if (p_char == 'J'){
   czChar[0] = p_char;
   changeTile32(row, col, 21, 0, 0, 0);
}
else if (p_char == 'K'){
   czChar[0] = p_char;
   changeTile32(row, col, 22, 0, 0, 0);
}
else if (p_char == 'L'){
   czChar[0] = p_char;
   changeTile32(row, col, 23, 0, 0, 0);
}
else if (p_char == 'M'){
   czChar[0] = p_char;
   changeTile32(row, col, 24, 0, 0, 0);
}
else if (p_char == 'N'){
```

```
    czChar[0] = p_char;
    changeTile32(row, col, 25, 0, 0, 0);
}
else if (p_char == 'O'){
    czChar[0] = p_char;
    changeTile32(row, col, 26, 0, 0, 0);
}
else if (p_char == 'P'){
    czChar[0] = p_char;
    changeTile32(row, col, 27, 0, 0, 0);
}
else if (p_char == 'Q'){
    czChar[0] = p_char;
    changeTile32(row, col, 28, 0, 0, 0);
}
else if (p_char == 'R'){
    czChar[0] = p_char;
    changeTile32(row, col, 29, 0, 0, 0);
}
else if (p_char == 'S'){
    czChar[0] = p_char;
    changeTile32(row, col, 30, 0, 0, 0);
}
else if (p_char == 'T'){
    czChar[0] = p_char;
    changeTile32(row, col, 31, 0, 0, 0);
}
else if (p_char == 'U'){
    czChar[0] = p_char;
    changeTile32(row, col, 32, 0, 0, 0);
}
else if (p_char == 'V'){
    czChar[0] = p_char;
    changeTile32(row, col, 33, 0, 0, 0);
}
else if (p_char == 'W'){
    czChar[0] = p_char;
    changeTile32(row, col, 34, 0, 0, 0);
}
else if (p_char == 'X'){
    czChar[0] = p_char;
    changeTile32(row, col, 35, 0, 0, 0);
}
else if (p_char == 'Y'){
    czChar[0] = p_char;
    changeTile32(row, col, 36, 0, 0, 0);
```

```
}
else if (p_char == 'Z'){
    czChar[0] = p_char;
    changeTile32(row, col, 37, 0, 0, 0);
}
else if (p_char == 'a'){
    czChar[0] = p_char;
    changeTile32(row, col, 38, 0, 0, 0);
}
else if (p_char == 'b'){
    czChar[0] = p_char;
    changeTile32(row, col, 39, 0, 0, 0);
}
else if (p_char == 'c'){
    czChar[0] = p_char;
    changeTile32(row, col, 40, 0, 0, 0);
}
else if (p_char == 'd'){
    czChar[0] = p_char;
    changeTile32(row, col, 41, 0, 0, 0);
}
else if (p_char == 'e'){
    czChar[0] = p_char;
    changeTile32(row, col, 42, 0, 0, 0);
}
else if (p_char == 'f'){
    czChar[0] = p_char;
    changeTile32(row, col, 43, 0, 0, 0);
}
else if (p_char == 'g'){
    czChar[0] = p_char;
    changeTile32(row, col, 44, 0, 0, 0);
}
else if (p_char == 'h'){
    czChar[0] = p_char;
    changeTile32(row, col, 45, 0, 0, 0);
}
else if (p_char == 'i'){
    czChar[0] = p_char;
    changeTile32(row, col, 46, 0, 0, 0);
}
else if (p_char == 'j'){
    czChar[0] = p_char;
    changeTile32(row, col, 47, 0, 0, 0);
}
else if (p_char == 'k'){
```

```
    czChar[0] = p_char;
    changeTile32(row, col, 48, 0, 0, 0);
}
else if (p_char == 'l'){
    czChar[0] = p_char;
    changeTile32(row, col, 49, 0, 0, 0);
}
else if (p_char == 'm'){
    czChar[0] = p_char;
    changeTile32(row, col, 50, 0, 0, 0);
}
else if (p_char == 'n'){
    czChar[0] = p_char;
    changeTile32(row, col, 51, 0, 0, 0);
}
else if (p_char == 'o'){
    czChar[0] = p_char;
    changeTile32(row, col, 52, 0, 0, 0);
}
else if (p_char == 'p'){
    czChar[0] = p_char;
    changeTile32(row, col, 53, 0, 0, 0);
}
else if (p_char == 'q'){
    czChar[0] = p_char;
    changeTile32(row, col, 54, 0, 0, 0);
}
else if (p_char == 'r'){
    czChar[0] = p_char;
    changeTile32(row, col, 55, 0, 0, 0);
}
else if (p_char == 's'){
    czChar[0] = p_char;
    changeTile32(row, col, 56, 0, 0, 0);
}
else if (p_char == 't'){
    czChar[0] = p_char;
    changeTile32(row, col, 57, 0, 0, 0);
}
else if (p_char == 'u'){
    czChar[0] = p_char;
    changeTile32(row, col, 58, 0, 0, 0);
}
else if (p_char == 'v'){
    czChar[0] = p_char;
    changeTile32(row, col, 59, 0, 0, 0);
```

```
}
else if (p_char == 'w'){
   czChar[0] = p_char;
   changeTile32(row, col, 60, 0, 0, 0);
}
else if (p_char == 'x'){
   czChar[0] = p_char;
   changeTile32(row, col, 61, 0, 0, 0);
}
else if (p_char == 'y'){
   czChar[0] = p_char;
   changeTile32(row, col, 62, 0, 0, 0);
}
else if (p_char == 'z'){
   czChar[0] = p_char;
   changeTile32(row, col, 63, 0, 0, 0);
}
else if (p_char == '0'){
   czChar[0] = p_char;
   changeTile32(row, col, 64, 0, 0, 0);
}
else if (p_char == '1'){
   czChar[0] = p_char;
   changeTile32(row, col, 65, 0, 0, 0);
}
else if (p_char == '2'){
   czChar[0] = p_char;
   changeTile32(row, col, 66, 0, 0, 0);
}
else if (p_char == '3'){
   czChar[0] = p_char;
   changeTile32(row, col, 67, 0, 0, 0);
}
else if (p_char == '4'){
   czChar[0] = p_char;
   changeTile32(row, col, 68, 0, 0, 0);
}
else if (p_char == '5'){
   czChar[0] = p_char;
   changeTile32(row, col, 69, 0, 0, 0);
}
else if (p_char == '6'){
   czChar[0] = p_char;
   changeTile32(row, col, 70, 0, 0, 0);
}
else if (p_char == '7'){
```

```
        czChar[0] = p_char;
        changeTile32(row, col, 71, 0, 0, 0);
    }
    else if (p_char == '8'){
        czChar[0] = p_char;
        changeTile32(row, col, 72, 0, 0, 0);
    }
    else if (p_char == '9'){
        czChar[0] = p_char;
        changeTile32(row, col, 73, 0, 0, 0);
    }
    else if (p_char == ':'){
        czChar[0] = p_char;
        changeTile32(row, col, 116, 0, 0, 0);
    }
    else if (p_char == '?'){
        czChar[0] = p_char;
        changeTile32(row, col, 117, 0, 0, 0);
    }
    else if (p_char == ' '){
        czChar[0] = p_char;
        changeTile32(row, col, 11, 0, 0, 0);
    }
    else{
        changeTile32(row, col, 11, 0, 0, 0);
    }
    return czChar;
}


//Display a character at a coordinate on the 16 side
char * displayChar16(char p_char, int row, int col){
    char * czChar = "\0\0";
    if (p_char == 'A'){
        czChar[0] = p_char;
        changeTile16(row, col, 3+12, 0, 0, 0);
    }
    else if (p_char == 'B'){
        czChar[0] = p_char;
        changeTile16(row, col, 3+13, 0, 0, 0);
    }
    else if (p_char == 'C'){
        czChar[0] = p_char;
        changeTile16(row, col, 3+14, 0, 0, 0);
    }
    else if (p_char == 'D'){
```

```
    czChar[0] = p_char;
    changeTile16(row, col, 3+15, 0, 0, 0);
}
else if (p_char == 'E'){
    czChar[0] = p_char;
    changeTile16(row, col, 3+16, 0, 0, 0);
}
else if (p_char == 'F'){
    czChar[0] = p_char;
    changeTile16(row, col, 3+17, 0, 0, 0);
}
else if (p_char == 'G'){
    czChar[0] = p_char;
    changeTile16(row, col, 3+18, 0, 0, 0);
}
else if (p_char == 'H'){
    czChar[0] = p_char;
    changeTile16(row, col, 3+19, 0, 0, 0);
}
else if (p_char == 'I'){
    czChar[0] = p_char;
    changeTile16(row, col, 3+20, 0, 0, 0);
}
else if (p_char == 'J'){
    czChar[0] = p_char;
    changeTile16(row, col, 3+21, 0, 0, 0);
}
else if (p_char == 'K'){
    czChar[0] = p_char;
    changeTile16(row, col, 3+22, 0, 0, 0);
}
else if (p_char == 'L'){
    czChar[0] = p_char;
    changeTile16(row, col, 3+23, 0, 0, 0);
}
else if (p_char == 'M'){
    czChar[0] = p_char;
    changeTile16(row, col, 3+24, 0, 0, 0);
}
else if (p_char == 'N'){
    czChar[0] = p_char;
    changeTile16(row, col, 3+25, 0, 0, 0);
}
else if (p_char == 'O'){
    czChar[0] = p_char;
    changeTile16(row, col, 3+26, 0, 0, 0);
```

```
}
else if (p_char == 'P'){
   czChar[0] = p_char;
   changeTile16(row, col, 3+27, 0, 0, 0);
}
else if (p_char == 'Q'){
   czChar[0] = p_char;
   changeTile16(row, col, 3+28, 0, 0, 0);
}
else if (p_char == 'R'){
   czChar[0] = p_char;
   changeTile16(row, col, 3+29, 0, 0, 0);
}
else if (p_char == 'S'){
   czChar[0] = p_char;
   changeTile16(row, col, 3+30, 0, 0, 0);
}
else if (p_char == 'T'){
   czChar[0] = p_char;
   changeTile16(row, col, 3+31, 0, 0, 0);
}
else if (p_char == 'U'){
   czChar[0] = p_char;
   changeTile16(row, col, 3+32, 0, 0, 0);
}
else if (p_char == 'V'){
   czChar[0] = p_char;
   changeTile16(row, col, 3+33, 0, 0, 0);
}
else if (p_char == 'W'){
   czChar[0] = p_char;
   changeTile16(row, col, 3+34, 0, 0, 0);
}
else if (p_char == 'X'){
   czChar[0] = p_char;
   changeTile16(row, col, 3+35, 0, 0, 0);
}
else if (p_char == 'Y'){
   czChar[0] = p_char;
   changeTile16(row, col, 3+36, 0, 0, 0);
}
else if (p_char == 'Z'){
   czChar[0] = p_char;
   changeTile16(row, col, 3+37, 0, 0, 0);
}
else if (p_char == 'a'){
```

```
    czChar[0] = p_char;
    changeTile16(row, col, 3+38, 0, 0, 0);
}
else if (p_char == 'b'){
    czChar[0] = p_char;
    changeTile16(row, col, 3+39, 0, 0, 0);
}
else if (p_char == 'c'){
    czChar[0] = p_char;
    changeTile16(row, col, 3+40, 0, 0, 0);
}
else if (p_char == 'd'){
    czChar[0] = p_char;
    changeTile16(row, col, 3+41, 0, 0, 0);
}
else if (p_char == 'e'){
    czChar[0] = p_char;
    changeTile16(row, col, 3+42, 0, 0, 0);
}
else if (p_char == 'f'){
    czChar[0] = p_char;
    changeTile16(row, col, 3+43, 0, 0, 0);
}
else if (p_char == 'g'){
    czChar[0] = p_char;
    changeTile16(row, col, 3+44, 0, 0, 0);
}
else if (p_char == 'h'){
    czChar[0] = p_char;
    changeTile16(row, col, 3+45, 0, 0, 0);
}
else if (p_char == 'i'){
    czChar[0] = p_char;
    changeTile16(row, col, 3+46, 0, 0, 0);
}
else if (p_char == 'j'){
    czChar[0] = p_char;
    changeTile16(row, col, 3+47, 0, 0, 0);
}
else if (p_char == 'k'){
    czChar[0] = p_char;
    changeTile16(row, col, 3+48, 0, 0, 0);
}
else if (p_char == 'l'){
    czChar[0] = p_char;
    changeTile16(row, col, 3+49, 0, 0, 0);
```

```
}
else if (p_char == 'm'){
   czChar[0] = p_char;
   changeTile16(row, col, 3+50, 0, 0, 0);
}
else if (p_char == 'n'){
   czChar[0] = p_char;
   changeTile16(row, col, 3+51, 0, 0, 0);
}
else if (p_char == 'o'){
   czChar[0] = p_char;
   changeTile16(row, col, 3+52, 0, 0, 0);
}
else if (p_char == 'p'){
   czChar[0] = p_char;
   changeTile16(row, col, 3+53, 0, 0, 0);
}
else if (p_char == 'q'){
   czChar[0] = p_char;
   changeTile16(row, col, 3+54, 0, 0, 0);
}
else if (p_char == 'r'){
   czChar[0] = p_char;
   changeTile16(row, col, 3+55, 0, 0, 0);
}
else if (p_char == 's'){
   czChar[0] = p_char;
   changeTile16(row, col, 3+56, 0, 0, 0);
}
else if (p_char == 't'){
   czChar[0] = p_char;
   changeTile16(row, col, 3+57, 0, 0, 0);
}
else if (p_char == 'u'){
   czChar[0] = p_char;
   changeTile16(row, col, 3+58, 0, 0, 0);
}
else if (p_char == 'v'){
   czChar[0] = p_char;
   changeTile16(row, col, 3+59, 0, 0, 0);
}
else if (p_char == 'w'){
   czChar[0] = p_char;
   changeTile16(row, col, 3+60, 0, 0, 0);
}
else if (p_char == 'x'){
```

```
    czChar[0] = p_char;
    changeTile16(row, col, 3+61, 0, 0, 0);
}
else if (p_char == 'y'){
    czChar[0] = p_char;
    changeTile16(row, col, 3+62, 0, 0, 0);
}
else if (p_char == 'z'){
    czChar[0] = p_char;
    changeTile16(row, col, 3+63, 0, 0, 0);
}
else if (p_char == '0'){
    czChar[0] = p_char;
    changeTile16(row, col, 3+64, 0, 0, 0);
}
else if (p_char == '1'){
    czChar[0] = p_char;
    changeTile16(row, col, 3+65, 0, 0, 0);
}
else if (p_char == '2'){
    czChar[0] = p_char;
    changeTile16(row, col, 3+66, 0, 0, 0);
}
else if (p_char == '3'){
    czChar[0] = p_char;
    changeTile16(row, col, 3+67, 0, 0, 0);
}
else if (p_char == '4'){
    czChar[0] = p_char;
    changeTile16(row, col, 3+68, 0, 0, 0);
}
else if (p_char == '5'){
    czChar[0] = p_char;
    changeTile16(row, col, 3+69, 0, 0, 0);
}
else if (p_char == '6'){
    czChar[0] = p_char;
    changeTile16(row, col, 3+70, 0, 0, 0);
}
else if (p_char == '7'){
    czChar[0] = p_char;
    changeTile16(row, col, 3+71, 0, 0, 0);
}
else if (p_char == '8'){
    czChar[0] = p_char;
    changeTile16(row, col, 3+72, 0, 0, 0);
```

```
    }
    else if (p_char == '9'){
        czChar[0] = p_char;
        changeTile16(row, col, 3+73, 0, 0, 0);
    }
    else if (p_char == ':'){
        czChar[0] = p_char;
        changeTile16(row, col, 124, 0, 0, 0);
    }
    else if (p_char == '?'){
        czChar[0] = p_char;
        changeTile16(row, col, 125, 0, 0, 0);
    }
    else if (p_char == ' '){
        czChar[0] = p_char;
        changeTile16(row, col, 14, 0, 0, 0);
    }
    else{
        changeTile16(row, col, 14, 0, 0, 0);
    }
    return czChar;
}

//Display a string on the 32 side
void displayString32(char* inputstring, int row, int col){
    int l, i;
    char* output;
    l = strlen(inputstring);
    for (i=0;i<l;i++)
        output = displayChar32(inputstring[i], row, col + i);
    return;
}
//Display a string on the 16 side
void displayString16(char* inputstring, int row, int col){
    int l, i;
    char* output;
    l = strlen(inputstring);
    for (i=0; i < l; i++)
        output = displayChar16(inputstring[i], row, col + i);
    return;
}


//Set up the basic screen display, with a blank ocean and player name for both screens.
void initializeDisplay(){
    //Display the 32 side
```

```
//Display border on the 32 side
int c, r, i;
for (c=0;c<5;c++)
   changeTile32(0,c,8,0,0,0);
for (r=0;r<15;r++)
   changeTile32(r,0,8,0,0,0);
for (c=0;c<12;c++)
   changeTile32(14,c,8,0,0,0);
//Display the Battleship Title on the 32 side
for (i=0;i<7;i++)
   changeTile32(0,5+i,109+i,0,0,0);

//Display the battleship title on the 16 side
for (c=0;c<3;c++){
   changeTile16(0,0 + (2*c),112 + (4*c),0,0,0);
   changeTile16(1,0 + (2*c),114 + (4*c),0,0,0);
   changeTile16(0,1 + (2*c),113 + (4*c),0,0,0);
   changeTile16(1,1 + (2*c),115 + (4*c),0,0,0);
}
for (c=3;c<8;c++){
   changeTile16(0,0 + (2*c),8,0,0,0);
   changeTile16(1,0 + (2*c),10,0,0,0);
   changeTile16(0,1 + (2*c),9,0,0,0);
   changeTile16(1,1 + (2*c),11,0,0,0);
}
//Display the blank text squares on the 32 side
for(r=1;r<3;r++)
   for(c=1;c<12;c++)
      changeTile32(r,c,11,0,0,0);
changeTile32(3,1,11,0,0,0);

//Display the numbers along the top of the battle screen on the 32 side
for (c=0;c<10;c++)
   changeTile32(3,2+c,65+c,0,0,0);

//Display the letters along the side of the battle screen on the 32 side
for (r=0;r<10;r++)
   changeTile32(4+r,1,12+r,0,0,0);

//Display the ocean on the 32 side
for (r=0;r<10;r++)
   for (c=0;c<10;c++)
      changeTile32(r+4,c+2,5,0,0,0);

//Display the 16 side
```

```
    //Display the border on the 16 side.  Fill the entire side with border.  This will be overwritten
as necessary.
    for (r=1;r<15;r++){
        for (c=0;c<8;c++){
            changeTile16(r*2+0,0 + (2*c),8,0,0,0);
            changeTile16(r*2+1,0 + (2*c),10,0,0,0);
            changeTile16(r*2+0,1 + (2*c),9,0,0,0);
            changeTile16(r*2+1,1 + (2*c),11,0,0,0);
        }
    }
    //Display the blank text squares on the 16 side
    for (r=0;r<2;r++){
        for (c=0;c<11;c++){
            changeTile16(r+3, c+2,14,0,0,0);
            changeTile16(r+19, c+2,14,0,0,0);
            changeTile16(r+22, c+2,14,0,0,0);
            changeTile16(r+25, c+2,14,0,0,0);
        }
    }
    changeTile16(6,2,14,0,0,0);

    //Display the numbers along the top of the battle screen on the 16 side
    for (c=0;c<10;c++)
        changeTile16(6,3+c,68+c,0,0,0);

    //Display the letters along the side of the battle screen on the 16 side
    for (r=0;r<10;r++)
        changeTile16(7+r,2,15+r,0,0,0);

    //Display the ocean on the 16 side
    for (r=0;r<10;r++)
        for (c=0;c<10;c++)
            changeTile16(r+7,c+3,5,0,0,0);
    return;
}

//Get the players name and display it
char* getPlayerName(){
    //Set the cursor to the beginning of the ENTERNAME box
    cursorX = 1;
    cursorY = 2;
    invert32(cursorY, cursorX);
    char* name = "";
    displayString32("Enter Name:", 1, 1);
    char* input = "";
    char* buffer;
```

```
    input = "";
    input = getInput();
    printf("I have received input: %s\n", input);
    while (strcmp(input, "Return") != 0){
        printf("I am trying to receive another input because I have not yet detected Return\n");
        if (strcmp(input, "BackSpace") == 0 && cursorX > 1){
            if (cursorX < 12)
                invert32(cursorY, cursorX);
            cursorX = cursorX - 1;
            changeTile32(cursorY, cursorX, 11, 0, 0, 0);
            invert32(cursorY, cursorX);
            strncpy(name, name, cursorX - 1);
        }
        else if (cursorX <= 11 && strcmp(input, "BackSpace") && strcmp(input, "Left") &&
strcmp(input, "Right") && strcmp(input, "Up") && strcmp(input, "Down") && strcmp(input,
"Escape")){
            if (strlen(input) == 1){
                buffer = displayChar32(input[0], cursorY, cursorX);
                name = strcat(name, buffer);
                cursorX = cursorX + 1;
                if (cursorX < 12)
                    invert32(cursorY, cursorX);
            }
        }
        input = "";
        input = getInput();
    }
    printf("Name: %s\n", name);
    printf("I have received input: %s\n", input);
    return name;
}

void checkPlayAgain(){
    char * input;
    displayString32("Play Again?", 1, 1);
    displayString32("    Y N    ", 2, 1);
    cursorX = 5;
    cursorY = 2;
    invert32(cursorY, cursorX);
    exitPlayer = -1;
    while  (exitPlayer == -1){
        input = getInput();
        if (strcmp(input, "Left") == 0 && cursorX == 7){
            invert32(cursorY, cursorX);
            cursorX = 5;
            invert32(cursorY, cursorX);
```

```
        }
        else if (strcmp(input, "Right") == 0 && cursorX == 5){
            invert32(cursorY, cursorX);
            cursorX = 7;
            invert32(cursorY, cursorX);
        }
        else if (strcmp(input, "Return") == 0){
            invert32(cursorY, cursorX);
            if (cursorX == 5){
                exitPlayer = 0;
            }
            else{
                exitPlayer = 1;
            }
        }
    }
    return;
}

void placeCarrier(int  x,int  y,char* direction){
    if ( strcmp(direction , "Left") == 0 ){
        changeTile32(y, x, 75, 1, 1, 0);
        changeTile32(y, x-1, 76, 1, 1, 0);
        changeTile32(y, x-2, 77, 1, 1, 0);
        changeTile32(y, x-3, 78, 1, 1, 0);
        changeTile32(y, x-4, 79, 1, 1, 0);
        }

    else if (strcmp(direction , "Right") == 0){
        changeTile32(y, x, 75, 1, 0, 0);
        changeTile32(y, x+1, 76, 1, 0, 0);
        changeTile32(y, x+2, 77, 1, 0, 0);
        changeTile32(y, x+3, 78, 1, 0, 0);
        changeTile32(y, x+4, 79, 1, 0, 0);
        }

    else if (strcmp(direction , "Up") == 0){
        changeTile32(y, x, 75, 0, 1, 0);
        changeTile32(y-1, x, 76, 0, 1, 0);
        changeTile32(y-2, x, 77, 0, 1, 0);
        changeTile32(y-3, x, 78, 0, 1, 0);
        changeTile32(y-4, x, 79, 0, 1, 0);
        }

    else if (strcmp(direction , "Down") == 0){
        changeTile32(y, x, 75, 0, 0, 0);
```

```
        changeTile32(y+1, x, 76, 0, 0, 0);
        changeTile32(y+2, x, 77, 0, 0, 0);
        changeTile32(y+3, x, 78, 0, 0, 0);
        changeTile32(y+4, x, 79, 0, 0, 0);
        }
    return;
}

void placeBattleship(int x,int  y,char*  direction){
    if (strcmp(direction , "Left") == 0){
        changeTile32(y, x, 80, 1, 1, 0);
        changeTile32(y, x-1, 81, 1, 1, 0);
        changeTile32(y, x-2, 82, 1, 1, 0);
        changeTile32(y, x-3, 83, 1, 1, 0);
        }

    else if  (strcmp(direction , "Right") == 0) {
        changeTile32(y, x, 80, 1, 0, 0);
        changeTile32(y, x+1, 81, 1, 0, 0);
        changeTile32(y, x+2, 82, 1, 0, 0);
        changeTile32(y, x+3, 83, 1, 0, 0);
        }

    else if (strcmp(direction , "Up") == 0){
        changeTile32(y, x, 80, 0, 1, 0);
        changeTile32(y-1, x, 81, 0, 1, 0);
        changeTile32(y-2, x, 82, 0, 1, 0);
        changeTile32(y-3, x, 83, 0, 1, 0);
        }

    else if (strcmp(direction , "Down") == 0){
        changeTile32(y, x, 80, 0, 0, 0);
        changeTile32(y+1, x, 81, 0, 0, 0);
        changeTile32(y+2, x, 82, 0, 0, 0);
        changeTile32(y+3, x, 83, 0, 0, 0);
        }
    return;
}


void placeDestroyer(int x,int  y,char* direction){
    if (strcmp(direction , "Left") == 0){
        changeTile32(y, x, 84, 1, 1, 0);
        changeTile32(y, x-1, 85, 1, 1, 0);
        changeTile32(y, x-2, 86, 1, 1, 0);
        }
```

```
    else if (strcmp(direction , "Right") == 0){
        changeTile32(y, x, 84, 1, 0, 0);
        changeTile32(y, x+1, 85, 1, 0, 0);
        changeTile32(y, x+2, 86, 1, 0, 0);
        }

    else if (strcmp(direction , "Up") == 0){
        changeTile32(y, x, 84, 0, 1, 0);
        changeTile32(y-1, x, 85, 0, 1, 0);
        changeTile32(y-2, x, 86, 0, 1, 0);
        }

    else if (strcmp(direction , "Down") == 0){
        changeTile32(y, x, 84, 0, 0, 0);
        changeTile32(y+1, x, 85, 0, 0, 0);
        changeTile32(y+2, x, 86, 0, 0, 0);
        }
    return;
}

void placeSubmarine(int x,int  y,char*  direction){
    if (strcmp(direction , "Left") == 0){
        changeTile32(y, x, 87, 1, 1, 0);
        changeTile32(y, x-1, 88, 1, 1, 0);
        changeTile32(y, x-2, 89, 1, 1, 0);
        }

    else if (strcmp(direction , "Right") == 0){
        changeTile32(y, x, 87, 1, 0, 0);
        changeTile32(y, x+1, 88, 1, 0, 0);
        changeTile32(y, x+2, 89, 1, 0, 0);
        }

    else if (strcmp(direction , "Up") == 0){
        changeTile32(y, x, 87, 0, 1, 0);
        changeTile32(y-1, x, 88, 0, 1, 0);
        changeTile32(y-2, x, 89, 0, 1, 0);
        }

    else if (strcmp(direction , "Down") == 0){
        changeTile32(y, x, 87, 0, 0, 0);
        changeTile32(y+1, x, 88, 0, 0, 0);
        changeTile32(y+2, x, 89, 0, 0, 0);
        }
    return;
```

```
}

void placePatrolBoat(int x,int y,char* direction){
   if (strcmp(direction , "Left") == 0){
      changeTile32(y, x, 90, 1, 1, 0);
      changeTile32(y, x-1, 91, 1, 1, 0);
      }

   else if (strcmp(direction , "Right") == 0){
      changeTile32(y, x, 90, 1, 0, 0);
      changeTile32(y, x+1, 91, 1, 0, 0);
      }

   else if (strcmp(direction , "Up") == 0){
      changeTile32(y, x, 90, 0, 1, 0);
      changeTile32(y-1, x, 91, 0, 1, 0);
      }

   else if (strcmp(direction , "Down") == 0){
      changeTile32(y, x, 90, 0, 0, 0);
      changeTile32(y+1, x, 91, 0, 0, 0);
      }
   return;
}


void getLocation32(){
   char* input = "";
   //global cursorY, cursorX
   invert32(cursorY, cursorX);
   while ( strcmp(input , "Return") != 0 ){
      if ( (cursorX > 11) || (cursorX < 2) || (cursorY < 4) || (cursorY > 13) ){
         printf( "Invalid cursor position:  %d,%d", cursorX, cursorY);
         printf( "Resetting cursor");
         cursorX = 2;
         cursorY = 4;
         }
      if ( ( strcmp(input , "Left" ) == 0) && (cursorX > 2) ){
         invert32(cursorY, cursorX);
         cursorX = cursorX - 1;
         invert32(cursorY, cursorX);
         }
      else if ( ( strcmp(input , "Right" ) == 0) && (cursorX < 11) ) {
         invert32(cursorY, cursorX);
         cursorX = cursorX + 1;
         invert32(cursorY, cursorX);
```

```
    }
else if ( ( strcmp(input , "Up" ) == 0) &&  (cursorY > 4) ){
    invert32(cursorY, cursorX);
    cursorY = cursorY - 1;
    invert32(cursorY, cursorX);
    }
else if ( ( strcmp(input , "Down" ) == 0) &&  (cursorY < 13) ){
    invert32(cursorY, cursorX);
    cursorY = cursorY + 1;
    invert32(cursorY, cursorX);
    }

else if ( (strcmp(input , "A") == 0) || ( strcmp(input , "a") == 0) ) {
    invert32(cursorY, cursorX);
    cursorY = 4;
    invert32(cursorY, cursorX);
    }
else if ( strcmp(input , "1") == 0 ) {
    invert32(cursorY, cursorX);
    cursorX = 2;
    invert32(cursorY, cursorX);
    }

else if ( (strcmp(input , "B") == 0) || ( strcmp(input , "b") == 0) ){
    invert32(cursorY, cursorX);
    cursorY = 5;
    invert32(cursorY, cursorX);
    }
else if ( strcmp(input , "2") == 0 ){
    invert32(cursorY, cursorX);
    cursorX = 3;
    invert32(cursorY, cursorX);
    }

else if ( (strcmp(input , "C") == 0) || ( strcmp(input , "c") == 0) ){
    invert32(cursorY, cursorX);
    cursorY = 6;
    invert32(cursorY, cursorX);
    }
else if ( strcmp(input , "3") == 0 ){
    invert32(cursorY, cursorX);
    cursorX = 4;
    invert32(cursorY, cursorX);
    }

else if ( (strcmp(input , "D") == 0) || ( strcmp(input , "d") == 0) ){
```

```
    invert32(cursorY, cursorX);
    cursorY = 7;
    invert32(cursorY, cursorX);
    }
else if ( strcmp(input , "4") == 0 ){
    invert32(cursorY, cursorX);
    cursorX = 5;
    invert32(cursorY, cursorX);
    }

else if ( (strcmp(input , "E") == 0) || ( strcmp(input , "e") == 0) ){
    invert32(cursorY, cursorX);
    cursorY = 8;
    invert32(cursorY, cursorX);
    }
else if ( strcmp(input , "5") == 0 ){
    invert32(cursorY, cursorX);
    cursorX = 6;
    invert32(cursorY, cursorX);
    }

else if ( (strcmp(input , "F") == 0) || ( strcmp(input , "f") == 0) ){
    invert32(cursorY, cursorX);
    cursorY = 9;
    invert32(cursorY, cursorX);
    }
else if ( strcmp(input , "6") == 0 ){
    invert32(cursorY, cursorX);
    cursorX = 7;
    invert32(cursorY, cursorX);
    }

else if ( (strcmp(input , "G") == 0) || ( strcmp(input , "g") == 0) ){
    invert32(cursorY, cursorX);
    cursorY = 10;
    invert32(cursorY, cursorX);
    }
else if ( strcmp(input , "7") == 0 ){
    invert32(cursorY, cursorX);
    cursorX = 8;
    invert32(cursorY, cursorX);
    }

else if ( (strcmp(input , "H") == 0) || ( strcmp(input , "h") == 0) ){
    invert32(cursorY, cursorX);
    cursorY = 11;
```

```
            invert32(cursorY, cursorX);
            }
        else if ( strcmp(input , "8") == 0 ){
            invert32(cursorY, cursorX);
            cursorX = 9;
            invert32(cursorY, cursorX);
            }

        else if ( (strcmp(input , "I") == 0) || ( strcmp(input , "i") == 0) ){
            invert32(cursorY, cursorX);
            cursorY = 12;
            invert32(cursorY, cursorX);
            }
        else if ( strcmp(input , "9") == 0 ){
            invert32(cursorY, cursorX);
            cursorX = 10;
            invert32(cursorY, cursorX);
            }

        else if ( (strcmp(input , "J") == 0) || ( strcmp(input , "j") == 0) ){
            invert32(cursorY, cursorX);
            cursorY = 13;
            invert32(cursorY, cursorX);
            }
        else if ( strcmp(input , "0") == 0 ){
            invert32(cursorY, cursorX);
            cursorX = 11;
            invert32(cursorY, cursorX);
            }

        input = getInput();
    }
    return;
}

void getLocation16(){
    char* input = "";
    //global cursorY, cursorX
    invert16(cursorY, cursorX);
    while  ( strcmp(input , "Return") != 0 ){
        if ( (cursorX > 12) || (cursorX < 3) || (cursorY < 7) || (cursorY > 16) ){
            printf( "Invalid cursor position:  %d,%d", cursorX, cursorY);
            printf( "Resetting cursor");
            cursorX = 2;
            cursorY = 4;
            }
```

```
if (  (strcmp(input , "Left") == 0) && (cursorX > 3)  ){
   invert16(cursorY, cursorX);
   cursorX = cursorX - 1;
   invert16(cursorY, cursorX);
   }
else if (  (strcmp(input , "Right") == 0) && (cursorX < 12)  ){
   invert16(cursorY, cursorX);
   cursorX = cursorX + 1;
   invert16(cursorY, cursorX);
   }
else if ( (strcmp(input , "Up") == 0) && (cursorY > 7)  ){
   invert16(cursorY, cursorX);
   cursorY = cursorY - 1;
   invert16(cursorY, cursorX);
   }
else if (  (strcmp(input , "Down") == 0) && (cursorY < 16)  ){
   invert16(cursorY, cursorX);
   cursorY = cursorY + 1;
   invert16(cursorY, cursorX);
   }

else if ( (strcmp(input , "A") == 0) || ( strcmp(input , "a") == 0) ) {
   invert16(cursorY, cursorX);
   cursorY = 7;
   invert16(cursorY, cursorX);
   }
else if ( strcmp(input , "1") == 0 ) {
   invert16(cursorY, cursorX);
   cursorX = 3;
   invert16(cursorY, cursorX);
   }

else if ( (strcmp(input , "B") == 0) || ( strcmp(input , "b") == 0) ) {
   invert16(cursorY, cursorX);
   cursorY = 8;
   invert16(cursorY, cursorX);
   }
else if ( strcmp(input , "2") == 0 ) {
   invert16(cursorY, cursorX);
   cursorX = 4;
   invert16(cursorY, cursorX);
   }

else if ( (strcmp(input , "C") == 0) || ( strcmp(input , "c") == 0) ) {
   invert16(cursorY, cursorX);
   cursorY = 9;
```

```
    invert16(cursorY, cursorX);
    }
else if ( strcmp(input , "3") == 0 ) {
    invert16(cursorY, cursorX);
    cursorX = 5;
    invert16(cursorY, cursorX);
    }
else if ( (strcmp(input , "D") == 0) || ( strcmp(input , "d") == 0) ) {
    invert16(cursorY, cursorX);
    cursorY = 10;
    invert16(cursorY, cursorX);
    }
else if ( strcmp(input , "4") == 0 ) {
    invert16(cursorY, cursorX);
    cursorX = 6;
    invert16(cursorY, cursorX);
    }

else if ( (strcmp(input , "E") == 0) || ( strcmp(input , "e") == 0) ) {
    invert16(cursorY, cursorX);
    cursorY = 11;
    invert16(cursorY, cursorX);
    }
else if ( strcmp(input , "5") == 0 ) {
    invert16(cursorY, cursorX);
    cursorX = 7;
    invert16(cursorY, cursorX);
    }

else if ( (strcmp(input , "F") == 0) || ( strcmp(input , "f") == 0) ) {
    invert16(cursorY, cursorX);
    cursorY = 12;
    invert16(cursorY, cursorX);
    }
else if ( strcmp(input , "6") == 0 ) {
    invert16(cursorY, cursorX);
    cursorX = 8;
    invert16(cursorY, cursorX);
    }

else if ( (strcmp(input , "G") == 0) || ( strcmp(input , "g") == 0) ) {
    invert16(cursorY, cursorX);
    cursorY = 13;
    invert16(cursorY, cursorX);
    }
else if ( strcmp(input , "7") == 0 ) {
```

```
            invert16(cursorY, cursorX);
            cursorX = 9;
            invert16(cursorY, cursorX);
            }

        else if ( (strcmp(input , "H") == 0) || ( strcmp(input , "h") == 0) ) {
            invert16(cursorY, cursorX);
            cursorY = 14;
            invert16(cursorY, cursorX);
            }
        else if ( strcmp(input , "8") == 0 ) {
            invert16(cursorY, cursorX);
            cursorX = 10;
            invert16(cursorY, cursorX);
            }

        else if ( (strcmp(input , "I") == 0) || ( strcmp(input , "i") == 0) ) {
            invert16(cursorY, cursorX);
            cursorY = 15;
            invert16(cursorY, cursorX);
            }
        else if ( strcmp(input , "9") == 0 ) {
            invert16(cursorY, cursorX);
            cursorX = 11;
            invert16(cursorY, cursorX);
            }

        else if ( (strcmp(input , "J") == 0) || ( strcmp(input , "j") == 0) ) {
            invert16(cursorY, cursorX);
            cursorY = 16;
            invert16(cursorY, cursorX);
            }
        else if ( strcmp(input , "0") == 0 ) {
            invert16(cursorY, cursorX);
            cursorX = 12;
            invert16(cursorY, cursorX);
            }

        input = getInput();
        }
    return;
}

//Conduct the players turn
void playerTurn(){
    //global cursorX, cursorY, playerScore
```

```
int fired = 0;
char* message, response;
cursorX = 3;
cursorY = 7;
displayString16("Select shot", 3, 2);
displayString16("location:  ", 4, 2);
while (fired == 0){
   getLocation16();
   if (display16[cursorY][cursorX][0] == 5)
      fired = 1;
}
int x = cursorX - 2;
if (x == 10)
   x = 0;
int y = cursorY - 6;
if (y == 10)
   y = 0;
printf("Shot: %d %d", x, y);
sprintf(message, "Shot: %d %d", x, y);
int kkk;
for(kkk = 0; kkk < 5; kkk++){
   sendPacket(message);
}
response = 0;
while (response == 0){
   message = receivePacket();
   if (strlen(message) >= 3 && strncmp(message, "Hit", 3)==0){
      playerScore = playerScore + 1;
      displayString16("Hit        ", 3, 2);
      changeTile16(cursorY, cursorX, 6, 0, 0, 0);
      response = 1;
   }
   else if (strlen(message) >= 4 && strncmp(message, "Miss", 4)==0){
      displayString16("Miss       ", 3, 2);
      changeTile16(cursorY, cursorX, 7, 0, 0, 0);
      response = 1;
   }
   else if (strlen(message) >= 4 && strncmp(message, "Sunk", 4)==0){
      char ship = message[6];
      changeTile16(cursorY, cursorX, 6, 0, 0, 0);
      playerScore = playerScore + 1;
      displayString16("Sunk       ", 3, 2);
      if (ship == 'B')
         displayString16("Battleship ", 4, 2);

      else if (ship == 'A')
```

```
            displayString16("Carrier    ", 4, 2);

        else if (ship == 'D')
            displayString16("Destroyer  ", 4, 2);

        else if (ship == 'S')
            displayString16("Submarine  ", 4, 2);

        else if (ship == 'P')
            displayString16("Patrol Boat", 4, 2);
      }
   }
   printf("I have sent a shot and received a response!\n");
   return;
}

//Conduct the opponents turn
void opponentTurn(){
   //global opponentScore, carrierleft, battleshipleft, destroyerleft, submarineleft, patrolboatleft
   int receivedShot, x, y;
   char* message;
   receivedShot = 0;
   x = -1;
   y = -1;
   while (receivedShot == 0){
      message = receivePacket();
      printf("XXX\n");
      if (strlen(message) >= 9 && strncmp(message,"Shot", 4)==0){
         receivedShot = 1; printf("AAA\n");
         x = message[6] - '0';
         y = message[8] - '0';
         if (x == 0)
            x = 10;
         if (y == 0)
            y = 10;
      } printf("BBB\n");
   }
   printf("I have received a shot\n");

   if (display32[y + 3][x + 1][0] == 5){
      sendPacket("Miss");
      changeTile32(y + 3, x + 1, 7, 0, 0, 0); //Display Miss
   }
   else if (display32[y + 3][x + 1][0] >= 75 && display32[y + 3][x + 1][0] <= 91){  //If this is an
unexploded ship
      opponentScore = opponentScore + 1;
```

```
        if (display32[y + 3][x + 1][0] >= 75 && display32[y + 3][x + 1][0] <= 79){ //Carrier
            carrierleft = carrierleft - 1;
            if (carrierleft == 0){
                sendPacket("Sunk A");
                printf("Carrier sunk!\n");
            }
            else
                sendPacket("Hit");
        }
        else if (display32[y + 3][x + 1][0] >= 80 && display32[y + 3][x + 1][0] <= 83){
//Battleship
            battleshipleft = battleshipleft - 1;
            if (battleshipleft == 0){
                sendPacket("Sunk B");
                printf("Battleship sunk!\n");
            }
            else
                sendPacket("Hit");
        }
        else if (display32[y + 3][x + 1][0] >= 84 && display32[y + 3][x + 1][0] <= 86){ //Destroyer
            destroyerleft = destroyerleft - 1;
            if (destroyerleft == 0){
                sendPacket("Sunk D");
                printf("Destroyer sunk!\n");
            }
            else
                sendPacket("Hit");
        }
        else if (display32[y + 3][x + 1][0] >= 87 && display32[y + 3][x + 1][0] <= 89){
//Submarine
            submarineleft = submarineleft - 1;
            if (submarineleft == 0){
                sendPacket("Sunk S");
                printf("Submarine sunk!\n");
            }
            else
                sendPacket("Hit");
        }
        else if (display32[y + 3][x + 1][0] >= 90 && display32[y + 3][x + 1][0] <= 91){ //Patrol
Boat
            patrolboatleft = patrolboatleft - 1;
            if (patrolboatleft == 0){
                sendPacket("Sunk P");
                printf("Patrol Boat sunk!\n");
            }
            else
```

```
            sendPacket("Hit");

        sendPacket("Hit");
        changeTile32(y + 3, x + 1, display32[y+3][x+1][0] + 17, 0, 0, 0); //Display exploded ship
part
        }
    }
    else{
        printf("Invalid shot attempted\n");
            sendPacket("Invalid shot");
    }
    return;
}

//If the player wins
void playerVictory(){
    displayString16("Victory    ", 25, 2);
    return;
}

//If the opponent wins
void opponentVictory(){
    displayString32("Defeat     ", 25, 2);
    return;
}

int checkValidShip(int x, int y, int length, char* direction){
    printf("I am testing the validity of %d, %d for a ship of length %d heading %s\n", x, y, length,
direction);
    int i;
    int valid = 1;
    if (strcmp(direction, "Left")==0){
        if (x-length >= 1){
            for (i=x-length;i<x;i++)
                if (display32[y][i+1][0] != 5)
                    valid = 0;
        }
        else
            valid = 0;
    }

    else if (strcmp(direction, "Right")==0){
        if (x+length <= 12){
            for (i=x;i<x+length;i++)
                if (display32[y][i][0] != 5)
                    valid = 0;
```

```
        }
        else
            valid = 0;
    }

    else if (strcmp(direction, "Up")==0){
        if (y - length >= 3){
            for (i=y-length;i<y;i++)
                if (display32[i+1][x][0] != 5)
                    valid = 0;
        }
        else
            valid = 0;
    }

    else if (strcmp(direction, "Down")==0){
        if (y + length <= 14){
            for (i=y;i<y+length;i++)
                if (display32[i][x][0] != 5)
                    valid = 0;
        }
        else
            valid = 0;
    }
    if (valid == 0)
        printf("I have found this configuration to be invalid\n");
    else
        printf("I have found this configuration to be valid\n");
    return valid;
}

void initializeRound(){
    //global playerName, cursorX, cursorY, patrolboatleft, submarineleft, destroyerleft,
battleshipleft, carrierleft, playerScore, opponentScore

    //Initialize score and ship health

    patrolboatleft = 2;
    submarineleft = 3;
    destroyerleft = 3;
    battleshipleft = 4;
    carrierleft = 5;
    playerScore = 0;
    opponentScore = 0;
    char* input;
```

```
    //Place the Carrier
    displayString32("Place 5 sq ", 1, 1);
    displayString32("Carrier    ", 2, 1);
    cursorY = 4;
    cursorX = 2;
    int carrier = 0;
    printf("I am now going to try to get a location for the carrier\n");
    while (carrier == 0)
    {
       getLocation32();
       printf("I got a location: %d %d\n", cursorX, cursorY);
       if (checkValidShip(cursorX, cursorY, 5, "Left") || checkValidShip(cursorX, cursorY, 5,
"Right") || checkValidShip(cursorX, cursorY, 5, "Up") || checkValidShip(cursorX, cursorY, 5,
"Down")){
          carrier = 1;
       }
       else{
          printf("Invalid placement for carrier\n");
          invert32(cursorY, cursorX);
       }
    }
    printf("Valid location for carrier\n");
    carrier = 0;
    displayString32("Direction? ", 1, 1);
    displayString32("Length: 5  ", 2, 1);
    while (carrier == 0)
    {
       input = getInput();
       if (strcmp(input, "Left") == 0 && checkValidShip(cursorX, cursorY, 5, "Left")){
          placeCarrier(cursorX, cursorY, "Left");
          carrier = 1;
       }
       if (strcmp(input, "Right") == 0 && checkValidShip(cursorX, cursorY, 5, "Right")){
          placeCarrier(cursorX, cursorY, "Right");
          carrier = 1;
       }
       if (strcmp(input, "Up") == 0 && checkValidShip(cursorX, cursorY, 5, "Up")){
          placeCarrier(cursorX, cursorY, "Up");
          carrier = 1;
       }
       if (strcmp(input, "Down") == 0 && checkValidShip(cursorX, cursorY, 5, "Down")){
          placeCarrier(cursorX, cursorY, "Down");
          carrier = 1;
       }
    }
    printf("You placed a carrier!\n");
```

```
    //Place the Battleship
    displayString32("Place 4 sq ", 1, 1);
    displayString32("Battleship ", 2, 1);
    cursorY = 4;
    cursorX = 2;
    int battleship = 0;
    printf("I am now going to try to get a location for the battleship\n");
    while (battleship == 0)
    {
        getLocation32();
        printf("I got a location: %d, %d\n", cursorX, cursorY);
        if (checkValidShip(cursorX, cursorY, 4, "Left") || checkValidShip(cursorX, cursorY, 4,
"Right") || checkValidShip(cursorX, cursorY, 4, "Up") || checkValidShip(cursorX, cursorY, 4,
"Down")){
            battleship = 1;
        }
        else{
            printf("Invalid placement for battleship\n");
            invert32(cursorY, cursorX);
        }
    }
    printf("Valid location for battleship\n");
    battleship = 0;
    displayString32("Direction? ", 1, 1);
    displayString32("Length: 4  ", 2, 1);
    while (battleship == 0){
        input = getInput();
        if (strcmp(input, "Left") == 0 && checkValidShip(cursorX, cursorY, 4, "Left")){
            placeBattleship(cursorX, cursorY, "Left");
            battleship = 1;
        }
        if (strcmp(input, "Right") == 0 && checkValidShip(cursorX, cursorY, 4, "Right")){
            placeBattleship(cursorX, cursorY, "Right");
            battleship = 1;
        }
        if (strcmp(input, "Up") == 0 && checkValidShip(cursorX, cursorY, 4, "Up")){
            placeBattleship(cursorX, cursorY, "Up");
            battleship = 1;
        }
        if (strcmp(input, "Down") == 0 && checkValidShip(cursorX, cursorY, 4, "Down")){
            placeBattleship(cursorX, cursorY, "Down");
            battleship = 1;
        }
    }
    printf("You placed a battleship!\n");
```

```
    //Place the Destroyer
    displayString32("Place 3 sq ", 1, 1);
    displayString32("Destroyer  ", 2, 1);
    cursorY = 4;
    cursorX = 2;
    int destroyer = 0;
    printf("I am now going to try to get a location for the destroyer\n");
    while (destroyer == 0)
    {
        getLocation32();
        printf("I got a location: %d, %d\n", cursorX, cursorY);
        if (checkValidShip(cursorX, cursorY, 3, "Left") || checkValidShip(cursorX, cursorY, 3,
"Right") || checkValidShip(cursorX, cursorY, 3, "Up") || checkValidShip(cursorX, cursorY, 3,
"Down")){
            destroyer = 1;
        }
        else{
            printf("Invalid placement for destroyer\n");
            invert32(cursorY, cursorX);
        }
    }

    printf("Valid location for destroyer\n");
    destroyer = 0;
    displayString32("Direction? ", 1, 1);
    displayString32("Length: 3  ", 2, 1);
    while (destroyer == 0)
    {
        input = getInput();
        if (strcmp(input, "Left") == 0 && checkValidShip(cursorX, cursorY, 3, "Left")){
            placeDestroyer(cursorX, cursorY, "Left");
            destroyer = 1;
        }
        if (strcmp(input, "Right") == 0 && checkValidShip(cursorX, cursorY, 3, "Right")){
            placeDestroyer(cursorX, cursorY, "Right");
            destroyer = 1;
        }
        if (strcmp(input, "Up") == 0 && checkValidShip(cursorX, cursorY, 3, "Up")){
            placeDestroyer(cursorX, cursorY, "Up");
            destroyer = 1;
        }
        if (strcmp(input, "Down") == 0 && checkValidShip(cursorX, cursorY, 3, "Down")){
            placeDestroyer(cursorX, cursorY, "Down");
            destroyer = 1;
        }
```

```
    }
    printf("You placed a destroyer!\n");

    //Place the Submarine
    displayString32("Place 3 sq ", 1, 1);
    displayString32("Submarine  ", 2, 1);
    cursorY = 4;
    cursorX = 2;
    int submarine = 0;
    printf("I am now going to try to get a location for the submarine\n");
    while (submarine == 0)
    {
        getLocation32();
        printf("I got a location: %d, %d\n", cursorX, cursorY);
        if (checkValidShip(cursorX, cursorY, 3, "Left") || checkValidShip(cursorX, cursorY, 3,
"Right") || checkValidShip(cursorX, cursorY, 3, "Up") || checkValidShip(cursorX, cursorY, 3,
"Down")){
            submarine = 1;
        }
        else{
            printf("Invalid placement for submarine");
            invert32(cursorY, cursorX);
        }
    }
    printf("Valid location for submarine\n");
    submarine = 0;
    displayString32("Direction? ", 1, 1);
    displayString32("Length: 3  ", 2, 1);
    while (submarine == 0)
    {
        input = getInput();
        if (strcmp(input, "Left") == 0 && checkValidShip(cursorX, cursorY, 3, "Left")){
            placeSubmarine(cursorX, cursorY, "Left");
            submarine = 1;
        }
        if (strcmp(input, "Right") == 0 && checkValidShip(cursorX, cursorY, 3, "Right")){
            placeSubmarine(cursorX, cursorY, "Right");
            submarine = 1;
        }
        if (strcmp(input, "Up") == 0 && checkValidShip(cursorX, cursorY, 3, "Up")){
            placeSubmarine(cursorX, cursorY, "Up");
            submarine = 1;
        }
        if (strcmp(input, "Down") == 0 && checkValidShip(cursorX, cursorY, 3, "Down")){
            placeSubmarine(cursorX, cursorY, "Down");
            submarine = 1;
```

```
        }
    }
    printf("You placed a submarine!\n");

    //Place the Patrol Boat
    displayString32("Place 2 sq ", 1, 1);
    displayString32("Patrol Boat", 2, 1);
    cursorY = 4;
    cursorX = 2;
    int patrolboat = 0;
    printf("I am now going to try to get a location for the patrol boat\n");
    while (patrolboat == 0)
    {
        getLocation32();
        printf("I got a location: %d, %d", cursorX, cursorY);
        if(checkValidShip(cursorX, cursorY, 2, "Left") || checkValidShip(cursorX, cursorY, 2,
"Right") || checkValidShip(cursorX, cursorY, 2, "Up") || checkValidShip(cursorX, cursorY, 2,
"Down")){
            patrolboat = 1;
        }
        else{
            printf("Invalid placement for patrol boat\n");
            invert32(cursorY, cursorX);
        }
    }
    printf("Valid location for patrol boat\n");
    patrolboat = 0;
    displayString32("Direction? ", 1, 1);
    displayString32("Length: 2  ", 2, 1);
    while(patrolboat == 0)
    {
        input = getInput();
        if (strcmp(input, "Left") == 0 && checkValidShip(cursorX, cursorY, 2, "Left")){
            placePatrolBoat(cursorX, cursorY, "Left");
            patrolboat = 1;
        }
        if (strcmp(input, "Right") == 0 && checkValidShip(cursorX, cursorY, 2, "Right")){
            placePatrolBoat(cursorX, cursorY, "Right");
            patrolboat = 1;
        }
        if (strcmp(input, "Up") == 0 && checkValidShip(cursorX, cursorY, 2, "Up")){
            placePatrolBoat(cursorX, cursorY, "Up");
            patrolboat = 1;
        }
        if (strcmp(input, "Down") == 0 && checkValidShip(cursorX, cursorY, 2, "Down")){
            placePatrolBoat(cursorX, cursorY, "Down");
```

```
            patrolboat = 1;
        }
    }
    printf("You placed a patrol boat!\n");

    displayString16(playerName, 19, 2);
    displayString16(opponentName, 22, 2);
    displayString16("00 Hits", 20, 4);
    displayString16("00 Hits", 23, 4);

    //Exchange Ethernet Handshake with other computer.  Send playerName and ready signal, and
wait for both to come back.
    //Skipping handshake to show 16 bit cursor
    /*sendPacket(strcat("Player Name ", playerName));
    sendPacket("Ready");
    char* message1 = receivePacket();
    int receivedName = 0;
    while (receivedName == 0){
        if (strlen(message1) >= 12 && strcmp(message1, "Player Name") == 0){
            strcpy(opponentName, message1+12);
            receivedName = 1;
        }
        else
            message1 = receivePacket();
    }
    char* message2 = receivePacket();
    int receivedReady = 0;
    while (receivedReady == 0){
        if (strcmp(message1, "Ready") == 0)
            receivedReady = 1;
        else
            message2 = receivePacket();
    }
    displayString16(playerName, 19, 2);
    displayString16(opponentName, 22, 2);
    return;*/
}


//Setup a game
void setupGame(){
    printf("I am initializing the display now\n");
    initializeDisplay();
    exitGame = 0;
    exitRound = 0;
    playerName = getPlayerName();
```

```
   //return;
}



//Play a game.  Initialize the round, then play a round.
void playGame(){

   initializeRound(); //This sets up the ships and waits until both players are ready

   while (exitRound == 0){
      //Play the game.  exitRound should be set to 1 if a player wins, 2 if a player quits the game
using the quit condition (implement), 3 if the game goes out of sync. or 4 if another error occurs
      if (turn == 0){ //If players turn, enact that.
         playerTurn();
         turn = 1; //Set turn to opponent.
         }
      else if (turn == 1){ //If opponent turn, enact that.
         opponentTurn();
         turn = 0; //Set turn to player.
         }
      else{ //Error condition.  Should never occur.
         exitRound = 4;
      }
      if (playerScore >= 17 || opponentScore >= 17)
         exitRound = 1;
      }
   //End of game loop

   //If game ended in victory
   if (exitRound == 1){
      if (playerScore == 17) //Player win condition.  Enact player victory.
         playerVictory();
      else if (opponentScore == 17) //Opponent win condition.  Enact opponent victory.
         opponentVictory();
   }
   //If player chose to exit game.
   else if (exitRound == 2){
      printf("Player quit game.  Goodbye!\n");
      sendPacket("Player Quit");
   }

   //If game falls out of sync.
   else if (exitRound == 3){
      printf("Game out of sync.  Game ending.  Sorry!\n");
      sendPacket("Out of Sync Quit");
```

```
    }
    //If other error condition occurs
    else{
        printf("Game error.  Game ending.  Sorry!\n") ;
        sendPacket("Game Error Quit");
    }
    return;
}

int main()
{

  // initialize ethernet
  DM9000_init(mac_address);
  interrupt_number = 0;
  alt_irq_register(DM9000A_IRQ, NULL, (void*)ethernet_interrupt_handler);  // listen to port
  printf("\nEthernet Initialized\n");

  // sprite and vga initialization
  write_sprite_data_to_memory();

  initializeDisplay();
  printf("\nDisplay Initialized\n");
  exitPlayer = 1;
  while (exitGame == 0){
        printf("I have determined that the game is not over\n");
        if (exitPlayer == 1){
            printf("I am setting up a game for new players\n");
            setupGame();
        }
        printf("I am going to play a new round now\n");
        playGame();
        printf("I am going to check if the player wants to play again\n");
        checkPlayAgain();
    }
    printf("I am going to quit the game now\n");
    return 0;

  return 0;
}
```

VHDL CODE:

Battleship: Battleship.vhd

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;

entity battleship is

  port (
    -- Clocks

    CLOCK_27,                          -- 27 MHz
    CLOCK_50,                          -- 50 MHz
    EXT_CLOCK : in std_logic;                  -- External Clock

    -- Buttons and switches

    KEY : in std_logic_vector(3 downto 0);        -- Push buttons
    SW : in std_logic_vector(17 downto 0);        -- DPDT switches

    --  displays

    HEX0, HEX1, HEX2, HEX3, HEX4, HEX5, HEX6, HEX7 -- 7-segment displays
      : out std_logic_vector(6 downto 0);
    LEDG : out std_logic_vector(8 downto 0);       -- Green LEDs
    LEDR : out std_logic_vector(17 downto 0);      -- Red LEDs

    -- RS-232 interface

    UART_TXD : out std_logic;                  -- UART transmitter
    UART_RXD : in std_logic;                   -- UART receiver

    -- IRDA interface

--    IRDA_TXD : out std_logic;                  -- IRDA Transmitter
    IRDA_RXD : in std_logic;                   -- IRDA Receiver

    -- SDRAM

    DRAM_DQ : inout std_logic_vector(15 downto 0); -- Data Bus
    DRAM_ADDR : out std_logic_vector(11 downto 0); -- Address Bus
    DRAM_LDQM,                                 -- Low-byte Data Mask
    DRAM_UDQM,                                 -- High-byte Data Mask
    DRAM_WE_N,                                 -- Write Enable
```

```
DRAM_CAS_N,                         -- Column Address Strobe
DRAM_RAS_N,                         -- Row Address Strobe
DRAM_CS_N,                          -- Chip Select
DRAM_BA_0,                          -- Bank Address 0
DRAM_BA_1,                          -- Bank Address 0
DRAM_CLK,                           -- Clock
DRAM_CKE : out std_logic;                   -- Clock Enable

-- FLASH

FL_DQ : inout std_logic_vector(7 downto 0);      -- Data bus
FL_ADDR : out std_logic_vector(21 downto 0);  -- Address bus
FL_WE_N,                            -- Write Enable
FL_RST_N,                           -- Reset
FL_OE_N,                            -- Output Enable
FL_CE_N : out std_logic;                    -- Chip Enable

-- SRAM

SRAM_DQ : inout std_logic_vector(15 downto 0); -- Data bus 16 Bits
SRAM_ADDR : out std_logic_vector(17 downto 0); -- Address bus 18 Bits
SRAM_UB_N,                          -- High-byte Data Mask
SRAM_LB_N,                          -- Low-byte Data Mask
SRAM_WE_N,                          -- Write Enable
SRAM_CE_N,                          -- Chip Enable
SRAM_OE_N : out std_logic;                  -- Output Enable

-- USB controller

OTG_DATA : inout std_logic_vector(15 downto 0); -- Data bus
OTG_ADDR : out std_logic_vector(1 downto 0);    -- Address
OTG_CS_N,                           -- Chip Select
OTG_RD_N,                           -- Write
OTG_WR_N,                           -- Read
OTG_RST_N,                          -- Reset
OTG_FSPEED,                 -- USB Full Speed, 0 = Enable, Z = Disable
OTG_LSPEED : out std_logic;     -- USB Low Speed, 0 = Enable, Z = Disable
OTG_INT0,                           -- Interrupt 0
OTG_INT1,                           -- Interrupt 1
OTG_DREQ0,                          -- DMA Request 0
OTG_DREQ1 : in std_logic;                   -- DMA Request 1
OTG_DACK0_N,                        -- DMA Acknowledge 0
OTG_DACK1_N : out std_logic;                -- DMA Acknowledge 1

-- 16 X 2 LCD Module
```

```
LCD_ON,                 -- Power ON/OFF
LCD_BLON,                 -- Back Light ON/OFF
LCD_RW,                 -- Read/Write Select, 0 = Write, 1 = Read
LCD_EN,                -- Enable
LCD_RS : out std_logic;     -- Command/Data Select, 0 = Command, 1 = Data
LCD_DATA : inout std_logic_vector(7 downto 0); -- Data bus 8 bits

-- SD card interface

SD_DAT,                 -- SD Card Data
SD_DAT3,                 -- SD Card Data 3
SD_CMD : inout std_logic;   -- SD Card Command
SD_CLK : out std_logic;     -- SD Card Clock

-- USB JTAG link

TDI,                  -- CPLD -> FPGA (data in)
TCK,                   -- CPLD -> FPGA (clk)
TCS : in std_logic;         -- CPLD -> FPGA (CS)
TDO : out std_logic;        -- FPGA -> CPLD (data out)

-- I2C bus

I2C_SDAT : inout std_logic; -- I2C Data
I2C_SCLK : out std_logic;   -- I2C Clock

-- PS/2 port

PS2_DAT,                -- Data
PS2_CLK : in std_logic;     -- Clock

-- VGA output

VGA_CLK,                                -- Clock
VGA_HS,                                -- H_SYNC
VGA_VS,                                -- V_SYNC
VGA_BLANK,                                -- BLANK
VGA_SYNC : out std_logic;                        -- SYNC
VGA_R,                                -- Red[9:0]
VGA_G,                                -- Green[9:0]
VGA_B : out unsigned(9 downto 0);                -- Blue[9:0]

--  Ethernet Interface

ENET_DATA : inout std_logic_vector(15 downto 0);    -- DATA bus 16Bits
ENET_CMD,          -- Command/Data Select, 0 = Command, 1 = Data
```

```
    ENET_CS_N,                          -- Chip Select
    ENET_WR_N,                           -- Write
    ENET_RD_N,                          -- Read
    ENET_RST_N,                          -- Reset
    ENET_CLK : out std_logic;                 -- Clock 25 MHz
    ENET_INT : in std_logic;                 -- Interrupt


    -- Audio CODEC

    AUD_ADCLRCK : inout std_logic;              -- ADC LR Clock
    AUD_ADCDAT : in std_logic;                 -- ADC Data
    AUD_DACLRCK : inout std_logic;              -- DAC LR Clock
    AUD_DACDAT : out std_logic;                -- DAC Data
    AUD_BCLK : inout std_logic;                -- Bit-Stream Clock
    AUD_XCK : out std_logic;                   -- Chip Clock

    -- Video Decoder

    TD_DATA : in std_logic_vector(7 downto 0);  -- Data bus 8 bits
    TD_HS,                              -- H_SYNC
    TD_VS : in std_logic;                    -- V_SYNC
    TD_RESET : out std_logic;                -- Reset

    -- General-purpose I/O

    GPIO_0,                             -- GPIO Connection 0
    GPIO_1 : inout std_logic_vector(35 downto 0) -- GPIO Connection 1
    );

end battleship;

architecture  battleship_rtl of battleship is
        signal red, green, blue : std_logic_vector(9 downto 0) := "1111111111";
        signal clk_50                   : std_logic;
        signal BA, DQM                          : std_logic_vector(1 downto 0);
        signal wrap_addr                : std_logic_vector(18 downto 0);
        signal wrap_rgb                         : std_logic_vector(7 downto 0);
        signal tmp_ledg                         : std_logic_vector(7 downto 0);
        signal clock_25,  reset_n                       : std_logic;
        signal counter : unsigned (15 downto 0);
begin

        PLL: entity work.pll port map(
                inclk0  =>      CLOCK_50,
                c0              => DRAM_CLK,
```

```
                c1              => clk_50,
                c2              => ENET_CLK
        );

  process (CLOCK_50)
  begin
    if rising_edge(CLOCK_50) then
      if counter = x"ffff" then
        reset_n <= '1';
      else
        reset_n <= '0';
        counter <= counter + 1;
      end if;
    end if;
  end process;
        nios: entity work.nios_system port map(
          -- 1) global s:
             clk => clk_50,
--                                  clk_sdram => clk_50,
             reset_n => reset_n,

                                  -- The PS2 KeyBoard
                                    PS2_Clk_to_the_ps2          => PS2_CLK,
             PS2_Data_to_the_ps2     => PS2_DAT,

                                  -- The SDRAM
                                    zs_addr_from_the_sdram    => DRAM_ADDR,
             zs_ba_from_the_sdram            => BA,
             zs_cas_n_from_the_sdram         => DRAM_CAS_N,
             zs_cke_from_the_sdram           => DRAM_CKE,
             zs_cs_n_from_the_sdram => DRAM_CS_N,
             zs_dq_to_and_from_the_sdram   => DRAM_DQ,
             zs_dqm_from_the_sdram           => DQM,
             zs_ras_n_from_the_sdram         => DRAM_RAS_N,
             zs_we_n_from_the_sdram          => DRAM_WE_N,

                                       user_addr_in_to_the_sram => wrap_addr,
                                       user_data_out_from_the_sram => wrap_rgb,

          -- the_sram
             SRAM_ADDR_from_the_sram => SRAM_ADDR,
             SRAM_CE_N_from_the_sram => SRAM_CE_N,
             SRAM_DQ_to_and_from_the_sram => SRAM_DQ,
             SRAM_LB_N_from_the_sram => SRAM_LB_N,
             SRAM_OE_N_from_the_sram => SRAM_OE_N,
             SRAM_UB_N_from_the_sram => SRAM_UB_N,
```

```
        SRAM_WE_N_from_the_sram => SRAM_WE_N,

                        -- VGA Wraparound
                        rgb_vector_in_to_the_vga => wrap_rgb,
                        sprite_addr_out_from_the_vga => wrap_addr,

       -- the_vga_md
          VGA_BLANK_from_the_vga => VGA_BLANK,
          VGA_B_from_the_vga => blue,
          VGA_CLK_from_the_vga => VGA_CLK,
          VGA_G_from_the_vga => green,
          VGA_HS_from_the_vga => VGA_HS,
          VGA_R_from_the_vga => red,
          VGA_SYNC_from_the_vga => VGA_SYNC,
          VGA_VS_from_the_vga => VGA_VS,

       -- the_DM9000A
          ENET_CMD_from_the_DM9000A => ENET_CMD,
          ENET_CS_N_from_the_DM9000A => ENET_CS_N,
          ENET_DATA_to_and_from_the_DM9000A => ENET_DATA,
          ENET_INT_to_the_DM9000A =>ENET_INT,
          ENET_RD_N_from_the_DM9000A => ENET_RD_N,
          ENET_RST_N_from_the_DM9000A => ENET_RST_N,
          ENET_WR_N_from_the_DM9000A => ENET_WR_N

       );

      DRAM_BA_0 <= BA(0);
      DRAM_BA_1 <= BA(1);
      DRAM_UDQM <= DQM(1);
      DRAM_LDQM <= DQM(0);


      VGA_R <= unsigned(red);
      VGA_G <= unsigned(green);
      VGA_B <= unsigned(blue);

--    tmp_ledg <= wrap_rgb when wrap_addr(17 downto 0) = SW else tmp_ledg;
--    LEDG <= "0" & tmp_ledg;


--    with wrap_addr(17 downto 0) select LEDG <=
--            "0" & wrap_rgb when SW
--            else LEDG;

  HEX7    <= "0000000"; -- Leftmost
```

```
  HEX6     <= "0001000";
  HEX5     <= "0000111";
  HEX4     <= "1000111";
  HEX3     <= "0010010";
  HEX2     <= "0001001";
  HEX1     <= "1001111";
  HEX0     <= "0001100";         -- Rightmost
  --LEDG     <= (others => '1');
  LEDR     <= (others => '1');
  LCD_ON   <= '1';
  LCD_BLON <= '1';
  LCD_RW <= '1';
  LCD_EN <= '0';
  LCD_RS <= '0';

  SD_DAT3 <= '1';
  SD_CMD <= '1';
  SD_CLK <= '1';

--  SRAM_DQ <= (others => 'Z');
--  SRAM_ADDR <= (others => '0');
--  SRAM_UB_N <= '1';
--  SRAM_LB_N <= '1';
--  SRAM_CE_N <= '1';
--  SRAM_WE_N <= '1';
--  SRAM_OE_N <= '1';

  UART_TXD <= '0';
--  DRAM_ADDR <= (others => '0');
--  DRAM_LDQM <= '0';
--  DRAM_UDQM <= '0';
--  DRAM_WE_N <= '1';
--  DRAM_CAS_N <= '1';
--  DRAM_RAS_N <= '1';
--  DRAM_CS_N <= '1';
--  DRAM_BA_0 <= '0';
--  DRAM_BA_1 <= '0';
--  DRAM_CLK <= '0';
--  DRAM_CKE <= '0';
  FL_ADDR <= (others => '0');
  FL_WE_N <= '1';
  FL_RST_N <= '0';
  FL_OE_N <= '1';
  FL_CE_N <= '1';
  OTG_ADDR <= (others => '0');
  OTG_CS_N <= '1';
```

```
OTG_RD_N <= '1';
OTG_RD_N <= '1';
OTG_WR_N <= '1';
OTG_RST_N <= '1';
OTG_FSPEED <= '1';
OTG_LSPEED <= '1';
OTG_DACK0_N <= '1';
OTG_DACK1_N <= '1';

TDO <= '0';

--ENET_CMD <= '0';
--ENET_CS_N <= '1';
--ENET_WR_N <= '1';
--ENET_RD_N <= '1';
--ENET_RST_N <= '1';
--ENET_CLK <= '0';

 -- DM9000A 25 MHz Clock
        process(clock_50 )
        begin
                if rising_edge (clock_50) then
                        clock_25 <= not clock_25;
                end if;
        end process;

        --ENET_CLK <= clock_25;

TD_RESET <= '0';

I2C_SCLK <= '1';

AUD_DACDAT <= '1';
AUD_XCK <= '1';

-- Set all bidirectional ports to tri-state
DRAM_DQ     <= (others => 'Z');
FL_DQ       <= (others => 'Z');
SRAM_DQ     <= (others => 'Z');
OTG_DATA    <= (others => 'Z');
LCD_DATA    <= (others => 'Z');
SD_DAT      <= 'Z';
I2C_SDAT    <= 'Z';
ENET_DATA   <= (others => 'Z');
AUD_ADCLRCK <= 'Z';
AUD_DACLRCK <= 'Z';
```

```
  AUD_BCLK    <= 'Z';
  GPIO_0      <= (others => 'Z');
  GPIO_1      <= (others => 'Z');
end battleship_rtl;
```

VGA Tiles and Image manipulation: vga_micro_driver.vhd

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;
--E: VGA Counter
entity vga_counter is

  port (
    reset : in std_logic;
    clk   : in std_logic;                  -- Should be 25.125 MHz

    VGA_CLK,                      -- Clock
    VGA_HS,                      -- H_SYNC
    VGA_VS,                      -- V_SYNC
    VGA_BLANK,                       -- BLANK
    VGA_SYNC : out std_logic;        -- SYNC


    VGA_X       : out unsigned(9 downto 0);
          VGA_Y          : out unsigned(8 downto 0);
          VGA_32         : out std_logic;
          VGA_16         : out std_logic
    );
end vga_counter;

architecture vga_counter_rtl of vga_counter is
  -- Video parameters

  constant HTOTAL       : integer := 800;
  constant HSYNC        : integer := 96;
  constant HBACK_PORCH  : integer := 48;
  constant HACTIVE      : integer := 640;
  constant HFRONT_PORCH : integer := 16;

  constant VTOTAL       : integer := 525;
  constant VSYNC        : integer := 2;
  constant VBACK_PORCH  : integer := 33;
  constant VACTIVE      : integer := 480;
  constant VFRONT_PORCH : integer := 10;

  constant HZONE                : integer := 384;
  constant LAG                          : integer := 5;
```

```
  -- Signals for the video controller
  signal Hcount, HCarryCount : unsigned(9 downto 0);  -- Horizontal position (0-800)
  signal Vcount : unsigned(9 downto 0);  -- Vertical position (0-524)
  signal EndOfLine, EndOfField : std_logic;

  signal vga_hblank, vga_hsync,
                   vga_vblank, vga_vsync,
                   vga_32_zone, vga_16_zone : std_logic;  -- Sync. signals

begin
  HCounter : process (clk)
  begin
    if rising_edge(clk) then
      if reset = '1' then
        Hcount <= (others => '0');
      elsif EndOfLine = '1' then
        Hcount <= (others => '0');
      else
        Hcount <= Hcount + 1;
      end if;
    end if;
  end process HCounter;

  EndOfLine <= '1' when Hcount = HTOTAL - 1 else '0';

  VCounter: process (clk)
  begin
    if rising_edge(clk) then
      if reset = '1' then
        Vcount <= (others => '0');
      elsif EndOfLine = '1' then
        if EndOfField = '1' then
          Vcount <= (others => '0');
        else
          Vcount <= Vcount + 1;
        end if;
      end if;
    end if;
  end process VCounter;

  EndOfField <= '1' when Vcount = VTOTAL - 1 else '0';

  -- State machines to generate HSYNC, VSYNC, HZONE

  HSyncGen : process (clk)
  begin
```

```
  if rising_edge(clk) then
    if reset = '1' or EndOfLine = '1' then
      vga_hsync <= '1';
    elsif Hcount = HSYNC - 1 then
      vga_hsync <= '0';
    end if;
  end if;
end process HSyncGen;


VSyncGen : process (clk)
begin
  if rising_edge(clk) then
    if reset = '1' then
      vga_vsync <= '1';
    elsif EndOfLine ='1' then
      if EndOfField = '1' then
        vga_vsync <= '1';
      elsif Vcount = VSYNC - 1 then
        vga_vsync <= '0';
      end if;
    end if;
  end if;
end process VSyncGen;

HZoneGen : process (clk)
begin
  if rising_edge(clk) then
    if reset = '1' then
      vga_32_zone <= '0';
                  vga_16_zone <= '0';
    elsif Hcount = HSYNC + HBACK_PORCH - LAG then
      vga_32_zone <= '1';
                  vga_16_zone <= '0';
          elsif Hcount = HSYNC + HBACK_PORCH + HZONE - LAG then
      vga_32_zone <= '0';
                  vga_16_zone <= '1';
    elsif Hcount = HSYNC + HBACK_PORCH + HACTIVE - LAG then
      vga_32_zone <= '0';
                  vga_16_zone <= '0';
    end if;
          HCarryCount <= HCount;
  end if;
end process HZoneGen;
```

```
--  HBlankGen : process (clk)
--  begin
--    if rising_edge(clk) then
--      if reset = '1' then
--        vga_hblank <= '1';
--      elsif Hcount = HSYNC + HBACK_PORCH then
--        vga_hblank <= '0';
--      elsif Hcount = HSYNC + HBACK_PORCH + HACTIVE then
--        vga_hblank <= '1';
--      end if;
--    end if;
--  end process HBlankGen;

  VBlankGen : process (clk)
  begin
    if rising_edge(clk) then
      if reset = '1' then
        vga_vblank <= '1';
      elsif EndOfLine = '1' then
        if Vcount = VSYNC + VBACK_PORCH - 1 then
          vga_vblank <= '0';
        elsif Vcount = VSYNC + VBACK_PORCH + VACTIVE - 1 then
          vga_vblank <= '1';
        end if;
      end if;
    end if;
  end process VBlankGen;




  VGA_CLK <= clk;
  VGA_HS <= not vga_hsync;
  VGA_VS <= not vga_vsync;
  VGA_SYNC <= '0';
  VGA_BLANK <= not (vga_hsync or vga_vsync);

  VGA_X <= unsigned(HCarryCount - HSYNC - HBACK_PORCH + LAG);
  VGA_Y <= unsigned(VCount - VSYNC - VBACK_PORCH)(8 downto 0);
  VGA_32 <= vga_32_zone AND NOT vga_vblank;
  VGA_16 <= vga_16_zone AND NOT vga_vblank;

end vga_counter_rtl;
----------------------------------------
library ieee;
use ieee.std_logic_1164.all;
```

```
use ieee.numeric_std.all;
entity ctoa_converter_32 is
	port(
		clk			: in	std_logic;
		c_vert,
		c_horiz		: in	unsigned(4 downto 0);
		ctrl_in		: in	std_logic_vector(1 downto 0);
		x_in, y_in	: in	unsigned(4 downto 0);
		ctrl_out	: out	std_logic_vector(1 downto 0);
		x_out, y_out: out	unsigned(4 downto 0);
		addr		: out	unsigned(10 downto 0)
	);
end ctoa_converter_32;

architecture ctoa_32_arch of ctoa_converter_32 is
--	signal  vert_8_32, vert_4_32, horiz_0_32,
--			vert_16_16, horiz_0_16  : unsigned(10 downto 0) := "000" & X"00";
--	signal addr_32, addr_16		: unsigned(10 downto 0) := "000" & X"00";
--	signal ctrl				: std_logic_vector(1 downto 0);
	signal addr_32, addr_16		: unsigned(10 downto 0) := "000" & X"00";
	signal x_temp, y_temp			: unsigned(4 downto 0);
	signal ctrl				: std_logic_vector(1 downto 0);
begin
	process(clk)
	begin
		if rising_edge(clk) then
			if ctrl_in = "10" then
				addr	<= ("00" &(("0" & c_vert(4 downto 1) & "0") + ("00" &
c_vert(4 downto 1)) + ("0000" & c_horiz(4 downto 3))) & c_horiz(2 downto 1) & "0");
			elsif ctrl_in = "01" then
				addr	<= (("0" & c_vert & (NOT c_horiz(3)) & c_horiz(2)) + 45)
& c_horiz(1 downto 0) & "0";
			else
				addr	<= "00000000000";
			end if;
			x_out	<= x_in;
			y_out	<= y_in;
			ctrl_out<= ctrl_in;
		end if;
	end process;

--	process(clk)
--	begin
--		if rising_edge(clk) then
--			addr_32		<= ("00" &(("0" & c_vert(4 downto 1) & "0") +
("00" & c_vert(4 downto 1)) + ("0000" & c_horiz(4 downto 3))) & c_horiz(2 downto 1) & "0");
```

```
--                                  addr_16         <= (("0" & c_vert & (NOT c_horiz(3)) &
c_horiz(2)) + 45) & c_horiz(1 downto 0) & "0";
--
--                                  x_temp<= x_in;
--                                  y_temp<= y_in;
--
--                                  ctrl      <= ctrl_in;
--                  end if;
--          end process;
--          x_out <= x_temp;
--          y_out <= y_temp;
--          ctrl_out <= ctrl;
--          with ctrl select addr <=
--                  addr_32 when "10",
--                  addr_16 when "01",
--                  "00000000000" when others;
end ctoa_32_arch;

----------------------------------------

library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;
entity ram_2_wrapper is
        port(
                clk,
                reset,
                read,
                write,
                chipselect      :       in std_logic;
                address         :       in std_logic_vector(9 downto 0);
                readdata        :       out std_logic_vector(15 downto 0);
                writedata       :       in std_logic_vector(15 downto 0);

                read_addr       :       in std_logic_vector(9 downto 0);
                ctrl_in         :       in      std_logic_vector(1 downto 0);
                ctrl_out        :       out     std_logic_vector(1 downto 0);
                sprite_addr     :       out std_logic_vector(9 downto 0);

                x_in, y_in      :       in      unsigned(4 downto 0);
                x_out, y_out:   out unsigned(4 downto 0)
   );
end ram_2_wrapper;

architecture ram_2_wrap_rtl of ram_2_wrapper is
        signal ram_out:         std_logic_vector(15 downto 0);
```

```
	signal ctrl		:	std_logic_vector(1 downto 0);
	signal tmp_ctrl :	std_logic_vector(1 downto 0);
	signal tmp_x,
			tmp_y	:	unsigned(4 downto 0);

	signal we, re, oe	:	std_logic;
	signal trd, rd	:	std_logic_vector(15 downto 0);
begin
	oe <= tmp_ctrl(1) OR tmp_ctrl(0);

	process(chipselect, read, write, oe)
	begin
		if chipselect = '1' then
			if write = '1' then
				ctrl <= "11";
			elsif read = '1' then
				ctrl <= "10";
			else
				ctrl <= "00";
			end if;
		elsif oe = '1' then
			ctrl <= "01";
		else
			ctrl <= "00";
		end if;
	end process;

	we <= '1' when ctrl = "11" else '0';


	MRAM: entity work.ram_2_port port map(
		clock		=> clk,
		address_a	=> std_logic_vector(address),
		data_a		=> writedata,
		wren_a		=> we,
		q_a			=> trd,
		address_b	=> std_logic_vector(read_addr),
		data_b		=> X"0000",
		wren_b		=> '0',
		q_b			=> ram_out
	);
	process(clk)
	begin
		if rising_edge(clk) then
			ctrl_out <= tmp_ctrl;
			tmp_ctrl <= ctrl_in;
```

```
                        x_out <= tmp_x;
                        y_out <= tmp_y;
                        tmp_x <= x_in;
                        tmp_y <= y_in;

                        re <= read;
                end if;
        end process;

        rd <= trd when re = '1' else rd;
        readdata <= rd;
        with oe select sprite_addr <=
                ram_out(9 downto 0) when '1',
                X"00" & "00" when '0';

end ram_2_wrap_rtl;
---------------------------------------

library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;
entity rotate_32 is
        port(
                clk                     : in    std_logic;
                SPRITE_IN       : in    std_logic_vector(9 downto 0);
                CTRL_IN                 : in    std_logic_vector(1 downto 0);
                VERT_IN,
                HORIZ_IN        : in    unsigned(4 downto 0);
                RT                      : in    std_logic_vector(1 downto 0);
                SPRITE_OUT      : out   std_logic_vector(9 downto 0);
                CTRL_OUT        : out   std_logic_vector(1 downto 0);
                VERT_OUT,
                HORIZ_OUT       : out   unsigned(4 downto 0)
        );
end rotate_32;

architecture rotate_32_rtl of rotate_32 is
begin
        process(clk)
        begin
                if rising_edge(clk) then
                        SPRITE_OUT <= SPRITE_IN;
                        CTRL_OUT <= CTRL_IN;
                        if RT = "11" then
                                VERT_OUT <= NOT HORIZ_IN;
```

```
                        HORIZ_OUT <= VERT_IN;
                    elsif RT = "10" then
                        VERT_OUT <= NOT VERT_IN;
                        HORIZ_OUT <= NOT HORIZ_IN;
                    elsif RT = "01" then
                        VERT_OUT <= HORIZ_IN;
                        HORIZ_OUT <= NOT VERT_IN;
                    else
                        VERT_OUT <= VERT_IN;
                        HORIZ_OUT <= HORIZ_IN;
                    end if;
                end if;
        end process;
end rotate_32_rtl;

---------------------------------

library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;
entity sprite_rom_p is
        port(
                clk,
                reset           : in    std_logic;
                s_addr          : in    unsigned(6 downto 0);
                c_vert,
                c_horiz         : in    unsigned(4 downto 0);
                invert          : in    std_logic;
                s16                     : in    std_logic;
                addr_out        : out   unsigned(18 downto 0);
                RGB_IN                  : in    std_logic_vector(7 downto 0);
                RGB_OUT                 : out   std_logic_vector(7 downto 0)
        );
end sprite_rom_p;


architecture sprite_arch_p of sprite_rom_p is
        signal p_addr : unsigned(18 downto 0);
        signal int_invert, int_reset : std_logic;
begin
        --p_addr <= "0100" & s_addr & c_vert(3 downto 0) & c_horiz(3 downto 0) when s16 =
'1'
        --                      else "00" & s_addr & c_vert & c_horiz;
        process(clk)
        begin
                if rising_edge(clk) then
```

```
                        if s16 = '1' then
                                addr_out <= "0100" & s_addr & c_vert(3 downto 0) & c_horiz(3
downto 0);
                        else
                                addr_out <= "00" & s_addr & c_vert & c_horiz;
                        end if;
                        --addr_out <= std_logic_vector(p_addr);
                        int_invert <= invert;
                        int_reset <= reset;
                        if int_reset = '0' then
                                RGB_OUT <= X"00";
                        elsif int_invert = '1' then
                                RGB_OUT <= NOT RGB_IN;
                        else
                                RGB_OUT <= RGB_IN;
                        end if;
                end if;
        end process;
end sprite_arch_p;


---------------------------------


library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;
entity rgb_to_vga is
        port(
                RGB_HIGH    : in      std_logic;
                RGB_IN              : in      std_logic_vector(2 downto 0);
                VGA_OUT             : out     unsigned(9 downto 0)
        );
end rgb_to_vga;

architecture rgb_to_vga_rtl of rgb_to_vga is
begin
        with RGB_HIGH select VGA_OUT <=
                (others => '1') when '1',
                unsigned(RGB_IN & "0000000") when '0';
end rgb_to_vga_rtl;

---------------------------------
```

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;
entity vga_micro_driver is

  port (                    -- External Clock
    clk                : in  std_logic;
    reset_n    : in  std_logic;
    read       : in  std_logic;
    write      : in  std_logic;
    chipselect : in  std_logic;
    address    : in  unsigned(9 downto 0);
    readdata   : out std_logic_vector(15 downto 0);
    writedata  : in  std_logic_vector(15 downto 0);

          -- Loop around signals
          sprite_addr_out : out std_logic_vector(18 downto 0);
          rgb_vector_in  : in  std_logic_vector(7 downto 0);

    -- VGA output

    VGA_CLK,                                  -- Clock
    VGA_HS,                                  -- H_SYNC
    VGA_VS,                                  -- V_SYNC
    VGA_BLANK,                                    -- BLANK
    VGA_SYNC : out std_logic;                         -- SYNC
    VGA_R,                                  -- Red[9:0]
    VGA_G,                                  -- Green[9:0]
    VGA_B : out unsigned(9 downto 0)                  -- Blue[9:0]
    );

end vga_micro_driver;


architecture datapath of vga_micro_driver is

          signal clk50 : std_logic := '0';
          signal clk25 : std_logic := '0';

          signal coord_x : unsigned(9 downto 0);
          signal coord_y : unsigned(8 downto 0);
          signal pixel_x, pixel_y,
                    pixel_temp_x, pixel_temp_y: unsigned(4 downto 0);
          signal enable_32, enable_16,
                              temp_32, temp_16      : std_logic;
```

```
	signal  ctrl_2,
			ctrl_3,
			ctrl_4  : std_logic_vector(1 downto 0);
	signal  pix_x_2,
			pix_y_2		: unsigned(4 downto 0);



	signal mem_addr: unsigned(10 downto 0);
	signal out_val, out_val_2: std_logic_vector(15 downto 0) := X"0000";
	signal sig_RGB_32, sig_RGB_16, sig_RGB, sig_RGB_PRC: std_logic_vector(7 downto
0);
	signal  sig_red, sig_green, sig_blue,
			sig_red_x, sig_green_x, sig_blue_x,
			sig_red_y, sig_green_y, sig_blue_y   : unsigned(9 downto 0);

begin

	clk50 <= clk;

	CLK_25: process(clk)
	begin
		if rising_edge(clk) then
			clk25 <= NOT clk25;
		end if;
	end process CLK_25;

--2 Clock Cycles (due to clk25)
	VC: entity work.vga_counter port map(
		reset		=> '0' ,
		clk		=> clk25,

		VGA_CLK	=> VGA_CLK,
		VGA_HS		=> VGA_HS,
		VGA_VS		=> VGA_VS,
		VGA_BLANK	=> VGA_BLANK,
		VGA_SYNC	=> VGA_SYNC,

		VGA_X		=> coord_x,
		VGA_Y		=> coord_y,
		VGA_32		=> enable_32,
		VGA_16		=> enable_16
   );

--Clock Cycle
```

```
	ctoa: entity work.ctoa_converter_32 port map(
		clk			=> clk50,

		c_vert		=> coord_y(8 downto 4),
		c_horiz		=> coord_x(8 downto 4),
		ctrl_in		=> enable_32 & enable_16,
		ctrl_out	=> ctrl_2,
		x_in		=> coord_x(4 downto 0),
		y_in		=> coord_y(4 downto 0),
		x_out		=> pix_x_2,
		y_out		=> pix_y_2,
		addr		=> mem_addr
	);

--2 Clock Cycles
	PRAM: entity work.ram_2_wrapper port map(
		clk				=> clk50,
		reset			=> reset_n,
		read			=> read,
		write			=> write,
		chipselect		=> chipselect,
		address			=> std_logic_vector(address),
		readdata		=> readdata,
		writedata		=> writedata,

		read_addr		=> std_logic_vector(mem_addr(10 downto 1)),
		ctrl_in			=> ctrl_2,
		ctrl_out		=> ctrl_3,
		sprite_addr		=> out_val(9 downto 0),
		x_in			=> pix_x_2,
		y_in			=> pix_y_2,
		x_out			=> pixel_temp_x,
		y_out			=> pixel_temp_y
  );

--Clock Cycle
	rotate: entity work.rotate_32 port map(
		clk		=> clk50,
		VERT_IN		=> pixel_temp_y,
		HORIZ_IN	=> pixel_temp_x,
		SPRITE_IN	=> out_val(9 downto 0),
		CTRL_IN		=> ctrl_3,
		RT => out_val(8 downto 7),
		SPRITE_OUT=> out_val_2(9 downto 0),
		CTRL_OUT	=> ctrl_4,
		VERT_OUT	=> pixel_y,
```

```
		HORIZ_OUT  => pixel_x
	);

--2 Clock Cycles
	sprite_rom_p: entity work.sprite_rom_p port map (
		clk => clk50,
		reset => ctrl_4(1) OR ctrl_4(0),
		s_addr => unsigned(out_val_2(6 downto 0)),
		c_vert => pixel_y,
		c_horiz => pixel_x,
		invert => out_val_2(9),
		s16 => ctrl_4(0),
		std_logic_vector(addr_out) => sprite_addr_out,
		RGB_IN => rgb_vector_in,
		RGB_OUT => sig_RGB_PRC
	);

	vga_red: entity work.rgb_to_vga port map (
		RGB_IN => sig_RGB_PRC(7 downto 5),
		RGB_HIGH => sig_RGB_PRC(7) AND sig_RGB_PRC(6) AND
sig_RGB_PRC(5),
		VGA_OUT => sig_red
	);
	vga_green: entity work.rgb_to_vga port map (
		RGB_IN => sig_RGB_PRC(4 downto 2),
		RGB_HIGH => sig_RGB_PRC(4) AND sig_RGB_PRC(3) AND
sig_RGB_PRC(2),
		VGA_OUT => sig_green
	);
	vga_blue: entity work.rgb_to_vga port map (
		RGB_IN => '0' & sig_RGB_PRC(1 downto 0),
		RGB_HIGH => sig_RGB_PRC(1) AND sig_RGB_PRC(0),
		VGA_OUT => sig_blue
	);

	--2 Clock Cycle
	STALL: process(clk50)
	begin
		if rising_edge(clk50) then
			VGA_R <= sig_red_x;
			VGA_G <= sig_green_x;
			VGA_B <= sig_blue_x;
			sig_red_x <= sig_red_y;
			sig_green_x <= sig_green_y;
			sig_blue_x <= sig_blue_y;
			sig_red_y <= sig_red;
```

```
                        sig_green_y <= sig_green;
                        sig_blue_y <= sig_blue;
                end if;
        end process STALL;


--      VGA_R <= sig_red;
--      VGA_G <= sig_green;
--      VGA_B <= sig_blue;
end datapath;


--RAM Megafunction


LIBRARY ieee;
USE ieee.std_logic_1164.all;

LIBRARY altera_mf;
USE altera_mf.all;

ENTITY ram_2_port IS
        PORT
        (
                address_a               : IN STD_LOGIC_VECTOR (9 DOWNTO 0);
                address_b               : IN STD_LOGIC_VECTOR (9 DOWNTO 0);
                clock           : IN STD_LOGIC ;
                data_a          : IN STD_LOGIC_VECTOR (15 DOWNTO 0);
                data_b          : IN STD_LOGIC_VECTOR (15 DOWNTO 0);
                wren_a          : IN STD_LOGIC  := '1';
                wren_b          : IN STD_LOGIC  := '1';
                q_a             : OUT STD_LOGIC_VECTOR (15 DOWNTO 0);
                q_b             : OUT STD_LOGIC_VECTOR (15 DOWNTO 0)
        );
END ram_2_port;


ARCHITECTURE SYN OF ram_2_port IS

        SIGNAL sub_wire0   : STD_LOGIC_VECTOR (15 DOWNTO 0);
        SIGNAL sub_wire1   : STD_LOGIC_VECTOR (15 DOWNTO 0);



        COMPONENT altsyncram
        GENERIC (
```

```
		address_reg_b		: STRING;
		clock_enable_input_a		: STRING;
		clock_enable_input_b		: STRING;
		clock_enable_output_a		: STRING;
		clock_enable_output_b		: STRING;
		indata_reg_b		: STRING;
		intended_device_family		: STRING;
		lpm_type		: STRING;
		numwords_a		: NATURAL;
		numwords_b		: NATURAL;
		operation_mode		: STRING;
		outdata_aclr_a		: STRING;
		outdata_aclr_b		: STRING;
		outdata_reg_a		: STRING;
		outdata_reg_b		: STRING;
		power_up_uninitialized		: STRING;
		read_during_write_mode_mixed_ports		: STRING;
		widthad_a		: NATURAL;
		widthad_b		: NATURAL;
		width_a		: NATURAL;
		width_b		: NATURAL;
		width_byteena_a		: NATURAL;
		width_byteena_b		: NATURAL;
		wrcontrol_wraddress_reg_b		: STRING
	);
	PORT (
			wren_a: IN STD_LOGIC ;
			clock0 : IN STD_LOGIC ;
			wren_b: IN STD_LOGIC ;
			address_a	: IN STD_LOGIC_VECTOR (9 DOWNTO 0);
			address_b	: IN STD_LOGIC_VECTOR (9 DOWNTO 0);
			q_a	: OUT STD_LOGIC_VECTOR (15 DOWNTO 0);
			q_b	: OUT STD_LOGIC_VECTOR (15 DOWNTO 0);
			data_a : IN STD_LOGIC_VECTOR (15 DOWNTO 0);
			data_b : IN STD_LOGIC_VECTOR (15 DOWNTO 0)
	);
	END COMPONENT;

BEGIN
	q_a	<= sub_wire0(15 DOWNTO 0);
	q_b	<= sub_wire1(15 DOWNTO 0);

	altsyncram_component : altsyncram
	GENERIC MAP (
		address_reg_b => "CLOCK0",
		clock_enable_input_a => "BYPASS",
```

```
		clock_enable_input_b => "BYPASS",
		clock_enable_output_a => "BYPASS",
		clock_enable_output_b => "BYPASS",
		indata_reg_b => "CLOCK0",
		intended_device_family => "Cyclone II",
		lpm_type => "altsyncram",
		numwords_a => 1024,
		numwords_b => 1024,
		operation_mode => "BIDIR_DUAL_PORT",
		outdata_aclr_a => "NONE",
		outdata_aclr_b => "NONE",
		outdata_reg_a => "CLOCK0",
		outdata_reg_b => "CLOCK0",
		power_up_uninitialized => "FALSE",
		read_during_write_mode_mixed_ports => "DONT_CARE",
		widthad_a => 10,
		widthad_b => 10,
		width_a => 16,
		width_b => 16,
		width_byteena_a => 1,
		width_byteena_b => 1,
		wrcontrol_wraddress_reg_b => "CLOCK0"
	)
	PORT MAP (
		wren_a => wren_a,
		clock0 => clock,
		wren_b => wren_b,
		address_a => address_a,
		address_b => address_b,
		data_a => data_a,
		data_b => data_b,
		q_a => sub_wire0,
		q_b => sub_wire1
	);



END SYN;
```

SRAM Controller: de2_sram_controller.vhd

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;

entity de2_sram_controller is

  port (
        clk       :         in std_logic;

    signal chipselect : in std_logic;
    signal write, read : in std_logic;
    signal address  :  in std_logic_vector(17 downto 0);
    signal readdata : out std_logic_vector(15 downto 0);
    signal writedata : in std_logic_vector(15 downto 0);
    signal byteenable : in std_logic_vector(1 downto 0);

        signal user_addr_in : in std_logic_vector(18 downto 0);
        signal user_data_out : out std_logic_vector(7 downto 0);

    signal SRAM_DQ   : inout std_logic_vector(15 downto 0);
    signal SRAM_ADDR : out std_logic_vector(17 downto 0);
    signal SRAM_UB_N, SRAM_LB_N : out std_logic;
    signal SRAM_WE_N, SRAM_CE_N : out std_logic;
    signal SRAM_OE_N            : out std_logic
    );

end de2_sram_controller;

architecture dp of de2_sram_controller is
begin

  process(chipselect)
  begin
        if chipselect = '1' then
                if write = '1' then
                        SRAM_DQ <= writedata;
                else
                        SRAM_DQ <= (others => 'Z');
                end if;
                readdata <= SRAM_DQ;
                SRAM_ADDR <= address;
                SRAM_UB_N <= not byteenable(1);
                SRAM_LB_N <= not byteenable(0);
                SRAM_WE_N <= not write;
```

```
                SRAM_CE_N <= not chipselect;
                SRAM_OE_N <= not read;
        else
                SRAM_DQ <= (others => 'Z');
                if user_addr_in(0) = '1' then
                        user_data_out <= SRAM_DQ(15 downto 8);
                else
                        user_data_out <= SRAM_DQ(7 downto 0);
                end if;
                SRAM_ADDR <= user_addr_in(18 downto 1);
                SRAM_UB_N <= '0';
                SRAM_LB_N <= '0';
                SRAM_WE_N <= '1';
                SRAM_CE_N <= '0';
                SRAM_OE_N <= '0';
        end if;
  end process;

end dp;
```

PS2 Keyboard: de2_ps2.vhd **NOTE: This file is mostly not our work.**

We only changed a few characters in it to make it work with our project.

```
--------------------------------------------------------------------
--
-- Simple (receive-only) PS/2 controller for the Altera Avalon bus
--
-- Presents a two-word interface:
--
-- Byte 0: LSB is a status bit: 1 = data received, 0 = no new data
-- Byte 4: least significant byte is received data,
--         reading it clears the input register
--
-- Make sure "Slave addressing" in the interfaces tab of SOPC Builder's
-- "New Component" dialog is set to "Register" mode.
--
--
-- Stephen A. Edwards and Yingjian Gu
-- Columbia University, sedwards@cs.columbia.edu
--
-- From an original by Bert Cuzeau
-- (c) ALSE. http://www.alse-fr.com
--
--------------------------------------------------------------------

-- ------------------------------------------------
--   Simplified PS/2 Controller  (kbd, mouse...)
-- ------------------------------------------------
-- Only the Receive function is implemented !
-- (c) ALSE. http://www.alse-fr.com
-- Author : Bert Cuzeau.
-- Fully synchronous solution, same Filter on PS2_Clk.
-- Still as compact as "Plain_wrong"...
-- Possible improvement : add TIMEOUT on PS2_Clk while shifting
-- Note: PS2_Data is resynchronized though this should not be
-- necessary (qualified by Fall_Clk and does not change at that time).
-- Note the tricks to correctly interpret 'H' as '1' in RTL simulation.

library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;

entity PS2_Ctrl is
  port(
    Clk       : in  std_logic;  -- System Clock
```

```
    Reset     : in  std_logic;  -- System Reset
    PS2_Clk   : in  std_logic;  -- Keyboard Clock Line
    PS2_Data  : in  std_logic;  -- Keyboard Data Line
    DoRead    : in  std_logic;  -- From outside when reading the scan code
    Scan_Err  : out std_logic;  -- To outside : Parity or Overflow error
    Scan_DAV  : out std_logic;  -- To outside when a scan code has arrived
    Scan_Code : out unsigned(7 downto 0) -- Eight bits Data Out
    );
end PS2_Ctrl;

architecture rtl of PS2_Ctrl is

  signal PS2_Datr  : std_logic;

  subtype Filter_t is unsigned(7 downto 0);
  signal Filter    : Filter_t;
  signal Fall_Clk  : std_logic;
  signal Bit_Cnt   : unsigned (3 downto 0);
  signal Parity    : std_logic;
  signal Scan_DAVi : std_logic;

  signal S_Reg     : unsigned(8 downto 0);

  signal PS2_Clk_f : std_logic;

  Type   State_t is (Idle, Shifting);
  signal State : State_t;

begin

  Scan_DAV <= Scan_DAVi;

-- This filters digitally the raw clock signal coming from the keyboard :
--  * Eight consecutive PS2_Clk=1 makes the filtered_clock go high
--  * Eight consecutive PS2_Clk=0 makes the filtered_clock go low
-- Implies a (FilterSize+1) x Tsys_clock delay on Fall_Clk wrt Data
-- Also in charge of the re-synchronization of PS2_Data

  process (Clk)
  begin
    if rising_edge(Clk) then
      if Reset = '1' then
        PS2_Datr  <= '0';
        PS2_Clk_f <= '0';
        Filter    <= (others => '0');
        Fall_Clk  <= '0';
```

```
      else
        PS2_Datr <= PS2_Data and PS2_Data; -- also turns 'H' into '1'
        Fall_Clk <= '0';
        Filter   <= (PS2_Clk and PS2_CLK) & Filter(Filter'high downto 1);
        if Filter = Filter_t'(others=>'1') then
          PS2_Clk_f <= '1';
        elsif Filter = Filter_t'(others=>'0') then
          PS2_Clk_f <= '0';
          if PS2_Clk_f = '1' then
            Fall_Clk <= '1';
          end if;
        end if;
      end if;
    end if;
  end process;

-- This simple State Machine reads in the Serial Data
-- coming from the PS/2 peripheral.

  process(Clk)
  begin
    if rising_edge(Clk) then
      if Reset = '1' then
        State     <= Idle;
        Bit_Cnt   <= (others => '0');
        S_Reg     <= (others => '0');
        Scan_Code <= (others => '0');
        Parity    <= '0';
        Scan_DAVi <= '0';
        Scan_Err  <= '0';
      else

        if DoRead = '1' then
          Scan_DAVi <= '0'; -- note: this assgnmnt can be overriden
        end if;

        case State is

          when Idle =>
            Parity  <= '0';
            Bit_Cnt <= (others => '0');
            -- note that we do not need to clear the Shift Register
            if Fall_Clk='1' and PS2_Datr='0' then -- Start bit
              Scan_Err <= '0';
              State <= Shifting;
            end if;
```

```
        when Shifting =>
          if Bit_Cnt >= 9 then
            if Fall_Clk = '1' then -- Stop Bit
              -- Error is (wrong Parity) or (Stop='0') or Overflow
              Scan_Err  <= (not Parity) or (not PS2_Datr) or Scan_DAVi;
              Scan_Davi <= '1';
              Scan_Code <= S_Reg(7 downto 0);
              State <= Idle;
            end if;
          elsif Fall_Clk = '1' then
            Bit_Cnt <= Bit_Cnt + 1;
            S_Reg <= PS2_Datr & S_Reg (S_Reg'high downto 1); -- Shift right
            Parity <= Parity xor PS2_Datr;
          end if;

        when others => -- never reached
          State <= Idle;

      end case;

      --Scan_Err <= '0'; -- to create a deliberate error

    end if;

  end if;

 end process;

end rtl;

--------------------------------------------------------------------------------

library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;

entity de2_ps2 is

 port (
  avs_s1_clk        : in std_logic;
  avs_s1_reset      : in std_logic;
  avs_s1_address    : in unsigned(0 downto 0);
  avs_s1_read       : in std_logic;
  avs_s1_chipselect : in std_logic;
  avs_s1_readdata   : out unsigned(7 downto 0);
```

```
    PS2_Clk          : in std_logic;
    PS2_Data         : in std_logic
    );
end de2_ps2;

architecture rtl of de2_ps2 is

  signal Data         : unsigned(7 downto 0);
  signal DataAvailable : std_logic;
  signal DoRead        : std_logic;

begin

  U1: entity work.PS2_CTRL port map(
    Clk       => avs_s1_clk,
    Reset     => not avs_s1_reset,
    DoRead    => DoRead,
    PS2_Clk   => PS2_Clk,
    PS2_Data  => PS2_Data,
    Scan_Code => Data,
    Scan_DAV  => DataAvailable );

  process (avs_s1_clk)
  begin
    if rising_edge(avs_s1_clk) then
      DoRead <= avs_s1_read and avs_s1_chipselect and avs_s1_address(0);
    end if;
  end process;

  process (Data, DataAvailable, avs_s1_address, avs_s1_chipselect)
  begin
    if avs_s1_chipselect = '1' then
      if avs_s1_address(0) = '1' then
        avs_s1_readdata <= Data;
      else
        avs_s1_readdata <= "0000000" & DataAvailable;
      end if;
    else
      avs_s1_readdata <= "00000000";
    end if;
  end process;

end rtl;
```