WiiMaze

Final Report

Shaun Salzberg
Mack Lu
Brian Ramos

Embedded Systems
5/10/08

Section I. Project Proposal

The Original, Ambitious Proposal
We propose to build an interactive marble maze game using the Nintendo WiiMote and a game called Triple Labyrinth. The player will control the tilt of the maze board by tilting the WiiMote up, down, left, and right. By doing so, the player will have to navigate a ball through a maze while avoiding numerous holes into which the ball may fall.

We will first connect the WiiMote to the DE2 board via a USB Bluetooth receiver, and then connect the DE2 board to the game using our own custom circuit that will extend from the serial port. We will have to write our own driver for the Bluetooth receiver, and will have to research the specifications of the WiiMote. The circuit we build will entail two motors that tilt the board in the X and Y directions.

The Revised, More Modest Proposal
We realize that getting the USB to work with Bluetooth to work with the WiiMote is a challenging task, so we also propose to accomplish various sub-projects as we work our way up to accomplishing the complete "ambitious proposal."

Iteration 1: Understand the USB protocol and attach a simple USB device to the board
After doing some research on the USB protocol and skimming the official 650-page specification of it, we have realized that USB is extremely difficult and interesting in its own right, and would be happy just to be able get a simple USB device such as a laser mouse to work with the board. This will entail writing VHDL code to interact with the USB mouse and perhaps even to make a pointer move on the VGA screen. We also realize that no other group has ever made use of the USB port, so we find it exciting to be the first.

Iteration 2: Understand the Bluetooth protocol and connect a USB Bluetooth dongle to the board
From what we have seen in our initial research, Bluetooth is also a non-trivial protocol. Thus, we would also be happy to simply be able to send and receive simple packets via Bluetooth. Once we have the USB port working from iteration 1, we will attempt to connect a Bluetooth USB dongle to the board and see if we can at least get an ACK from the device. Once we can do that, we will attempt to do more intricate parsing of the packets, and deal with the various parts such as the SYNC and PID.

Iteration 3: Understand the WiiMote specs and connect it to the board
The WiiMote uses the Bluetooth protocol to send and receive message, so our next step would be to figure out exactly what packets the WiiMote sends when, say, the user presses the A button, or tilts it 45 degrees to the left. Once we understand this, connecting the WiiMote to the board via our USB Bluetooth dongle will be fairly straightforward. At this stage, we will simply try to get the board to turn on an LED when a button on the WiiMote is pressed.

Iteration 4: Connect the board to the maze game via the serial port

If we get this far, we would be ecstatic. Once our WiiMote can communicate with the DE2 board, we can begin to use it to do some neat things. We have already done some research on how to use the serial port to control motors, so we intend to build this circuit and attach the motors to the marble maze game. We will then write some more VHDL code to translate the packets coming in from Bluetooth to motor movements.

The Final, Most Modest Proposal

In the interest of not reinventing the wheel, we may decide to boot uCLinux onto the DE2 board so that we may use it's built in USB driver. We believe that we would also be able to find modules for Bluetooth support that would make connecting the WiiMote to the board much easier. If we take this approach, a large amount of our efforts will be to modify and optimize the USB and Bluetooth drivers for our particular project. Also, rather than hooking the WiiMote up to a physical board with motors, we may decide to implement this game as a video game using the VGA screen. We will thus need to write our own VGA controller hardware in VHDL.

Section II. Project Design

Project Goals

With the last, most modest proposal in mind, the goals of this project were two-fold:

1. Connect the Nintendo WiiMote up to the DE2 board using uCLinux.
2. Implement a video game that uses the WiiMote as an input device.

Section IIa. The Hardware

Overview of the WiiMaze Game

In the WiiMaze game, the player uses the WiiMote to navigate a character through a maze. The player's aim is to find and move the character to the "goal" position in the maze, which is marked in each level by a blue cross-hair. In addition to not being able to move through typical maze walls, the player also cannot move through any of the colored (red, blue, or green) "doors" that may be present in the maze. In order to open these doors and be able to pass into that particular section of the maze, the player must first find and collect a "key" of the same color, which will be hidden at some other location in the maze. As an added difficulty, in many levels, the entire maze is not visible at once -- only a small "window" of tiles around the character's current position will be visible. Thus, the player must move the character around to explore the maze in order to build up a mental map of where things are. Once the player successfully navigates the character to the goal, the player will be taken to the next level.

Overview of the Hardware Needed to Display the Game

First, it is important to note that the only part of the hardware that we modified was the VGA controller -- all other components came directly from the DE2_NET project that Altera provides. So, at a quick glance, the VGA controller that we wrote for our project is custom tailored to display our game's graphics. The major components of it are the

numerous built in tiles, or bit arrays, that describe our maze sections (a left wall or top-right corner for example), our objects (a red key or right, blue door for example), our character, and the goal position. The controller also stores the size of the window at the center of the screen. It is the VGA controller's job, then, given the positions and sizes of all of these tiles as set by the software, to display everything properly on screen.

Hardware Architecture
The main components of the VGA controller hardware are the actual maze, object, goal, and character tiles, the virtual maze and object tiles, the maze and object descriptors, the various other signals, the virtual screen space, and the pixel drawing scheme. All of these are described in detail below.

Actual Tiles
All tiles in our game were chosen to be 32 by 32 pixels in size.

The goal tile (*GOAL_TILE*) is simply one of these 32 by 32 bit arrays, where all bits are 0 except for a cross-hair shaped section of 1's towards the center.

For the character tiles (*CHAR_TILES*), there are actually four of these 32 by 32 bit arrays. The first of these consists of a man-shaped section of 1's towards the center, and the next three progressively have the man moving his arms upwards. This is used for animating the man as he moves through the maze; there is a signal called *which_char_tile* that is settable by software and indicates which of these four tiles currently represents the character. Thus, as the software is moving the man's position, it can also set which tile to display, giving the appearance that the man is waving his arms.

The maze and object tiles (*MAZE_TILES* and *OBJ_TILES*) are less straight forward. In the interest of not wasting the space to store a 32 by 32 bit array for all possible types of wall combinations (left wall, right wall, top wall, top-left corner, top-right corner, bottom-left corner, bottom-right corner, left and right walls, top and bottom walls, top/left/bottom walls, left/bottom/right walls, bottom/right/top walls, and right/top/left walls) and key/door/color combinations (keys of all three colors and doors of all three colors and all four orientations), we chose only to store a minimal subset of all these tiles, and then store “virtual tile” descriptions that describe how to permute these basic tiles (via such things as rotations) in order to produce all possible tile combinations. Thus, we only physically stored four maze tiles and two object tiles.

The maze tiles stored are a left wall of 1's, a top-left corner of 1's, a left and right wall of 1's, and a left, right and bottom wall of 1's. The object tiles stored are a key-shaped section of 1's and a left wall of 1's.

Virtual Tiles
As mentioned, we thought it would be a waste to physically store all possible maze and object tiles, so we came up with this “virtual tile” scheme in order to describe all possible tiles from only a minimal subset of actual tiles.

The virtual maze tiles (*VIRTUAL_MAZE_TILES*) is an array of 15 8-bit bytes where each bit has been defined to mean the following:

Bits 7 - 6 indicate which tile from the *MAZE_TILES* array we are using
Bits 5 - 4 indicate a rotation on that tile (00 = none, 01 = 90 degrees clockwise, 10 = 180 degrees clockwise, 11 = 270 degrees clockwise)
[Bits 3 - 0 used to indicate, when the game logic was being developed in hardware, whether, if a player were on top of this tile, a wall was blocking his path in one of the four directions. However, as the game logic is now done in software, these bits are no longer used and just remain here as a vestigial feature.]

Thus with this scheme, a virtual maze tile description of “0101XXXX” indicates a top-right corner, since the first “01” picks out the second tile from *MAZE_TILES* (a top-left corner) and the second “01” rotates it by 90 degrees clockwise.

The virtual object tiles (*VIRTUAL_OBJ_TILES*) is an array of 16 9-bit strings where each bit has been defined to mean the following:

Bit 8 indicates which tile from the *OBJ_TILES* array we are using
Bits 7 - 6 indicate a rotation on that tile (00 = none, 01 = 90 degrees clockwise, 10 = 180 degrees clockwise, 11 = 270 degrees clockwise)
Bits 5 - 4 indicate which color the object is (00 = red, 01 = green, 10 = blue, 11 = invalid)
[Bits 3 - 0 again used to indicate whether a move in particular direction would be valid, but they are also no longer used]

Thus, with this scheme, a virtual object tile description of “01000XXXX” represents a right, red door, since the first “0” picks out the first tile from the OBJ_TILES array (the left door), the next “10” indicates a 180 degree clockwise rotation, and the next “00” indicates the color red.

By doing this, we can store a representation of all 15 maze tiles and all 16 object tiles, which would normally require about 3 MB of space, in just 7 KB of space.

Maze and Object Descriptors
The maze and object descriptors (*maze_desc* and *obj_desc*) are simply 2-D arrays of corresponding virtual tile numbers (indexes into the virtual tile arrays). A virtual tile number's position within the array indicates where on screen that particular tile goes relative to the others. For example, if the top left corner of the maze is at pixel (100, 100) and with our tile sizes of 32 pixels, if a virtual maze tile number resides at index (2,1) in the *maze_desc* array, that tile's top left corner begins at pixel (164, 132). We have chosen the maximum size of these arrays, and thus any maze, to be 16 by 16 tiles.

Since there are 15 virtual maze tiles and 16 virtual object tiles, the virtual tile numbers (indexes into the virtual tile arrays) need only be 4 bits. Thus, each entry in the descriptor arrays is 4 bits wide. Since the Avalon bus can read and write 16 bits at a time, we

assigned every group of 4 consecutive virtual tile numbers in these array one unique address, so that the Avalon bus can read from and write to four entries within the descriptor arrays at a time. As there are 2 * 16*16 / 4 of these addresses that we need in order for the software to be able to modify all maze and object tiles, the majority of the *GetData* process is used to get and set these values.

Other Signals
We also used some signals that are settable by software in order to correctly display the maze, character, objects, and window. These are the *maze_x_offset*, *maze_y_offset*, *cur_xpos_pix*, *cur_ypos_pix*, *end_xpos_pix*, *end_ypos_pix*, *half_window_size_x*, *half_window_size_y*, *which_keys*, and *which_char_tile*.

Virtual Screen Space
The way our game works is that the window is almost like a camera that follows the character. Thus, when the player "moves" it is actually the maze that is moving around the character. Thus, we encounter the problem of when we have a large maze, say one that is 16 by 16 tiles in size, if the character is all the way at one end of the maze, a good portion of the maze may actually be off of the physical screen. This is a major problem if this is done using the same VGA controller that was used for Lab 3. With that VGA controller, when an object's location occurs off screen or in the SYNC or BACK_PORCH area, the VGA controller essentially "freaks out" and multiple copies of the object may appear all over the screen and possibly in a distorted manner.

In order to correct this, we have introduced the notion of virtual screen space, which is essentially a change of the pixel coordinate system used by the VGA controller. The constant signals EXTRA_LEFT and EXTRA_TOP denote the extra amount of space in pixels we have added to the VGA screen at the left and top edges of the screen. Thus, we have defined pixel (0,0) to be above and to the left of the top left corner of the actual screen. Using these signals, all tile locations are described in this coordinate system, and thus even when off of the physical screen, they are always on our virtual screen and can be handled appropriately. If, however, a tile's location goes above or to the left of our defined borders, bad things may happen. However, with the constraints we have built into the game, this will never happen.

Pixel Drawing Scheme
The basic strategy in drawing pixels for our game is similar to that for the ball bouncing assignment of Lab 3; the VGA controller loops through all pixels on screen raster-style, and sets the color of each pixel depending on whether there is a tile at that location and whether the corresponding bit within that tile is a 0 or a 1. In our case, using all of the previously mentioned constructs (actual tiles, virtual tiles, extra screen space, etc), each pixel is set in the following way:

1. If the pixel is outside of the window, it is set to black.
2. If the pixel is on the window border, it is set to white.
3. If the pixel is within the window:
    a. If the pixel falls within the bounds of the character tile:

i. If the particular bit corresponding to that pixel within the tile is 1, it is set to yellow.
ii. If the particular bit corresponding to that pixel within the tile is 0:
    1. If the pixel falls within the bounds of an object tile:
        a. If the particular bit corresponding to that pixel within the tile is 1, it is set to the color of that object (red, blue, or green).
        b. If the particular bit corresponding to that pixel within the tile is 0:
            i. If the pixel falls within the bounds of the goal tile:
                1. If the particular bit corresponding to that pixel within the tile is 1, it is set to blue.
                2. If the particular bit corresponding to that pixel within the tile is 0:
                    a. If the pixel falls within the bounds of a maze tile:
                        i. If the particular bit corresponding to that pixel within the tile is 1, it is set to white.
                        ii. Otherwise it is set to black.

Thus, there is an implicit ordering in the way that tiles are displayed if multiple ones overlap. The character is always displayed on top, then the object tiles, then the goal tile, then the maze tile. So, if the character appears in the same location as a key and the goal, the character would be completely visible, the key would be obscured by the character, and the goal would be obscured by both the character and the key. This is appropriate for how we would like our game to look.

Hardware Design Conclusion

This ends the description of the underlying hardware. The software controls all other aspects of game logic, including loading the maze tiles to create a level, setting the positions of the goal, object, and character tiles, setting the size of the window, getting input from the user, animating the character, detecting when the character collects a key, removing the corresponding doors, and detecting when the goal location is reached. For more details on either the hardware or software, please see the comments within the actual code.

Section IIb. uCLinux

uCLinux

After some research, we decided to use uClinux to implement the WiiMote side of our project. uClinux is a stripped down version of a traditional linux kernel designed especially for embedded systems without an MMU. We found a version of uClinux tuned especially for nios2 processors and were able to compile it, load it onto the board and use it. uClinux has a lot of subsystems that can be enabled and disabled based on the project needs. To get the WiiMote working, we had to enable Host-side USB, ISP1362 HCD Support, Input Devices support, Bluetooth subsystem under Networking, The L2CAP, RFCOMM, and HIDP Protocols. It was especially important to enable input devices support first because this allowed the HIDP protocol to show up, which is the Bluetooth profile protocol across which the WiiMote communicates.

Hardware Integration

To compile uClinux for the right hardware specification, we used the system_0.ptf. Just as we had to use the system_0.ptf in the nios2 IDE to pick the CPU and memory element to load our program onto, uClinux has a compilation option called vendor_hwselect that must be called first and is tied to the hardware sof that gets loaded onto the board.

The full command is as follows:

(in the uClinux-dist directory)
make vendor_select SYSPTF=<aboslute path to system_0.ptf>
make

Bluetooth Libraries

A number of libraries had to be cross compiled for the nios2 processor using the nios-linux-uclibc-gcc set of compiler tools. This was the additional difficulty of compiling for a processor without an MMU so forking, daemons, and shared libraries were off-limits and had to be either commented out or changed. Cross compiling is more of an art than an exact science and many nights were spent trying to figure out esoteric compiler errors and we thank Dave for his tremendous help during the entire process.

There were a number of bluetooth libraries and dependencies we needed to get in order to make the WiiMote work. Bluez is the de-facto open source bluetooth stack and we used bluez-libs-2.25 and bluez-utils-2.25 for our project. bluez-libs-2.25 has the lower level core communication components like sending packets through RFCOMM and L2CAP, the connection protocol, and the service discovery protocol. Bluez-utils uses bluez-libs to provide user-level tools to test connectivity. For example 'hcitool scan' would scan for bluetooth devices that were on and in discoverable mode. Bluez-utils required expat-2.0.1 to compile. The exact commands to successfully cross compile the libraries as well as their dependencies can be found in the appendix.

Libwiimote

We were able to find a library called libwiimote that built on top of bluez-libs and provided us with a WiiMote API to directly connect to the WiiMote and send and receive WiiMote status update messages to and from the WiiMote. This allowed us to experiment with what the WiiMote could do and was the API we eventually used in our app to

control the WiiMote. Note that in order to cross compile the libwiimote correctly, the libwiimote.so shared library has to be removed from the makefiles.

Compiling applications for uClinux
Simple apps for uClinux can be compiled by using the nios2-linux-uclibc-gcc cross compiler and placed in the romfs/bin directory under uClinux-dist. The binary files have to be in an FLT format in order to run on uClinux and nios2-linux-uclibc-flthdr is a useful tool to check if a binary is in correct FLT form.

Because our wiimaze software requires direct access to hardware memory offsets as specified when the hardware was compiled, a few additional files were necessary to get these programs to cross compile correctly. First, we needed the io.h and alt_types.h under the altera7.2 directory. We also needed the system specific system.h under the syslib folder of the nios-ide program. In essence we tried to duplicate what the nios-ide compilation process did, but at a very stripped down level.

Installing and running
To load our project onto the board, we needed the hardware specification for the board in the form of an sof file and the software uClinux image compiled with the apps, usually named zImage. We used the nios2-configure-sof command to load the hardware onto the board, the nios2-download to load the software image, and nios2-terminal commands to start the terminal and the uClinux boot process.

After uClinux successfully completed, we used the hciconfig command to enable the bluetooth interface and the hcitool to scan for available WiiMotes. We then ran the wiimaze program with the correct address to connect to the WiiMote.

Section IIc. The WiiMote

Overview: The Task at Hand
The WiiMote has an on-board Broadcom BCM2042 Bluetooth chip that it uses to communicate with host devices. This communication is based on the Bluetooth HID protocol, which is the same one that is used for standard input devices such as mice and keyboards, along with L2CAP, a packet based communication protocol. However, unlike these devices, the WiiMote does not make use of the standard data types or HID descriptors and instead only communicates its report length to the connected host device. Thus, while connection establishment is similar, if not identical, to the way other peripherals do it, communicating particular information, such as key button presses, accelerometer data and speaker commands, is unique to the WiiMote.

For our intents and purposes, we are interested in three main components: button presses, accelerometer data, and speaker controls. Thus, we have to be able to transmit, receive and process the HID descriptors and data that define these three features. The following is a breakdown of the data reports that are sent from the WiiMote to the host device and vice versa:

Report Type	Communication Direction	Format	Per Byte Information			
Button / Accelerometer Data	WiiMote to Host	(a1) 31 BB BB XX YY ZZ	a1: HID command (data input)	31: Report ID (Core Buttons and Accelerometer	BB BB: 2 bytes of button data	XX,YY,ZZ: 1 byte for force in each direction
Enable Speaker	Host to Wiimote	(52) 14 04	52: HID command (data output)	14: Report ID (Speaker Enable)	04: Bit Mask to enable speaker	n/a
Unmute Speaker	Host to Wiimote	(52) 14 00	52: HID command (data output)	14: Report ID (Speaker Enable)	00: Bit Mask to unmute speaker	n/a
Speaker Data	Host to Wiimote	(52) 18 LL DD DD DD DD DD DD DD DD DD DD DD DD DD DD DD DD DD	52: HID command (data output)	18: Report ID (Speaker Data)	LL: Length, in bytes, of trailing data	DD: Up to 20 unsigned bytes representing data

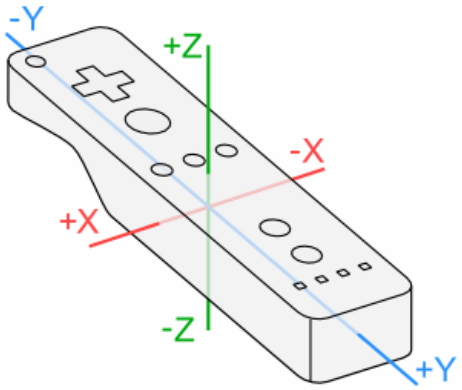
Note: Each space delimited double is representative of a hexadecimal byte

Each WiiMote report is sent in the form of a data packet using Bluetooth's L2CAP protocol, which is similar to UDP in the sense that these packets are sent in a best effort fashion (more on this later).

Libwiimote: The Solution

Rather than deal directly with these details, we made use of a simple and lightweight C based library called libwiimote, which allowed for a clean interface with which to communicate with the WiiMote device. Written on top of the BlueZ tools and libraries already included in our uCLinux image, it provides us with access to the necessary components of the WiiMote, including the key buttons, the accelerometer, and the speaker.

Additionally, this library abstracts away much of the details of writing to the speaker, as well as translating accelerometer data. For example, the WiiMote uses a three axis, linear accelerometer (Analog Device IC ADXL330) that reports the force being exerted in any particular axis. These axes can be seen in the diagram below:



The library translates this raw XX YY ZZ force data into 3 tilt variables of the `wiimote_t` data type it uses to represent a WiiMote.

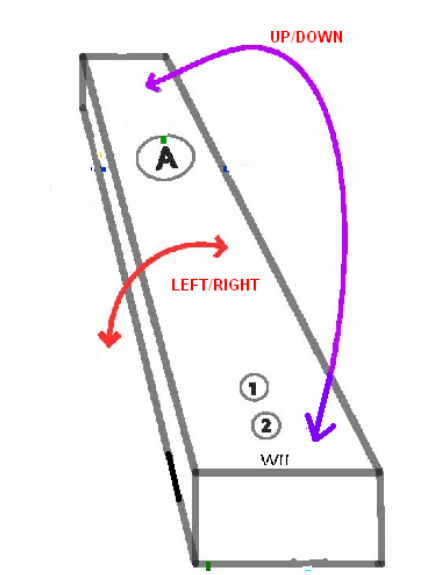
Using libwiimote in the WiiMaze:

Using the library to programmatically read and write the necessary data to the WiiMote involves these steps:

1. Declaring and initializing a `wiimote_t` data structure

2. Calling the `wiimote_connect` function on this data type, using the 6-byte Bluetooth device address of the WiiMote (acquired through the BlueZ hciscan tool)
3. Initializing the speaker via the `wiimote_speaker_init` function
4. Making various function calls to get and set data from the WiiMote:
    a. `wiimote.keys.a`: checks if "A" was pressed. Enables accelerometer
    b. `wiimote.keys.home`: checks if "Home" was pressed. Disconnects WiiMote
    c. `wiimote.tilt.x`/`wiimote.tilt.y`/`wiimote.tilt.z`:reads accelerometer data and returns a value in degrees (normalized over 180^{o}). Used to move character
    d. `wiimote_speaker_freq`/`wiimote_speaker_volume`/`wiimote_speaker_play`: sets the frequency of the speaker, sets the volume of the speaker, plays a given 20 byte tone, respectively. Used to play event tones
    e. `wiimote_update`: used to synchronize the state of the wiimote_t structure and the physical WiiMote
    f. `wiimote_get_state`: used to get the next state of the WiiMote.
5. Continually make calls to `wiimote_is_open`, `wiimote_update`, `wiimote_get_state` to make sure the Wiimote is still connected and to synchronize software and hardware states of the WiiMote
6. Close connection to the WiiMote by a call to `wiimote_disconnect`

The first three steps are merely set up tasks and are done upon loading the game. The WiiMaze game, upon being executed at the uCLinux command line with the appropriate WiiMote address argument, blocks (steps 1 and 2) until communication is established with the specified device. Then the speaker is initialized (step 3) and the game is loaded. Pressing the "A" button enables the accelerometer and the game begins. From then on, the game responds to the user's inputs via the tilting of the remote in the x and y positions, as seen below:



Readings are sampled every time the user passes a direction threshold that has been set and calibrated to about +/- 50 degrees in the x direction (LEFT/RIGHT in the diagram) and +/- 30 degrees in the y direction (UP/DOWN in the diagram). This causes checks for boundary conditions in the maze and, either registers a valid move and animates in that direction, or registers an object (wall, key or end of maze) and plays a sound. During each action, calls to `wiimote_update` and `wiimote_get_state` are made to make the game as responsive as possible. If a user presses the HOME key at any time, the WiiMote will drop the Bluetooth connection and the game will end.

Caveats/Issues

A key feature necessary for the success of this game is the real time responsiveness of the WiiMote actions: if a user tilts the device upwards, we want the on screen sprite to move in that direction as soon as possible, with as little jitter as possible. This is directly dependent on how fast we can receive and process accelerometer data coming in from the WiiMote. However, what we did not count on was that there would be a characteristic of the underlying Bluetooth protocol that would make this a bit more difficult that initially imagined.

During the course of our game, we have to make numerous calls to the `wiimote_update` function in order to guarantee that the physical state of the WiiMote is in sync with that of our programmatic representation of it. This means that, for every read or write that is made, we have to force an update that causes a certain amount of delay. When it comes to reading accelerometer data, this delay increases due to the sheer volume of data that is coming in. For each slight change in the WiiMote's position, a packet is sent from the device to the host and each packet is processed and acknowledged by the host only when a call to `wiimote_update` is made. In between calls to this function, the transport protocol puts each unacknowledged packet onto a queue. This is because the connection that the host uCLinux device establishes with the WiiMote is built on top of the L2CAP (Logical Link Control and Adaptation Protocol) transport protocol. Like UDP, it executes packet delivery in a best effort fashion. However, it adds a particular guarantee that affects our actions: in order delivery of packets. Thus, whenever we want to get a tilt reading from the WiiMote, we have to flush the queue to get to the particular move that was made at the instant we queried the device.

Since there is no functionality in the libwiimote library for flushing this queue, we had to use a couple of alternatives. First, we made liberal use of the `wiimote_update` function when game logic and animation were taking place. This gave us a chance to blindly process some packets and removing them from the queue, allowing us to process the current move only when we wanted to. However, this did not give us the exact behavior we desired because of the fact that calls to this function come at a processing cost, which caused a noticeable jitter in the animation. This processing involves syncing each and every component, and thus variable, of the WiiMote so that the device and host would have a similar view of the WiiMote state. As a second attempt, we stripped a level off the libwiimote library and made use of one of its underlying functions: `wiimote_get_state`. Rather than processing the packets that are received in between calls to the WiiMote, this function merely gets the next state of the device and does nothing to it. This essentially works to help clear the buffer, while also avoiding the overhead of syncing.

This same type of synchronization problem arises when it comes to using the WiiMote speaker as well. Playing a tone through the device involves sending it a constant stream of bytes, using the speaker data packets (described in the earlier table). The problem with this is that the delay caused by the processing overhead for each packet causes the speed at which the device receives the data to vary greatly. Thus, playing a constant melody breaks down to just an output of dispersed tones. Using the `wiimote_get_state` function method described above to alleviate the problem proved fruitless. Thus, we only include tones for events, such as wall hits, key pickups and level wins.

Section III. Roles and Reflections

VGA Controller Hardware: Shaun and Brian
Game Logic Software: Shaun, Mack, and Brian
Cross-Compiling uCLinux: Mack
WiiMote Interactions: Brian and Mack

Reflections -- By Shaun Salzberg
A major regret in the design of the hardware was making it way too custom tailored for our particular WiiMaze game. For example, we had specific, unique constructs for all of the different tile types (goal, character, maze, and objects). Doing it like this made it very difficult to change the game later on when we wanted to make modifications to the game rules or add in new features to the game. For example, it might have been nice to add special power-up to the game that one could collect to increase the size of the viewing window. Doing this, however, would have been a major pain. What we should have done instead was create a very generic tile "object" to represent all graphics on screen including the maze, objects, goal, and character. The software could then "instantiate" as many of these as needed and associate with each not only a bitmap but also a position on screen and perhaps one or multiple colors. Doing this would have made adding in a new feature such as a power-up as easy as a simple software modification.

Reflections – By Mack Lu
Doing this design project really made me appreciate my computer more and I developed a much better understanding of what it takes to build a functioning computing system that does something useful. I learned a lot about the software/hardware interface and how to use the nios2 board as a hardware platform to build software. I was also very fortunate to have very good team members who worked hard and who each contributed their own share in moving the project forward. I think the fact that we talked about our ideas, designed how the components were going to work together and modularly divided the project up into 3 parts before we even started any coding was very beneficial towards making progress. It kept us all on the same page and allowed us to come up with individual milestones that we could hit. The advice I have for future teams would be to definitely start early and also to have weekly goals or objectives that you try to reach so that you can be constantly making progress. These small victories strung together over the course of many weeks will really help drive the project towards completion.

Reflections – By Brian Ramos

I found 4840 to be one of the most intense, and at the same time, most rewarding, classes I have taken while at Columbia. At first, it seemed a bit intimidating because of the fact that I had come into it with almost no experience in logic design or VHDL. However, the three introductory labs helped create a solid foundation upon which to build from throughout the semester. Although these did not teach me everything I thought I had to know for the actual implementation of our project, they definitely put me on the right path. Be that as it may, I still think that taking an advanced logic design PRIOR to this class is invaluable. I tried taking it in tandem and realized that I would definitely would have been better off knowing much of the things I know now prior to starting the project.

On the flip side of things, I think that doing this project without all the background knowledge actually helped make this an even greater accomplishment. I feel that our team did a great job utilizing the resources we were given and we were able to band together to produce a well functioning product. Along with good communication in terms of work distribution among the team members, keeping in constant contact with the TA and/or professor during the whole of the project design and implementation phases was definitely very helpful. It allowed us to catch bugs early on and also find solutions to problems that we could have spent weeks trying to figure out on our own.

All in all, I would suggest the following for future teams:

1. Make sure to take advanced logic design first! It helps with overall understanding, as well as with debugging issues.
2. Stay in constant communication with your team and course staff (professor and TA). They are always there to help.
3. Put effort into planning and organizing the work to be done. Take advantage of the project proposal to plan your milestones properly.
4. Use source control. Trying to pass files around is never effective and can lead to a lot of mistakes. SVN is your friend.

Section IV. Files Included

wii_vga.vhd

This is the VGA Controller hardware.

DE2_NET.qpf

This is the DE2_NET project file.

DE2_NET.sof

This is the compiled hardware ready to load onto the board.

zImage

This is the completely compiled uCLinux kernel ready to load onto the board.

wiimaze.c

This is the software containing the game logic and WiiMote interactions.

Section V. Appendix

Wiimaze Compilation Readme
Steps:

===uClinux===
1. Get uClinux (follow instructions from
http://nioswiki.jot.com/WikiHome/OperatingSystems/UClinuxDist)
2. Setup (make menuconfig) uClinux with the following:
    SCSI, SCSI write
    USB, USB file system, Philips 1362 controller
    Input Device drivers
    Bluetooth - L2CAP, SCO, HIDP protocols
3. configure
4. make vendor_hwselect SYSPTF=<path to ptf>
5. make

In order to load uClinux onto the board,

===HARDWARE===
DE2_NET.sof - hardware pin placements that gets sent to the board
system_0.ptf - system configuration file uClinux needs

update hardware:
make vendor_hwselect
SYSPTF=/home/mack/code/2008/embedded/uc_hardware/system_0.ptf
make

In order to load the software onto the hardware, we need to use the compiled sof file.

===BLUETOOTH LIBRARIES===
Dependencies:
bluez-utils: needs expat and bluez-libs
libwiimote: bluez-libs
programs: bluez-libs and libwiimote

expat-2.0.1
./configure --host=nios2-linux-uclibc --enable-static --disable-shared LDFLAGS=-elf2flt
not, sure, didn't compile it but bluez-utils needs this

bluez-libs-2.25
in nios-configure.sh:
./configure --host=nios2-linux-uclibc --enable-static --disable-shared --with-gnu-ld
LDFLAGS=-elf2flt --prefix=<absolute path to install directory>
make

bluez-utils-2.25

in nios-configure.sh:
./configure --host=nios2-linux-uclibc --disable-shared --enable-static LDFLAGS="-L<bluez-libs lib directory> -elf2flt" CPPFLAGS="-I<bluez-libs include directory> -DUCLINUX" --disable-dbus --disable-pcmcia --disable-bluepin --disable-obex --disable-alsa --disable-obex --disable-dfutool --prefix=<absolute path to install directory>

libwiimote
autoconf (generates configure)
in nios-configure.sh:
./configure --host=nios2-linux-uclibc --disable-shared --enable-static LDFLAGS="-L<bluez-libs lib directory> -elf2flt -lbluetooth" CFLAGS="-I<bluez-libs include directory> -DUCLINUX"

Modify src/Makefile:
comment out libcwiimote.so in all: libcwiimote.a libcwiimote.so

Modify the main Makefile:
commenting out the $(INSTALL) -m 755 %(LIBDIR)/libcwiimote.so line

This needs to be done because uClinux can't handle shared libraries (no MMU)

make
test/make (creates the examples)

===APPS===
From /opt/e4840/altera7.2/
io.h
alt_types.h

From syslib folder in nios2-ide:
system.h

wiimaze.sh (cross compiles wiimaze.c)
nios2-linux-uclibc-gcc -Os -Wall -pipe -D_ENABLE_TILT -D_ENABLE_FORCE
-I<absolute path to nios2 headers> (io.h, alt_types.h, system.h path)
-I<bluez-libs include directory> (bluetooth headers path)
-I<aboslute path to>/uc_libwiimote/src (libwiimote headers path)
-DSYSTEM_BUS_WIDTH=32
-DUCLINUX
wiimaze.c
-o wiimaze
-L<absolute path to bluez-libs>/lib (lbluetooth.a library path)
-L<absolute path to>/uc_libwiimote/lib (lwiimote.a library path)
-lcwiimote -lbluetooth -lm (wiimote, bluetooth, and math libraries)
-elf2flt (convert to flt for uClinux)

nios2-linux-uclibc-flthdr wiimaze #checks if the binary is a valid flt file

Move software to uClinux-dist/romfs/bin. This will cause the software to be compiled into the uClinux image when uClinux is compiled.
Compile uClinux again.

===INSTALLING===
Needs:
DE2_NET.sof - hardware configuration
zImage - compiled uClinux image with apps

nios2-configure-sof DE2_NET.sof
nios2-download -g zImage
nios2-terminal (uClinux should boot)

===RUNNING===
>hciconfig
Displays bluetooth devices connected to the board. If USB controller is up, USB driver is corrected configured and Bluetooth subsystem is correctly installed, this should show hci0 when you plug in a USB Bluetooth Dongle.

>hciconfig hci0 up
Brings the dongle online.

>hcitool scan
Shows the Bluetooth address of devices in range

>wiimaze <BT address>
Runs game with wiimote's bt address given by BT address

wiimaze.c

```
#include <io.h>
#include <system.h>
#include <stdio.h>
#include <stdlib.h>

#include "wiimote.h"
#include "wiimote_api.h"

// vga settings in hardware
#define HPAD 96+48
#define VPAD 2+33
#define EXTRA_LEFT 800
#define EXTRA_TOP 525
#define SCREEN_LEFT EXTRA_LEFT+HPAD
#define SCREEN_TOP EXTRA_TOP+VPAD
#define SCREEN_CENTER_X SCREEN_LEFT+320
#define SCREEN_CENTER_Y SCREEN_TOP+240

// game constants
#define TILE_SIZE 32
#define MAX_MAZE_SIZE_X 16  // in # of tiles
#define MAX_MAZE_SIZE_Y 16  // in # of tiles
```

```
// settings
#define PAUSE_AMT 40000000     // in # of for loops
#define SPEED_PIX 2         // should be 1, 2, 4, or 8

// memory loc offsets
#define MAZE_X_OFFSET 0             // in pixels
#define MAZE_Y_OFFSET 1             // in pixels
#define CHAR_X_OFFSET 2             // in pixels
#define CHAR_Y_OFFSET 3             // in pixels
#define GOAL_X_OFFSET 4             // in maze tiles
#define GOAL_Y_OFFSET 5             // in maze tiles
#define WINDOW_SIZE_OFFSET 6        // in maze tiles
#define WHICH_KEYS_OFFSET 7         // see key defs
#define MAZE_TILES_OFFSET 8         // see maze tile types
#define OBJ_TILES_OFFSET 73         // see obj tile types
#define WHICH_CHAR_TILE_OFFSET 137  // which char tile to draw



// due to poor foresight, it will be very annoying to add more keys now
// so just leave these as is unless we really want more keys
#define KEY1 1
#define KEY2 2
#define KEY3 4

// maze tile types
#define LEFT_WALL 0
#define TOP_WALL 1
#define RIGHT_WALL 2
#define BOTTOM_WALL 3
#define TOP_LEFT_CORNER 4
#define TOP_RIGHT_CORNER 5
#define BOTTOM_RIGHT_CORNER 6
#define BOTTOM_LEFT_CORNER  7
#define VERTICAL_HALL 8
#define HORIZONTAL_HALL 9
#define BOTTOM_DEADEND 10
#define LEFT_DEADEND 11
#define TOP_DEADEND 12
#define RIGHT_DEADEND 13
#define NO_MAZE_TILE 14

// obj tile types
#define LEFT_WALL_RED 0
#define TOP_WALL_RED 1
#define RIGHT_WALL_RED 2
#define BOTTOM_WALL_RED 3
#define LEFT_WALL_GREEN 4
#define TOP_WALL_GREEN 5
#define RIGHT_WALL_GREEN 6
#define BOTTOM_WALL_GREEN 7
#define LEFT_WALL_BLUE 8
#define TOP_WALL_BLUE 9
#define RIGHT_WALL_BLUE 10
#define BOTTOM_WALL_BLUE 11
#define KEY_RED 12
#define KEY_GREEN 13
#define KEY_BLUE 14
#define NO_OBJ_TILE 15

// level descriptor offsets
#define DESC_BASIC_OFFSET 0
#define DESC_MAZE_OFFSET 1
#define DESC_OBJ_OFFSET MAX_MAZE_SIZE_Y+1

// basic level descriptor offsets
#define DESC_BASIC_XPOS_MAZE_OFFSET 0
#define DESC_BASIC_YPOS_MAZE_OFFSET 1
#define DESC_BASIC_XPOS_CHAR_OFFSET 2
#define DESC_BASIC_YPOS_CHAR_OFFSET 3
```

```
#define DESC_BASIC_XPOS_GOAL_OFFSET 4
#define DESC_BASIC_YPOS_GOAL_OFFSET 5
#define DESC_BASIC_HALF_XWIN_OFFSET 6
#define DESC_BASIC_HALF_YWIN_OFFSET 7

// character movements directions
#define UP 0
#define DOWN 1
#define LEFT 2
#define RIGHT 3

// Number of levels
#define NUM_LEVELS 4

// function declarations
void move_maze_to(int x, int y);
int get_maze_x();
int get_maze_y();

void move_char_to_pix(int x, int y);
void move_char_to_cell(int col, int row);
int get_char_start_x(int **the_level);
int get_char_start_y(int **the_level);

void set_goal_at(int col, int row);

void hide_keys_and_doors(int which);
void show_keys_and_doors(int which);

void set_window_size(int half_cols, int half_rows);

void animate(int dir, wiimote_t *wiimote);
void toggle_char_tile();

void get_level(int **the_level, int level);
void get_level1(int **the_level);
void get_level2(int **the_level);
void get_level3(int **the_level);
void get_level4(int **the_level);
// add more get_levelX fns here and have them be called by get_level

void create_level(int **level_desc);

void set4maze_tiles(int offset, int tile1, int tile2, int tile3, int tile4);
void set4obj_tiles(int offset, int tile1, int tile2, int tile3, int tile4);

int can_move(int **the_level, int fromx, int fromy, int dir);
int contains_wall(int tile_type, int dir);
int contains_door(int tile_type, int dir);

int is_at_goal_loc(int **the_level, int x, int y);
int ontop_of_which_key(int **the_level, int x, int y);

int get_next_move(wiimote_t *wiimote);

// wiimote sound fns
void make_got_to_goal_sound(wiimote_t *wiimote);
void make_got_key_sound(wiimote_t *wiimote);
void make_moving_sound();
void make_hit_wall_sound(wiimote_t *wiimote);

void pause();
void animate_pause(wiimote_t *wiimote);


/* sounds */
uint8_t sample[20] = {
       0x3c,0x3c,0x3c,0x3c,0x3c,0x3c,0x3c,0x3c,0x3c,0x3c,
       0x3c,0x3c,0x3c,0x3c,0x3c,0x3c,0x3c,0x3c,0x3c,0x3c
};
```

```
int main(int argc, char **argv)
{
    int i;
    int count=0;

    wiimote_t wiimote = WIIMOTE_INIT;

    /* check command line args */
    if(argc<2){
      printf("You didn't provide a remote address.\nUse \'hcitool sca\'n to get the
address of your wiimote\n");
      exit(1);
    }

    /* connect the wiimote */
    if(wiimote_connect(&wiimote, argv[1])<0){
      printf("Unable to connect to wiimote: %s\n",wiimote_get_error());
      exit(1);
    }

    printf("Connection established!\n");
    wiimote.led.one = 1;
    wiimote_speaker_init(&wiimote, WIIMOTE_FMT_S4, 0xff);

    // set up level descriptor
    int **my_level = NULL;
    my_level = (int **)malloc((MAX_MAZE_SIZE_Y * 2 + 1)*sizeof(int *));
    my_level[DESC_BASIC_OFFSET] = (int *)malloc(8*sizeof(int));
    for( i = DESC_MAZE_OFFSET; i < DESC_MAZE_OFFSET+MAX_MAZE_SIZE_Y; i++ )
        my_level[i] = (int *)malloc(MAX_MAZE_SIZE_X * sizeof(int));
    for( i = DESC_OBJ_OFFSET; i < DESC_OBJ_OFFSET+MAX_MAZE_SIZE_Y; i++ )
        my_level[i] = (int *)malloc(MAX_MAZE_SIZE_X * sizeof(int));

    // game logic
    int char_x, char_y, move_dir, which_key, cur_level = 1;


    while(1) {
        get_level(my_level,cur_level);
        create_level(my_level);

        show_keys_and_doors(KEY1);
        show_keys_and_doors(KEY2);
        show_keys_and_doors(KEY3);

        char_x = get_char_start_x(my_level);
        char_y = get_char_start_y(my_level);

        while(1) {
          fprintf(stderr, "iteration: %d:\n", count);
          count++;
          move_dir = get_next_move(&wiimote);
          if( can_move(my_level,char_x,char_y,move_dir) ) {
            // Used to animate. Helps clear input queue of incoming packets via
wiimote_update
            animate(move_dir, &wiimote);

            if( move_dir == UP ) char_y--;
            else if( move_dir == DOWN ) char_y++;
            else if( move_dir == LEFT ) char_x--;
            else if( move_dir == RIGHT ) char_x++;

            which_key = ontop_of_which_key(my_level,char_x,char_y);
            if( which_key != -1 ) {
              hide_keys_and_doors(which_key);
              make_got_key_sound(&wiimote);
            }

            if( is_at_goal_loc( my_level, char_x, char_y ) ) {
              make_got_to_goal_sound(&wiimote);
              cur_level = cur_level+1;
```

```
            if(cur_level > NUM_LEVELS)
              cur_level = 1;
            break;
          }
        }
        else
          make_hit_wall_sound(&wiimote);
      }
    }

}

void move_maze_to(int x, int y) {
  IOWR(VGA_BASE,MAZE_X_OFFSET,x);
  IOWR(VGA_BASE,MAZE_Y_OFFSET,y);
}

int get_maze_x() {
  return IORD(VGA_BASE,MAZE_X_OFFSET);
}

int get_maze_y() {
  return IORD(VGA_BASE,MAZE_Y_OFFSET);
}

void move_char_to_pix(int x, int y) {
    IOWR(VGA_BASE,CHAR_X_OFFSET,x);
    IOWR(VGA_BASE,CHAR_Y_OFFSET,y);
}

// be sure to set maze where you want it first
void move_char_to_cell(int col, int row) {
    IOWR(VGA_BASE,CHAR_X_OFFSET,get_maze_x()+col*TILE_SIZE);
    IOWR(VGA_BASE,CHAR_Y_OFFSET,get_maze_y()+row*TILE_SIZE);
}

void set_goal_at(int col, int row) {
    IOWR(VGA_BASE,GOAL_X_OFFSET,col);
    IOWR(VGA_BASE,GOAL_Y_OFFSET,row);
}

void hide_keys_and_doors(int which) {
    IOWR(VGA_BASE,WHICH_KEYS_OFFSET,IORD(VGA_BASE,WHICH_KEYS_OFFSET)&(~which));
}

void show_keys_and_doors(int which) {
    IOWR(VGA_BASE,WHICH_KEYS_OFFSET,IORD(VGA_BASE,WHICH_KEYS_OFFSET)|which);
}

int has_key(int which) {
    return !(IORD(VGA_BASE,WHICH_KEYS_OFFSET)&which);
}

// pass to this function HALF of the size of the window you want in number of tiles
// i.e. if you want a 4x6 window, pass 2 and 3
// these should be at most 8 bit numbers
void set_window_size(int half_cols, int half_rows) {
    IOWR(VGA_BASE,WINDOW_SIZE_OFFSET,(half_cols<<8)|half_rows);
}

void set4maze_tiles(int offset, int tile1, int tile2, int tile3, int tile4) {
    IOWR(VGA_BASE,MAZE_TILES_OFFSET+offset,(tile1<<12)|(tile2<<8)|(tile3<<4)|(tile4));
}

void set4obj_tiles(int offset, int tile1, int tile2, int tile3, int tile4) {
    IOWR(VGA_BASE,OBJ_TILES_OFFSET+offset,(tile1<<12)|(tile2<<8)|(tile3<<4)|(tile4));
}

void create_level(int **level_desc) {
    // set basic info
```

```
set_window_size(level_desc[DESC_BASIC_OFFSET][DESC_BASIC_HALF_XWIN_OFFSET],level_desc[DES
C_BASIC_OFFSET][DESC_BASIC_HALF_YWIN_OFFSET]);

move_maze_to(level_desc[DESC_BASIC_OFFSET][DESC_BASIC_XPOS_MAZE_OFFSET],level_desc[DESC_B
ASIC_OFFSET][DESC_BASIC_YPOS_MAZE_OFFSET]); pause(); pause(); pause(); pause(); // pauses
are stupid hacks

move_char_to_cell(level_desc[DESC_BASIC_OFFSET][DESC_BASIC_XPOS_CHAR_OFFSET],level_desc[D
ESC_BASIC_OFFSET][DESC_BASIC_YPOS_CHAR_OFFSET]);

set_goal_at(level_desc[DESC_BASIC_OFFSET][DESC_BASIC_XPOS_GOAL_OFFSET],level_desc[DESC_BA
SIC_OFFSET][DESC_BASIC_YPOS_GOAL_OFFSET]); pause();

    // set maze and obj tiles
    int i,j, off;
    for(i = 0, off = 0; i < MAX_MAZE_SIZE_Y; i++ ) {
        for( j = 0; j < MAX_MAZE_SIZE_X; j+=4, off++ ) {
          set4maze_tiles(off,
level_desc[DESC_MAZE_OFFSET+i][j],level_desc[DESC_MAZE_OFFSET+i][j+1],level_desc[DESC_MAZ
E_OFFSET+i][j+2],level_desc[DESC_MAZE_OFFSET+i][j+3]);
            set4obj_tiles(off,
level_desc[DESC_OBJ_OFFSET+i][j],level_desc[DESC_OBJ_OFFSET+i][j+1],level_desc[DESC_OBJ_O
FFSET+i][j+2],level_desc[DESC_OBJ_OFFSET+i][j+3]);
        }
    }
}

void get_level(int **the_level, int level)
{
    if( level == 1 ) get_level1(the_level);
    else if( level == 2 ) get_level2(the_level);
    else if( level == 3 ) get_level3(the_level);
    else if( level == 4 ) get_level4(the_level);

}


void get_level1(int **the_level)
{
    int i,j;

    // basic level info
    the_level[DESC_BASIC_OFFSET][DESC_BASIC_XPOS_MAZE_OFFSET] = SCREEN_CENTER_X-
TILE_SIZE;    // x pos of top left corner of maze in pixels to start
    the_level[DESC_BASIC_OFFSET][DESC_BASIC_YPOS_MAZE_OFFSET] = SCREEN_CENTER_Y-
TILE_SIZE;    // y coordinate of top left corner of maze in pixels to start
    the_level[DESC_BASIC_OFFSET][DESC_BASIC_XPOS_CHAR_OFFSET] = 0;
// x coordinate of starting position of character within the maze
    the_level[DESC_BASIC_OFFSET][DESC_BASIC_YPOS_CHAR_OFFSET] = 0;
// y coordinate of starting position of character within the maze
    the_level[DESC_BASIC_OFFSET][DESC_BASIC_XPOS_GOAL_OFFSET] = 1;
// x coordinate of goal position within the maze
    the_level[DESC_BASIC_OFFSET][DESC_BASIC_YPOS_GOAL_OFFSET] = 1;
// y coordinate of goal position within the maze
    the_level[DESC_BASIC_OFFSET][DESC_BASIC_HALF_XWIN_OFFSET] = 3;
// half window size in x direction
    the_level[DESC_BASIC_OFFSET][DESC_BASIC_HALF_YWIN_OFFSET] = 3;
// half window size in y direction

    // maze tile info
    // first row
    the_level[DESC_MAZE_OFFSET+0][0] = TOP_LEFT_CORNER;
    the_level[DESC_MAZE_OFFSET+0][1] = TOP_RIGHT_CORNER;
    for(i = 2; i < MAX_MAZE_SIZE_X; i++ ) the_level[DESC_MAZE_OFFSET+0][i] =
NO_MAZE_TILE;

    // second row
    the_level[DESC_MAZE_OFFSET+1][0] = BOTTOM_LEFT_CORNER;
    the_level[DESC_MAZE_OFFSET+1][1] = BOTTOM_RIGHT_CORNER;
```

```
    for(i = 2; i < MAX_MAZE_SIZE_X; i++ ) the_level[DESC_MAZE_OFFSET+1][i] =
NO_MAZE_TILE;

    // rest of the rows
    for(i = DESC_MAZE_OFFSET+2; i <= MAX_MAZE_SIZE_Y; i++ ) {
        for( j = 0; j < MAX_MAZE_SIZE_X; j++ )
            the_level[i][j] = NO_MAZE_TILE;
    }

    // obj tile info
    // set everything to empty first
    for(i = DESC_OBJ_OFFSET; i < DESC_OBJ_OFFSET+MAX_MAZE_SIZE_Y; i++ ) {
        for( j = 0; j < MAX_MAZE_SIZE_X; j++ )
            the_level[i][j] = NO_OBJ_TILE;
    }

}


void get_level2(int **the_level)
{
    int i,j;

    // basic level info
    the_level[DESC_BASIC_OFFSET][DESC_BASIC_XPOS_MAZE_OFFSET] = SCREEN_CENTER_X-
3*TILE_SIZE;    // x pos of top left corner of maze in pixels to start
    the_level[DESC_BASIC_OFFSET][DESC_BASIC_YPOS_MAZE_OFFSET] = SCREEN_CENTER_Y-
TILE_SIZE;    // y coordinate of top left corner of maze in pixels to start
    the_level[DESC_BASIC_OFFSET][DESC_BASIC_XPOS_CHAR_OFFSET] = 2;
// x coordinate of starting position of character within the maze
    the_level[DESC_BASIC_OFFSET][DESC_BASIC_YPOS_CHAR_OFFSET] = 0;
// y coordinate of starting position of character within the maze
    the_level[DESC_BASIC_OFFSET][DESC_BASIC_XPOS_GOAL_OFFSET] = 0;
// x coordinate of goal position within the maze
    the_level[DESC_BASIC_OFFSET][DESC_BASIC_YPOS_GOAL_OFFSET] = 0;
// y coordinate of goal position within the maze
    the_level[DESC_BASIC_OFFSET][DESC_BASIC_HALF_XWIN_OFFSET] = 6;
// half window size in x direction
    the_level[DESC_BASIC_OFFSET][DESC_BASIC_HALF_YWIN_OFFSET] = 2;
// half window size in y direction

    // maze tile info
    // first row
    the_level[DESC_MAZE_OFFSET+0][0] = LEFT_DEADEND;
    the_level[DESC_MAZE_OFFSET+0][1] = HORIZONTAL_HALL;
    the_level[DESC_MAZE_OFFSET+0][2] = HORIZONTAL_HALL;
    the_level[DESC_MAZE_OFFSET+0][3] = HORIZONTAL_HALL;
    the_level[DESC_MAZE_OFFSET+0][4] = HORIZONTAL_HALL;
    the_level[DESC_MAZE_OFFSET+0][5] = RIGHT_DEADEND;
    for(i = 6; i < MAX_MAZE_SIZE_X; i++ ) the_level[DESC_MAZE_OFFSET+0][i] =
NO_MAZE_TILE;

    // rest of the rows
    for(i = DESC_MAZE_OFFSET+1; i <= MAX_MAZE_SIZE_Y; i++ ) {
        for( j = 0; j < MAX_MAZE_SIZE_X; j++ )
            the_level[i][j] = NO_MAZE_TILE;
    }

    // obj tile info
    // set everything to empty first
    for(i = DESC_OBJ_OFFSET; i < DESC_OBJ_OFFSET+MAX_MAZE_SIZE_Y; i++ ) {
        for( j = 0; j < MAX_MAZE_SIZE_X; j++ )
            the_level[i][j] = NO_OBJ_TILE;
    }

    // then set where the objects are
    the_level[DESC_OBJ_OFFSET+0][0] = RIGHT_WALL_GREEN;
    the_level[DESC_OBJ_OFFSET+0][1] = KEY_BLUE;
    the_level[DESC_OBJ_OFFSET+0][2] = LEFT_WALL_RED;
    the_level[DESC_OBJ_OFFSET+0][3] = KEY_RED;
    the_level[DESC_OBJ_OFFSET+0][4] = RIGHT_WALL_BLUE;
```

```
    the_level[DESC_OBJ_OFFSET+0][5] = KEY_GREEN;

}


void get_level3(int **the_level)
{
    int i,j;

    // basic level info
    the_level[DESC_BASIC_OFFSET][DESC_BASIC_XPOS_MAZE_OFFSET] = SCREEN_CENTER_X-
2*TILE_SIZE;    // x pos of top left corner of maze in pixels to start
    the_level[DESC_BASIC_OFFSET][DESC_BASIC_YPOS_MAZE_OFFSET] = SCREEN_CENTER_Y-
2*TILE_SIZE;    // y coordinate of top left corner of maze in pixels to start
    the_level[DESC_BASIC_OFFSET][DESC_BASIC_XPOS_CHAR_OFFSET] = 2;
// x coordinate of starting position of character within the maze
    the_level[DESC_BASIC_OFFSET][DESC_BASIC_YPOS_CHAR_OFFSET] = 2;
// y coordinate of starting position of character within the maze
    the_level[DESC_BASIC_OFFSET][DESC_BASIC_XPOS_GOAL_OFFSET] = 4;
// x coordinate of goal position within the maze
    the_level[DESC_BASIC_OFFSET][DESC_BASIC_YPOS_GOAL_OFFSET] = 2;
// y coordinate of goal position within the maze
    the_level[DESC_BASIC_OFFSET][DESC_BASIC_HALF_XWIN_OFFSET] = 2;
// half window size in x direction
    the_level[DESC_BASIC_OFFSET][DESC_BASIC_HALF_YWIN_OFFSET] = 2;
// half window size in y direction

    // maze tile info
    // first row
    the_level[DESC_MAZE_OFFSET+0][0] = TOP_DEADEND;
    the_level[DESC_MAZE_OFFSET+0][1] = TOP_LEFT_CORNER;
    the_level[DESC_MAZE_OFFSET+0][2] = HORIZONTAL_HALL;
    the_level[DESC_MAZE_OFFSET+0][3] = HORIZONTAL_HALL;
    the_level[DESC_MAZE_OFFSET+0][4] = TOP_RIGHT_CORNER;
    for(i = 5; i < MAX_MAZE_SIZE_X; i++ ) the_level[DESC_MAZE_OFFSET+0][i] =
NO_MAZE_TILE;

    // second row
    the_level[DESC_MAZE_OFFSET+1][0] = VERTICAL_HALL;
    the_level[DESC_MAZE_OFFSET+1][1] = LEFT_WALL;
    the_level[DESC_MAZE_OFFSET+1][2] = TOP_WALL;
    the_level[DESC_MAZE_OFFSET+1][3] = TOP_RIGHT_CORNER;
    the_level[DESC_MAZE_OFFSET+1][4] = VERTICAL_HALL;
    for(i = 5; i < MAX_MAZE_SIZE_X; i++ ) the_level[DESC_MAZE_OFFSET+1][i] =
NO_MAZE_TILE;

     // third row
    the_level[DESC_MAZE_OFFSET+2][0] = VERTICAL_HALL;
    the_level[DESC_MAZE_OFFSET+2][1] = BOTTOM_DEADEND;
    the_level[DESC_MAZE_OFFSET+2][2] = TOP_DEADEND;
    the_level[DESC_MAZE_OFFSET+2][3] = RIGHT_WALL;
    the_level[DESC_MAZE_OFFSET+2][4] = BOTTOM_DEADEND;
    for(i = 5; i < MAX_MAZE_SIZE_X; i++ ) the_level[DESC_MAZE_OFFSET+2][i] =
NO_MAZE_TILE;

    // fourth row;
    the_level[DESC_MAZE_OFFSET+3][0] = BOTTOM_LEFT_CORNER;
    the_level[DESC_MAZE_OFFSET+3][1] = BOTTOM_WALL;
    the_level[DESC_MAZE_OFFSET+3][2] = RIGHT_WALL;
    the_level[DESC_MAZE_OFFSET+3][3] = NO_MAZE_TILE;
    the_level[DESC_MAZE_OFFSET+3][4] = RIGHT_WALL;
    for(i = 5; i < MAX_MAZE_SIZE_X; i++ ) the_level[DESC_MAZE_OFFSET+3][i] =
NO_MAZE_TILE;

    // fifth row
    the_level[DESC_MAZE_OFFSET+4][0] = LEFT_DEADEND;
    the_level[DESC_MAZE_OFFSET+4][1] = HORIZONTAL_HALL;
    the_level[DESC_MAZE_OFFSET+4][2] = BOTTOM_WALL;
    the_level[DESC_MAZE_OFFSET+4][3] = BOTTOM_WALL;
    the_level[DESC_MAZE_OFFSET+4][4] = BOTTOM_DEADEND;
```

```
    for(i = 5; i < MAX_MAZE_SIZE_X; i++ ) the_level[DESC_MAZE_OFFSET+4][i] =
NO_MAZE_TILE;

    // rest of the rows
    for(i = DESC_MAZE_OFFSET+5; i <= MAX_MAZE_SIZE_Y; i++ ) {
        for( j = 0; j < MAX_MAZE_SIZE_X; j++ )
            the_level[i][j] = NO_MAZE_TILE;
    }

    // obj tile info
    // set everything to empty first
    for(i = DESC_OBJ_OFFSET; i < DESC_OBJ_OFFSET+MAX_MAZE_SIZE_Y; i++ ) {
        for( j = 0; j < MAX_MAZE_SIZE_X; j++ )
            the_level[i][j] = NO_OBJ_TILE;
    }

    // then set where the objects are
    the_level[DESC_OBJ_OFFSET+0][0] = KEY_RED;
    the_level[DESC_OBJ_OFFSET+0][4] = BOTTOM_WALL_BLUE;
    the_level[DESC_OBJ_OFFSET+1][1] = BOTTOM_WALL_GREEN;
    the_level[DESC_OBJ_OFFSET+2][1] = KEY_BLUE;
    the_level[DESC_OBJ_OFFSET+3][4] = BOTTOM_WALL_RED;
    the_level[DESC_OBJ_OFFSET+4][4] = KEY_GREEN;

}



void get_level4(int **the_level)
{
    int i,j;

    // basic level info
    the_level[DESC_BASIC_OFFSET][DESC_BASIC_XPOS_MAZE_OFFSET] = SCREEN_CENTER_X-
8*TILE_SIZE;    // x pos of top left corner of maze in pixels to start
    the_level[DESC_BASIC_OFFSET][DESC_BASIC_YPOS_MAZE_OFFSET] = SCREEN_CENTER_Y-
8*TILE_SIZE;    // y coordinate of top left corner of maze in pixels to start
    the_level[DESC_BASIC_OFFSET][DESC_BASIC_XPOS_CHAR_OFFSET] = 7;
// x coordinate of starting position of character within the maze
    the_level[DESC_BASIC_OFFSET][DESC_BASIC_YPOS_CHAR_OFFSET] = 7;
// y coordinate of starting position of character within the maze
    the_level[DESC_BASIC_OFFSET][DESC_BASIC_XPOS_GOAL_OFFSET] = 0;
// x coordinate of goal position within the maze
    the_level[DESC_BASIC_OFFSET][DESC_BASIC_YPOS_GOAL_OFFSET] = 15;
// y coordinate of goal position within the maze
    the_level[DESC_BASIC_OFFSET][DESC_BASIC_HALF_XWIN_OFFSET] = 4;
// half window size in x direction
    the_level[DESC_BASIC_OFFSET][DESC_BASIC_HALF_YWIN_OFFSET] = 4;
// half window size in y direction

    // maze tile info
    // first row
    the_level[DESC_MAZE_OFFSET+0][0] = TOP_DEADEND;
    the_level[DESC_MAZE_OFFSET+0][1] = TOP_LEFT_CORNER;
    the_level[DESC_MAZE_OFFSET+0][2] = TOP_RIGHT_CORNER;
    the_level[DESC_MAZE_OFFSET+0][3] = TOP_LEFT_CORNER;
    the_level[DESC_MAZE_OFFSET+0][4] = HORIZONTAL_HALL;
    the_level[DESC_MAZE_OFFSET+0][5] = HORIZONTAL_HALL;
    the_level[DESC_MAZE_OFFSET+0][6] = TOP_RIGHT_CORNER;
    the_level[DESC_MAZE_OFFSET+0][7] = TOP_LEFT_CORNER;
    the_level[DESC_MAZE_OFFSET+0][8] = HORIZONTAL_HALL;
    the_level[DESC_MAZE_OFFSET+0][9] = HORIZONTAL_HALL;
    the_level[DESC_MAZE_OFFSET+0][10] = TOP_WALL;
    the_level[DESC_MAZE_OFFSET+0][11] = HORIZONTAL_HALL;
    the_level[DESC_MAZE_OFFSET+0][12] = HORIZONTAL_HALL;
    the_level[DESC_MAZE_OFFSET+0][13] = HORIZONTAL_HALL;
    the_level[DESC_MAZE_OFFSET+0][14] = RIGHT_DEADEND;
    the_level[DESC_MAZE_OFFSET+0][15] = TOP_DEADEND;

    // second row
```

```
the_level[DESC_MAZE_OFFSET+1][0] = VERTICAL_HALL;
the_level[DESC_MAZE_OFFSET+1][1] = VERTICAL_HALL;
the_level[DESC_MAZE_OFFSET+1][2] = VERTICAL_HALL;
the_level[DESC_MAZE_OFFSET+1][3] = VERTICAL_HALL;
the_level[DESC_MAZE_OFFSET+1][4] = LEFT_DEADEND;
the_level[DESC_MAZE_OFFSET+1][5] = HORIZONTAL_HALL;
the_level[DESC_MAZE_OFFSET+1][6] = BOTTOM_RIGHT_CORNER;
the_level[DESC_MAZE_OFFSET+1][7] = VERTICAL_HALL;
the_level[DESC_MAZE_OFFSET+1][8] = LEFT_DEADEND;
the_level[DESC_MAZE_OFFSET+1][9] = HORIZONTAL_HALL;
the_level[DESC_MAZE_OFFSET+1][10] = RIGHT_WALL;
the_level[DESC_MAZE_OFFSET+1][11] = TOP_LEFT_CORNER;
the_level[DESC_MAZE_OFFSET+1][12] = TOP_WALL;
the_level[DESC_MAZE_OFFSET+1][13] = HORIZONTAL_HALL;
the_level[DESC_MAZE_OFFSET+1][14] = RIGHT_DEADEND;
the_level[DESC_MAZE_OFFSET+1][15] = VERTICAL_HALL;

// third row
the_level[DESC_MAZE_OFFSET+2][0] = VERTICAL_HALL;
the_level[DESC_MAZE_OFFSET+2][1] = VERTICAL_HALL;
the_level[DESC_MAZE_OFFSET+2][2] = VERTICAL_HALL;
the_level[DESC_MAZE_OFFSET+2][3] = BOTTOM_LEFT_CORNER;
the_level[DESC_MAZE_OFFSET+2][4] = HORIZONTAL_HALL;
the_level[DESC_MAZE_OFFSET+2][5] = HORIZONTAL_HALL;
the_level[DESC_MAZE_OFFSET+2][6] = HORIZONTAL_HALL;
the_level[DESC_MAZE_OFFSET+2][7] = NO_MAZE_TILE;
the_level[DESC_MAZE_OFFSET+2][8] = HORIZONTAL_HALL;
the_level[DESC_MAZE_OFFSET+2][9] = TOP_RIGHT_CORNER;
the_level[DESC_MAZE_OFFSET+2][10] = VERTICAL_HALL;
the_level[DESC_MAZE_OFFSET+2][11] = VERTICAL_HALL;
the_level[DESC_MAZE_OFFSET+2][12] = VERTICAL_HALL;
the_level[DESC_MAZE_OFFSET+2][13] = TOP_DEADEND;
the_level[DESC_MAZE_OFFSET+2][14] = TOP_DEADEND;
the_level[DESC_MAZE_OFFSET+2][15] = VERTICAL_HALL;

// fourth row
the_level[DESC_MAZE_OFFSET+3][0] = VERTICAL_HALL;
the_level[DESC_MAZE_OFFSET+3][1] = VERTICAL_HALL;
the_level[DESC_MAZE_OFFSET+3][2] = LEFT_WALL;
the_level[DESC_MAZE_OFFSET+3][3] = HORIZONTAL_HALL;
the_level[DESC_MAZE_OFFSET+3][4] = HORIZONTAL_HALL;
the_level[DESC_MAZE_OFFSET+3][5] = TOP_WALL;
the_level[DESC_MAZE_OFFSET+3][6] = TOP_RIGHT_CORNER;
the_level[DESC_MAZE_OFFSET+3][7] = VERTICAL_HALL;
the_level[DESC_MAZE_OFFSET+3][8] = TOP_DEADEND;
the_level[DESC_MAZE_OFFSET+3][9] = VERTICAL_HALL;
the_level[DESC_MAZE_OFFSET+3][10] = VERTICAL_HALL;
the_level[DESC_MAZE_OFFSET+3][11] = VERTICAL_HALL;
the_level[DESC_MAZE_OFFSET+3][12] = LEFT_WALL;
the_level[DESC_MAZE_OFFSET+3][13] = BOTTOM_WALL;
the_level[DESC_MAZE_OFFSET+3][14] = BOTTOM_RIGHT_CORNER;
the_level[DESC_MAZE_OFFSET+3][15] = VERTICAL_HALL;

// fifth row
the_level[DESC_MAZE_OFFSET+4][0] = VERTICAL_HALL;
the_level[DESC_MAZE_OFFSET+4][1] = BOTTOM_DEADEND;
the_level[DESC_MAZE_OFFSET+4][2] = VERTICAL_HALL;
the_level[DESC_MAZE_OFFSET+4][3] = LEFT_DEADEND;
the_level[DESC_MAZE_OFFSET+4][4] = HORIZONTAL_HALL;
the_level[DESC_MAZE_OFFSET+4][5] = BOTTOM_RIGHT_CORNER;
the_level[DESC_MAZE_OFFSET+4][6] = VERTICAL_HALL;
the_level[DESC_MAZE_OFFSET+4][7] = BOTTOM_DEADEND;
the_level[DESC_MAZE_OFFSET+4][8] = VERTICAL_HALL;
the_level[DESC_MAZE_OFFSET+4][9] = BOTTOM_LEFT_CORNER;
the_level[DESC_MAZE_OFFSET+4][10] = RIGHT_WALL;
the_level[DESC_MAZE_OFFSET+4][11] = BOTTOM_DEADEND;
the_level[DESC_MAZE_OFFSET+4][12] = VERTICAL_HALL;
the_level[DESC_MAZE_OFFSET+4][13] = LEFT_DEADEND;
the_level[DESC_MAZE_OFFSET+4][14] = TOP_WALL;
the_level[DESC_MAZE_OFFSET+4][15] = BOTTOM_RIGHT_CORNER;
```

```
// sixth row
the_level[DESC_MAZE_OFFSET+5][0] = BOTTOM_LEFT_CORNER;
the_level[DESC_MAZE_OFFSET+5][1] = HORIZONTAL_HALL;
the_level[DESC_MAZE_OFFSET+5][2] = BOTTOM_WALL;
the_level[DESC_MAZE_OFFSET+5][3] = HORIZONTAL_HALL;
the_level[DESC_MAZE_OFFSET+5][4] = HORIZONTAL_HALL;
the_level[DESC_MAZE_OFFSET+5][5] = TOP_WALL;
the_level[DESC_MAZE_OFFSET+5][6] = BOTTOM_WALL;
the_level[DESC_MAZE_OFFSET+5][7] = HORIZONTAL_HALL;
the_level[DESC_MAZE_OFFSET+5][8] = NO_MAZE_TILE;
the_level[DESC_MAZE_OFFSET+5][9] = TOP_RIGHT_CORNER;
the_level[DESC_MAZE_OFFSET+5][10] = LEFT_WALL;
the_level[DESC_MAZE_OFFSET+5][11] = HORIZONTAL_HALL;
the_level[DESC_MAZE_OFFSET+5][12] = NO_MAZE_TILE;
the_level[DESC_MAZE_OFFSET+5][13] = RIGHT_DEADEND;
the_level[DESC_MAZE_OFFSET+5][14] = VERTICAL_HALL;
the_level[DESC_MAZE_OFFSET+5][15] = TOP_DEADEND;

// seventh row
the_level[DESC_MAZE_OFFSET+6][0] = LEFT_DEADEND;
the_level[DESC_MAZE_OFFSET+6][1] = TOP_WALL;
the_level[DESC_MAZE_OFFSET+6][2] = HORIZONTAL_HALL;
the_level[DESC_MAZE_OFFSET+6][3] = HORIZONTAL_HALL;
the_level[DESC_MAZE_OFFSET+6][4] = RIGHT_DEADEND;
the_level[DESC_MAZE_OFFSET+6][5] = VERTICAL_HALL;
the_level[DESC_MAZE_OFFSET+6][6] = TOP_LEFT_CORNER;
the_level[DESC_MAZE_OFFSET+6][7] = RIGHT_DEADEND;
the_level[DESC_MAZE_OFFSET+6][8] = VERTICAL_HALL;
the_level[DESC_MAZE_OFFSET+6][9] = BOTTOM_DEADEND;
the_level[DESC_MAZE_OFFSET+6][10] = VERTICAL_HALL;
the_level[DESC_MAZE_OFFSET+6][11] = TOP_DEADEND;
the_level[DESC_MAZE_OFFSET+6][12] = LEFT_WALL;
the_level[DESC_MAZE_OFFSET+6][13] = HORIZONTAL_HALL;
the_level[DESC_MAZE_OFFSET+6][14] = RIGHT_WALL;
the_level[DESC_MAZE_OFFSET+6][15] = VERTICAL_HALL;

// eighth row
the_level[DESC_MAZE_OFFSET+7][0] = TOP_DEADEND;
the_level[DESC_MAZE_OFFSET+7][1] = VERTICAL_HALL;
the_level[DESC_MAZE_OFFSET+7][2] = TOP_LEFT_CORNER;
the_level[DESC_MAZE_OFFSET+7][3] = TOP_RIGHT_CORNER;
the_level[DESC_MAZE_OFFSET+7][4] = TOP_LEFT_CORNER;
the_level[DESC_MAZE_OFFSET+7][5] = BOTTOM_RIGHT_CORNER;
the_level[DESC_MAZE_OFFSET+7][6] = VERTICAL_HALL;
the_level[DESC_MAZE_OFFSET+7][7] = LEFT_DEADEND;
the_level[DESC_MAZE_OFFSET+7][8] = NO_MAZE_TILE;
the_level[DESC_MAZE_OFFSET+7][9] = HORIZONTAL_HALL;
the_level[DESC_MAZE_OFFSET+7][10] = BOTTOM_RIGHT_CORNER;
the_level[DESC_MAZE_OFFSET+7][11] = VERTICAL_HALL;
the_level[DESC_MAZE_OFFSET+7][12] = VERTICAL_HALL;
the_level[DESC_MAZE_OFFSET+7][13] = TOP_DEADEND;
the_level[DESC_MAZE_OFFSET+7][14] = BOTTOM_LEFT_CORNER;
the_level[DESC_MAZE_OFFSET+7][15] = RIGHT_WALL;

// ninth row
the_level[DESC_MAZE_OFFSET+8][0] = VERTICAL_HALL;
the_level[DESC_MAZE_OFFSET+8][1] = VERTICAL_HALL;
the_level[DESC_MAZE_OFFSET+8][2] = LEFT_WALL;
the_level[DESC_MAZE_OFFSET+8][3] = RIGHT_WALL;
the_level[DESC_MAZE_OFFSET+8][4] = VERTICAL_HALL;
the_level[DESC_MAZE_OFFSET+8][5] = TOP_LEFT_CORNER;
the_level[DESC_MAZE_OFFSET+8][6] = NO_MAZE_TILE;
the_level[DESC_MAZE_OFFSET+8][7] = TOP_RIGHT_CORNER;
the_level[DESC_MAZE_OFFSET+8][8] = BOTTOM_LEFT_CORNER;
the_level[DESC_MAZE_OFFSET+8][9] = TOP_WALL;
the_level[DESC_MAZE_OFFSET+8][10] = HORIZONTAL_HALL;
the_level[DESC_MAZE_OFFSET+8][11] = BOTTOM_RIGHT_CORNER;
the_level[DESC_MAZE_OFFSET+8][12] = VERTICAL_HALL;
the_level[DESC_MAZE_OFFSET+8][13] = LEFT_WALL;
the_level[DESC_MAZE_OFFSET+8][14] = TOP_RIGHT_CORNER;
the_level[DESC_MAZE_OFFSET+8][15] = VERTICAL_HALL;
```

```
// tenth row
the_level[DESC_MAZE_OFFSET+9][0] = VERTICAL_HALL;
the_level[DESC_MAZE_OFFSET+9][1] = VERTICAL_HALL;
the_level[DESC_MAZE_OFFSET+9][2] = BOTTOM_DEADEND;
the_level[DESC_MAZE_OFFSET+9][3] = VERTICAL_HALL;
the_level[DESC_MAZE_OFFSET+9][4] = VERTICAL_HALL;
the_level[DESC_MAZE_OFFSET+9][5] = VERTICAL_HALL;
the_level[DESC_MAZE_OFFSET+9][6] = BOTTOM_DEADEND;
the_level[DESC_MAZE_OFFSET+9][7] = BOTTOM_LEFT_CORNER;
the_level[DESC_MAZE_OFFSET+9][8] = RIGHT_DEADEND;
the_level[DESC_MAZE_OFFSET+9][9] = VERTICAL_HALL;
the_level[DESC_MAZE_OFFSET+9][10] = LEFT_DEADEND;
the_level[DESC_MAZE_OFFSET+9][11] = HORIZONTAL_HALL;
the_level[DESC_MAZE_OFFSET+9][12] = BOTTOM_RIGHT_CORNER;
the_level[DESC_MAZE_OFFSET+9][13] = VERTICAL_HALL;
the_level[DESC_MAZE_OFFSET+9][14] = BOTTOM_DEADEND;
the_level[DESC_MAZE_OFFSET+9][15] = VERTICAL_HALL;

// eleventh row
the_level[DESC_MAZE_OFFSET+10][0] = VERTICAL_HALL;
the_level[DESC_MAZE_OFFSET+10][1] = LEFT_WALL;
the_level[DESC_MAZE_OFFSET+10][2] = HORIZONTAL_HALL;
the_level[DESC_MAZE_OFFSET+10][3] = BOTTOM_RIGHT_CORNER;
the_level[DESC_MAZE_OFFSET+10][4] = VERTICAL_HALL;
the_level[DESC_MAZE_OFFSET+10][5] = LEFT_WALL;
the_level[DESC_MAZE_OFFSET+10][6] = TOP_WALL;
the_level[DESC_MAZE_OFFSET+10][7] = TOP_WALL;
the_level[DESC_MAZE_OFFSET+10][8] = TOP_WALL;
the_level[DESC_MAZE_OFFSET+10][9] = BOTTOM_WALL;
the_level[DESC_MAZE_OFFSET+10][10] = HORIZONTAL_HALL;
the_level[DESC_MAZE_OFFSET+10][11] = TOP_WALL;
the_level[DESC_MAZE_OFFSET+10][12] = TOP_RIGHT_CORNER;
the_level[DESC_MAZE_OFFSET+10][13] = LEFT_WALL;
the_level[DESC_MAZE_OFFSET+10][14] = HORIZONTAL_HALL;
the_level[DESC_MAZE_OFFSET+10][15] = BOTTOM_RIGHT_CORNER;

// twelfth row
the_level[DESC_MAZE_OFFSET+11][0] = BOTTOM_LEFT_CORNER;
the_level[DESC_MAZE_OFFSET+11][1] = BOTTOM_WALL;
the_level[DESC_MAZE_OFFSET+11][2] = TOP_WALL;
the_level[DESC_MAZE_OFFSET+11][3] = HORIZONTAL_HALL;
the_level[DESC_MAZE_OFFSET+11][4] = BOTTOM_RIGHT_CORNER;
the_level[DESC_MAZE_OFFSET+11][5] = VERTICAL_HALL;
the_level[DESC_MAZE_OFFSET+11][6] = VERTICAL_HALL;
the_level[DESC_MAZE_OFFSET+11][7] = VERTICAL_HALL;
the_level[DESC_MAZE_OFFSET+11][8] = VERTICAL_HALL;
the_level[DESC_MAZE_OFFSET+11][9] = TOP_LEFT_CORNER;
the_level[DESC_MAZE_OFFSET+11][10] = RIGHT_DEADEND;
the_level[DESC_MAZE_OFFSET+11][11] = VERTICAL_HALL;
the_level[DESC_MAZE_OFFSET+11][12] = VERTICAL_HALL;
the_level[DESC_MAZE_OFFSET+11][13] = VERTICAL_HALL;
the_level[DESC_MAZE_OFFSET+11][14] = LEFT_DEADEND;
the_level[DESC_MAZE_OFFSET+11][15] = TOP_RIGHT_CORNER;

// thirteenth row
the_level[DESC_MAZE_OFFSET+12][0] = TOP_LEFT_CORNER;
the_level[DESC_MAZE_OFFSET+12][1] = HORIZONTAL_HALL;
the_level[DESC_MAZE_OFFSET+12][2] = BOTTOM_RIGHT_CORNER;
the_level[DESC_MAZE_OFFSET+12][3] = TOP_LEFT_CORNER;
the_level[DESC_MAZE_OFFSET+12][4] = HORIZONTAL_HALL;
the_level[DESC_MAZE_OFFSET+12][5] = BOTTOM_RIGHT_CORNER;
the_level[DESC_MAZE_OFFSET+12][6] = VERTICAL_HALL;
the_level[DESC_MAZE_OFFSET+12][7] = VERTICAL_HALL;
the_level[DESC_MAZE_OFFSET+12][8] = VERTICAL_HALL;
the_level[DESC_MAZE_OFFSET+12][9] = VERTICAL_HALL;
the_level[DESC_MAZE_OFFSET+12][10] = TOP_DEADEND;
the_level[DESC_MAZE_OFFSET+12][11] = BOTTOM_DEADEND;
the_level[DESC_MAZE_OFFSET+12][12] = BOTTOM_DEADEND;
the_level[DESC_MAZE_OFFSET+12][13] = BOTTOM_LEFT_CORNER;
the_level[DESC_MAZE_OFFSET+12][14] = HORIZONTAL_HALL;
```

```
the_level[DESC_MAZE_OFFSET+12][15] = RIGHT_WALL;

// fourteenth row
the_level[DESC_MAZE_OFFSET+13][0] = BOTTOM_LEFT_CORNER;
the_level[DESC_MAZE_OFFSET+13][1] = HORIZONTAL_HALL;
the_level[DESC_MAZE_OFFSET+13][2] = TOP_RIGHT_CORNER;
the_level[DESC_MAZE_OFFSET+13][3] = VERTICAL_HALL;
the_level[DESC_MAZE_OFFSET+13][4] = TOP_LEFT_CORNER;
the_level[DESC_MAZE_OFFSET+13][5] = HORIZONTAL_HALL;
the_level[DESC_MAZE_OFFSET+13][6] = BOTTOM_RIGHT_CORNER;
the_level[DESC_MAZE_OFFSET+13][7] = VERTICAL_HALL;
the_level[DESC_MAZE_OFFSET+13][8] = VERTICAL_HALL;
the_level[DESC_MAZE_OFFSET+13][9] = VERTICAL_HALL;
the_level[DESC_MAZE_OFFSET+13][10] = LEFT_WALL;
the_level[DESC_MAZE_OFFSET+13][11] = TOP_WALL;
the_level[DESC_MAZE_OFFSET+13][12] = HORIZONTAL_HALL;
the_level[DESC_MAZE_OFFSET+13][13] = TOP_WALL;
the_level[DESC_MAZE_OFFSET+13][14] = TOP_RIGHT_CORNER;
the_level[DESC_MAZE_OFFSET+13][15] = BOTTOM_DEADEND;

// fifteenth row
the_level[DESC_MAZE_OFFSET+14][0] = TOP_LEFT_CORNER;
the_level[DESC_MAZE_OFFSET+14][1] = HORIZONTAL_HALL;
the_level[DESC_MAZE_OFFSET+14][2] = BOTTOM_RIGHT_CORNER;
the_level[DESC_MAZE_OFFSET+14][3] = VERTICAL_HALL;
the_level[DESC_MAZE_OFFSET+14][4] = BOTTOM_LEFT_CORNER;
the_level[DESC_MAZE_OFFSET+14][5] = HORIZONTAL_HALL;
the_level[DESC_MAZE_OFFSET+14][6] = RIGHT_DEADEND;
the_level[DESC_MAZE_OFFSET+14][7] = VERTICAL_HALL;
the_level[DESC_MAZE_OFFSET+14][8] = BOTTOM_LEFT_CORNER;
the_level[DESC_MAZE_OFFSET+14][9] = BOTTOM_WALL;
the_level[DESC_MAZE_OFFSET+14][10] = BOTTOM_RIGHT_CORNER;
the_level[DESC_MAZE_OFFSET+14][11] = BOTTOM_DEADEND;
the_level[DESC_MAZE_OFFSET+14][12] = TOP_DEADEND;
the_level[DESC_MAZE_OFFSET+14][13] = VERTICAL_HALL;
the_level[DESC_MAZE_OFFSET+14][14] = BOTTOM_LEFT_CORNER;
the_level[DESC_MAZE_OFFSET+14][15] = RIGHT_DEADEND;

// sixeenth row
the_level[DESC_MAZE_OFFSET+15][0] = BOTTOM_DEADEND;
the_level[DESC_MAZE_OFFSET+15][1] = LEFT_DEADEND;
the_level[DESC_MAZE_OFFSET+15][2] = HORIZONTAL_HALL;
the_level[DESC_MAZE_OFFSET+15][3] = BOTTOM_WALL;
the_level[DESC_MAZE_OFFSET+15][4] = HORIZONTAL_HALL;
the_level[DESC_MAZE_OFFSET+15][5] = HORIZONTAL_HALL;
the_level[DESC_MAZE_OFFSET+15][6] = HORIZONTAL_HALL;
the_level[DESC_MAZE_OFFSET+15][7] = BOTTOM_WALL;
the_level[DESC_MAZE_OFFSET+15][8] = HORIZONTAL_HALL;
the_level[DESC_MAZE_OFFSET+15][9] = HORIZONTAL_HALL;
the_level[DESC_MAZE_OFFSET+15][10] = HORIZONTAL_HALL;
the_level[DESC_MAZE_OFFSET+15][11] = HORIZONTAL_HALL;
the_level[DESC_MAZE_OFFSET+15][12] = BOTTOM_RIGHT_CORNER;
the_level[DESC_MAZE_OFFSET+15][13] = BOTTOM_LEFT_CORNER;
the_level[DESC_MAZE_OFFSET+15][14] = HORIZONTAL_HALL;
the_level[DESC_MAZE_OFFSET+15][15] = RIGHT_DEADEND;

// obj tile info
// set everything to empty first
for(i = DESC_OBJ_OFFSET; i < DESC_OBJ_OFFSET+MAX_MAZE_SIZE_Y; i++ ) {
    for( j = 0; j < MAX_MAZE_SIZE_X; j++ )
        the_level[i][j] = NO_OBJ_TILE;
}

// then set where the objects are
the_level[DESC_OBJ_OFFSET+0][0] = KEY_RED;
the_level[DESC_OBJ_OFFSET+0][15] = KEY_BLUE;
the_level[DESC_OBJ_OFFSET+6][13] = LEFT_WALL_RED;
the_level[DESC_OBJ_OFFSET+11][8] = TOP_WALL_BLUE;
the_level[DESC_OBJ_OFFSET+11][2] = BOTTOM_WALL_GREEN;
the_level[DESC_OBJ_OFFSET+15][15] = KEY_GREEN;
```

```
}


// assumes fromx and fromy are valid
int can_move(int **the_level, int fromx, int fromy, int dir) {
    int maze_self, maze_up, maze_down, maze_left, maze_right;
    int obj_self, obj_up, obj_down, obj_left, obj_right;

    maze_self = the_level[DESC_MAZE_OFFSET+fromy][fromx];
    obj_self = the_level[DESC_OBJ_OFFSET+fromy][fromx];

    switch( dir ) {
        case UP:
            // check maze tile ok
            if( fromy == 0 ) maze_up = -1;
            else maze_up = the_level[DESC_MAZE_OFFSET+fromy-1][fromx];

            if( maze_up == -1 || contains_wall(maze_self,UP) ||
contains_wall(maze_up,DOWN) ) return 0;

            // check obj tile ok
            if( fromy == 0 ) obj_up = -1;
            else obj_up = the_level[DESC_OBJ_OFFSET+fromy-1][fromx];

            if( obj_up == -1 || contains_door(obj_self,UP) ||
contains_door(obj_up,DOWN) ) return 0;

            return 1;
        case DOWN:
            // check maze tile ok
            if( fromy == MAX_MAZE_SIZE_Y-1 ) maze_down = -1;
            else maze_down = the_level[DESC_MAZE_OFFSET+fromy+1][fromx];

            if( maze_down == -1 || contains_wall(maze_self,DOWN) ||
contains_wall(maze_down,UP) ) return 0;

            // check obj tile ok
            if( fromy == MAX_MAZE_SIZE_Y-1 ) obj_down = -1;
            else obj_down = the_level[DESC_OBJ_OFFSET+fromy+1][fromx];

            if( obj_down == -1 || contains_door(obj_self,DOWN) ||
contains_door(obj_down,UP) ) return 0;

            return 1;
        case LEFT:
            // check maze tiles ok
            if( fromx == 0 ) maze_left = -1;
            else maze_left = the_level[DESC_MAZE_OFFSET+fromy][fromx-1];

            if( maze_left == -1 || contains_wall(maze_self,LEFT) ||
contains_wall(maze_left,RIGHT) ) return 0;

            // check obj tiles ok
            if( fromx == 0 ) obj_left = -1;
            else obj_left = the_level[DESC_OBJ_OFFSET+fromy][fromx-1];

            if( obj_left == -1 || contains_door(obj_self,LEFT) ||
contains_door(obj_left,RIGHT) ) return 0;

            return 1;
        case RIGHT:
            // check maze tiles ok
            if( fromx == MAX_MAZE_SIZE_X-1 ) maze_right = -1;
            else maze_right = the_level[DESC_MAZE_OFFSET+fromy][fromx+1];

            if( maze_right == -1 || contains_wall(maze_self,RIGHT) ||
contains_wall(maze_right,LEFT) ) return 0;

            // check obj tiles ok
            if( fromx == MAX_MAZE_SIZE_X-1 ) obj_right = -1;
            else obj_right = the_level[DESC_OBJ_OFFSET+fromy][fromx+1];
```

```
            if( obj_right == -1 || contains_door(obj_self,RIGHT) ||
contains_door(obj_right,LEFT) ) return 0;

            return 1;
    }
    return 0;
}

int contains_wall(int tile_type, int dir) {
    if( dir == UP )
        return (tile_type == TOP_WALL || tile_type == TOP_LEFT_CORNER || tile_type ==
TOP_RIGHT_CORNER || tile_type == HORIZONTAL_HALL || tile_type == LEFT_DEADEND ||
tile_type == TOP_DEADEND || tile_type == RIGHT_DEADEND );
    else if( dir == DOWN )
        return (tile_type == BOTTOM_WALL || tile_type == BOTTOM_LEFT_CORNER || tile_type
== BOTTOM_RIGHT_CORNER || tile_type == HORIZONTAL_HALL || tile_type == LEFT_DEADEND ||
tile_type == RIGHT_DEADEND || tile_type == BOTTOM_DEADEND );
    else if( dir == LEFT )
        return (tile_type == LEFT_WALL || tile_type == BOTTOM_LEFT_CORNER || tile_type ==
TOP_LEFT_CORNER || tile_type == VERTICAL_HALL || tile_type == TOP_DEADEND || tile_type ==
LEFT_DEADEND || tile_type == BOTTOM_DEADEND );
    else if( dir == RIGHT )
        return (tile_type == RIGHT_WALL || tile_type == BOTTOM_RIGHT_CORNER || tile_type
== TOP_RIGHT_CORNER || tile_type == VERTICAL_HALL || tile_type == TOP_DEADEND ||
tile_type == RIGHT_DEADEND || tile_type == BOTTOM_DEADEND );

    return 0;
}

int contains_door(int tile_type, int dir) {
    if( dir == UP )
        return ( (tile_type == TOP_WALL_RED && !has_key(KEY1)) || (tile_type ==
TOP_WALL_GREEN && !has_key(KEY2)) || (tile_type == TOP_WALL_BLUE && !has_key(KEY3)) );
    else if( dir == DOWN )
        return ( (tile_type == BOTTOM_WALL_RED && !has_key(KEY1)) || (tile_type ==
BOTTOM_WALL_GREEN && !has_key(KEY2)) || (tile_type == BOTTOM_WALL_BLUE
&& !has_key(KEY3)) );
    else if( dir == LEFT )
        return ( (tile_type == LEFT_WALL_RED && !has_key(KEY1)) || (tile_type ==
LEFT_WALL_GREEN && !has_key(KEY2)) || (tile_type == LEFT_WALL_BLUE && !has_key(KEY3)));
    else if( dir == RIGHT )
        return ( (tile_type == RIGHT_WALL_RED && !has_key(KEY1)) || (tile_type ==
RIGHT_WALL_GREEN && !has_key(KEY2)) || (tile_type == RIGHT_WALL_BLUE && !has_key(KEY3)));

    return 0;
}

void toggle_char_tile()
{
    IOWR(VGA_BASE,WHICH_CHAR_TILE_OFFSET,(IORD(VGA_BASE,WHICH_CHAR_TILE_OFFSET)+1)%4);
}

// TODO make it so that the character moves instead of the maze if moving
// the maze would mean we can see outside the boundaries of the maze
void animate(int dir, wiimote_t *wiimote) {
    int yinc, xinc, num;

    int maze_x = get_maze_x();
    int maze_y = get_maze_y();

    if( dir == UP ) { yinc = SPEED_PIX; xinc = 0; }
    else if( dir == DOWN ) { yinc = -SPEED_PIX; xinc = 0; }
    else if( dir == LEFT ) { yinc = 0; xinc = SPEED_PIX; }
    else if( dir == RIGHT) { yinc = 0; xinc = -SPEED_PIX; }


    for( num = 0; num < TILE_SIZE/SPEED_PIX; num++ )
    {
      /*
       * Was initially going to be used to play a jingle during gameplay.
```

```
       * Due to issues discussed in the final report, this wasn't possible
       */
      //make_moving_sound();
      move_maze_to(maze_x+xinc,maze_y+yinc);
      if( num%4 == 0 ) toggle_char_tile();

      // Used to help clear input queue of accelerometer data.
      animate_pause(wiimote);

      maze_x += xinc;
      maze_y += yinc;
    }

}

// returns 1 if x,y is the goal loc, 0 otherwise
int is_at_goal_loc(int **the_level, int x, int y) {
    return (the_level[DESC_BASIC_OFFSET][DESC_BASIC_XPOS_GOAL_OFFSET] == x &&
the_level[DESC_BASIC_OFFSET][DESC_BASIC_YPOS_GOAL_OFFSET] == y);
}

// returns which key (KEY1, KEY2, KEY3) is at x,y if not already collected
// or -1 if no key is there (including if it has been collected)
int ontop_of_which_key(int **the_level, int x, int y) {
    if( the_level[DESC_OBJ_OFFSET+y][x] == KEY_RED && !has_key(KEY1) ) return KEY1;
    else if( the_level[DESC_OBJ_OFFSET+y][x] == KEY_GREEN && !has_key(KEY2) ) return
KEY2;
    else if( the_level[DESC_OBJ_OFFSET+y][x] == KEY_BLUE && !has_key(KEY3) ) return KEY3;
    return -1;
}

int get_char_start_x(int **the_level) {
    return the_level[DESC_BASIC_OFFSET][DESC_BASIC_XPOS_CHAR_OFFSET];
}

int get_char_start_y(int **the_level) {
    return the_level[DESC_BASIC_OFFSET][DESC_BASIC_YPOS_CHAR_OFFSET];
}

// TODO make this get the wiimote buttons that were pressed
int get_next_move(wiimote_t *wiimote) {

  int move=0;

  while(1){
    if(wiimote_is_open(wiimote)){
      if(wiimote_update(wiimote)<0){
        wiimote_disconnect(wiimote);
        break;
      }

      // Checks for A button.  Turns on accelerometer.
      if (wiimote->keys.a && wiimote->mode.acc == 0) {
        printf("Accelerometer enabled.\n");
        wiimote->mode.acc = 1;
      }

      /*
       * These statements check for certain tilt thresholds.  If they
       * are registered, we move in the respective direction.
       */
      if(wiimote->tilt.x < -50.0) {
        printf("Moving Left, got x acceleration , -50 \n");
        move = LEFT;
        wiimote_update(wiimote);
                break;
      }

      if(wiimote->tilt.x > 50.0 ) {
        printf("Moving Right, got x acceleration,  > 50 \n");
        move = RIGHT;
```

```
        wiimote_update(wiimote);
        break;
      }

      if(wiimote->tilt.y > 30.0 ) {
        printf("Moving Down, got y acceleration, > 30 \n");
        move = DOWN;
        wiimote_update(wiimote);
        break;
      }

      if(wiimote->tilt.y < -30.0) {
        printf("Moving Up, got y acceleration, < -30 \n");
        move = UP;
        wiimote_update(wiimote);
        break;
      }

      /*
       * These functions provide the same functionality of movement but use
       * the directional pad instead.
       *
       *  if(wiimote->keys.up) {
       *  move = UP;
       *  printf("up was pressed\n");
       *  break;
       *  }
       *
       *  if(wiimote->keys.down) {
       *  move = DOWN;
       *  printf("down was pressed\n");
       *  break;
       *  }
       *
       *  if(wiimote->keys.left){
       *  move = LEFT;
       *  printf("left was pressed\n");
       *  break;
       *  }
       *
       *  if(wiimote->keys.right) {
       *  move = RIGHT;
       *  printf("right was pressed\n");
       *  break;
       *  }
      */

      // Checks if Home key was pressed. Disconnects if it was.
      if(wiimote->keys.home) {
        printf("Home key pressed.  WiiMote disconnected. \n");
        wiimote_disconnect(wiimote);
        exit(0);
      }
    }
  }
  return move;
}

/*
 * Plays a tone three times when goal is reached
 */
void make_got_to_goal_sound(wiimote_t *wiimote) {
        int i=0;
        for(i=0; i<3; i++) {
                make_got_key_sound(wiimote);
        }
}

/*
 * Checks if the Wiimote is still connected and plays a 20 byte sample
 * if it is. Calls wiimote_update to synchronize
```

```
 */
void make_got_key_sound(wiimote_t *wiimote) {
  if (wiimote_is_open(wiimote)) {
    printf("hit key sound \n");
    wiimote_speaker_freq(wiimote, WIIMOTE_FREQ_44800HZ);
    wiimote_speaker_volume(wiimote, 60);
    wiimote_speaker_play(wiimote, sample, 20);
    wiimote_update(wiimote);
  }
}

/*
 * Was going to be used for jingle sounds.
 * Limitations discussed in final report
 */
void make_moving_sound() {

}

/*
 * Checks if the WiiMote is still connected and plays a single
 * 20 byte sample. Calls wiimote_update to synchronize
 */
void make_hit_wall_sound(wiimote_t *wiimote)
{
  if (wiimote_is_open(wiimote)) {
    printf("hit wall sound \n");
    wiimote_speaker_freq(wiimote, 90);
    wiimote_speaker_volume(wiimote, 60);
    wiimote_speaker_play(wiimote, sample, 20);
    wiimote_update(wiimote);
  }
}

void pause() {
    int foo;
    for(foo = 0; foo < PAUSE_AMT; foo++);
}

/*
 * This function was used to help synchronize the wiimote without the
 * processing overhead of wiimote_update.  Helped clear the input queue
 * of packets waiting to be processed by just getting the wiimotes state
 * and doing nothing with it
 */
void animate_pause(wiimote_t *wiimote) {
       wiimote_state_t ev ={{0}};

       int foo;
       int animate_pause_amt = 3;

       for(foo = 0; foo < animate_pause_amt; foo++) {
         wiimote_get_state(wiimote, &ev);
       }
}
```

wii_vga.vhd

```
--------------------------------------------------------------------------------
--
-- Simple VGA raster display
--
-- Stephen A. Edwards
-- sedwards@cs.columbia.edu
--
--------------------------------------------------------------------------------
library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;
```

```
entity wii_vga is

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

    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 std_logic_vector(9 downto 0) -- Blue[9:0]
    );

end wii_vga;

architecture rtl of wii_vga 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;

  -- Signals for the video controller
  signal Hcount, sprite_x : unsigned(15 downto 0);  -- Horizontal position (0-800)
  signal Vcount, sprite_y : unsigned(15 downto 0);  -- Vertical position (0-524)
  signal EndOfLine, EndOfField : std_logic;
  signal tmp_vga_r : std_logic_vector(9 downto 0) := "0000000000";
  signal tmp_vga_g : std_logic_vector(9 downto 0) := "0000000000";
  signal tmp_vga_b : std_logic_vector(9 downto 0) := "0000000000";

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

  -- Game VGA signals
  signal draw_char, draw_obj, draw_maze, draw_goal : std_logic;

  signal which_maze_tile : integer := 0;                                          --
which of the 4 actual maze tiles to use
  signal maze_x, maze_y  : integer := 0;                                          --
rotation-adjusted indexes into that maze tile
  signal rect_maze_h, rect_maze_v : std_logic;                                    -- are
we currently within the bounds of the maze tile?
  signal maze_tile_x, maze_tile_y : integer := 0;                         -- indexes
into the maze_desc array
  signal valid_maze_tile : std_logic;                                             -- is
this the empty tile?

  signal which_obj_tile : integer := 0;                                           --
which of the 4 actual obj tiles to use
  signal obj_x, obj_y  : integer := 0;                                            --
rotation-adjusted indexes into that obj tile
  signal rect_obj_h, rect_obj_v : std_logic;                        -- are we currently
within the bounds of the obj tile?
```

```
  signal obj_tile_x, obj_tile_y : integer := 0;                            -- indexes
into the obj_desc array
  signal can_show_cur_color : std_logic;                                   -- allowed to
show the obj of the color currently being drawn?
  signal valid_obj_tile : std_logic;                                       -- is this the
empty tile?

  signal rect_goal_h, rect_goal_v : std_logic;                             -- are we
currently within the bounds of the goal tile?

  signal which_char_tile : unsigned(15 downto 0) := "0000000000000000";    -- which
character tile to use
  signal tmp_char_tile : unsigned(15 downto 0) :=   "0000000000000000";    -- which
character tile to use (temp)
  signal char_x, char_y  : integer := 0;
       -- indexes into that char tile
  signal rect_char_h, rect_char_v : std_logic;
       -- are we currently within the bounds of the char tile?

  signal count_x, count_y : unsigned(4 downto 0) := "00000";
              -- unrotated indexes into maze and object arrays
  signal tmp_count_x, tmp_count_y : unsigned(15 downto 0) := "0000000000000000";  --
unrotated indexes into maze and object arrays (temp)
  signal anim_count   : integer := 0;
                             -- a count of clock cycles so that we can animate at a
reasonable speed

  -- Game parameters
  constant TILE_SIZE        : integer := 32;
       -- the size of a tile in pixels
  constant MAX_MAZE_SIZE_X  : integer := 16;
              -- the size of the maze in number of tiles
  constant MAX_MAZE_SIZE_Y  : integer := 16;
              -- the size of the maze in number of tiles
  constant EXTRA_LEFT         : unsigned(15 downto 0) := to_unsigned(HTOTAL,16);   -- The
extra amount of screen available at the left edge of the physical screen
  constant EXTRA_TOP          : unsigned(15 downto 0) := to_unsigned(VTOTAL,16);  -- The
extra amount of screen available at the top edge of the physical screen

  -- Actual game tiles
  type tile_type is array(0 to TILE_SIZE-1) of unsigned(0 to TILE_SIZE-1);
  type maze_tile_type is array(0 to 3) of tile_type;
  constant MAZE_TILES : maze_tile_type :=

       ( ( "11111000000000000000000000000000",

       "11111000000000000000000000000000",

       "11111000000000000000000000000000",
                                              "11111000000000000000000000000000",
                                              "11111000000000000000000000000000",
                                              "11111000000000000000000000000000",
                                              "11111000000000000000000000000000",
                                              "11111000000000000000000000000000",
                                              "11111000000000000000000000000000",
                                              "11111000000000000000000000000000",
                                              "11111000000000000000000000000000",
                                              "11111000000000000000000000000000",
                                              "11111000000000000000000000000000",
                                              "11111000000000000000000000000000",
                                              "11111000000000000000000000000000",
                                              "11111000000000000000000000000000",
                                              "11111000000000000000000000000000",
                                              "11111000000000000000000000000000",
                                              "11111000000000000000000000000000",
                                              "11111000000000000000000000000000",
                                              "11111000000000000000000000000000",
                                              "11111000000000000000000000000000",
                                              "11111000000000000000000000000000",
                                              "11111000000000000000000000000000",
                                              "11111000000000000000000000000000",
```

```
"11111000000000000000000000000000",
"11111000000000000000000000000000",
"11111000000000000000000000000000",
"11111000000000000000000000000000",
"11111000000000000000000000000000",

"11111000000000000000000000000000",

"11111000000000000000000000000000" ),


( "11111111111111111111111111111111",

"11111111111111111111111111111111",

"11111111111111111111111111111111",
"11111111111111111111111111111111",
"11111111111111111111111111111111",
"11111000000000000000000000000000",
"11111000000000000000000000000000",
"11111000000000000000000000000000",
"11111000000000000000000000000000",
"11111000000000000000000000000000",
"11111000000000000000000000000000",
"11111000000000000000000000000000",
"11111000000000000000000000000000",
"11111000000000000000000000000000",
"11111000000000000000000000000000",
"11111000000000000000000000000000",
"11111000000000000000000000000000",
"11111000000000000000000000000000",
"11111000000000000000000000000000",
"11111000000000000000000000000000",
"11111000000000000000000000000000",
"11111000000000000000000000000000",
"11111000000000000000000000000000",
"11111000000000000000000000000000",
"11111000000000000000000000000000",
"11111000000000000000000000000000",
"11111000000000000000000000000000",
"11111000000000000000000000000000",
"11111000000000000000000000000000",
"11111000000000000000000000000000",

"11111000000000000000000000000000",

"11111000000000000000000000000000" ),


( "11111000000000000000000000011111",

"11111000000000000000000000011111",

"11111000000000000000000000011111",
"11111000000000000000000000011111",
"11111000000000000000000000011111",
"11111000000000000000000000011111",
"11111000000000000000000000011111",
"11111000000000000000000000011111",
"11111000000000000000000000011111",
"11111000000000000000000000011111",
"11111000000000000000000000011111",
"11111000000000000000000000011111",
"11111000000000000000000000011111",
"11111000000000000000000000011111",
"11111000000000000000000000011111",
"11111000000000000000000000011111",
"11111000000000000000000000011111",
"11111000000000000000000000011111",
"11111000000000000000000000011111",
"11111000000000000000000000011111",
```

```
"11111000000000000000000000011111",
"11111000000000000000000000011111",
"11111000000000000000000000011111",
"11111000000000000000000000011111",
"11111000000000000000000000011111",
"11111000000000000000000000011111",
"11111000000000000000000000011111",
"11111000000000000000000000011111",
"11111000000000000000000000011111",
"11111000000000000000000000011111",

"11111000000000000000000000011111",

"11111000000000000000000000011111" ),


( "11111000000000000000000000011111",

"11111000000000000000000000011111",

"11111000000000000000000000011111",
"11111000000000000000000000011111",
"11111000000000000000000000011111",
"11111000000000000000000000011111",
"11111000000000000000000000011111",
"11111000000000000000000000011111",
"11111000000000000000000000011111",
"11111000000000000000000000011111",
"11111000000000000000000000011111",
"11111000000000000000000000011111",
"11111000000000000000000000011111",
"11111000000000000000000000011111",
"11111000000000000000000000011111",
"11111000000000000000000000011111",
"11111000000000000000000000011111",
"11111000000000000000000000011111",
"11111000000000000000000000011111",
"11111000000000000000000000011111",
"11111000000000000000000000011111",
"11111000000000000000000000011111",
"11111000000000000000000000011111",
"11111000000000000000000000011111",
"11111000000000000000000000011111",
"11111000000000000000000000011111",
"11111000000000000000000000011111",
"11111111111111111111111111111111",
"11111111111111111111111111111111",
"11111111111111111111111111111111",

"11111111111111111111111111111111",

"11111111111111111111111111111111" ) );

type obj_tile_type is array(0 to 1) of tile_type;
constant OBJ_TILES : obj_tile_type :=

( ( "11111000000000000000000000000000",

"11111000000000000000000000000000",

"11111000000000000000000000000000",
"11111000000000000000000000000000",
"11111000000000000000000000000000",
"11111000000000000000000000000000",
"11111000000000000000000000000000",
"11111000000000000000000000000000",
"11111000000000000000000000000000",
"11111000000000000000000000000000",
"11111000000000000000000000000000",
"11111000000000000000000000000000",
"11111000000000000000000000000000",
```

```
                                                    "11111000000000000000000000000000",
                                                    "11111000000000000000000000000000",
                                                    "11111000000000000000000000000000",
                                                    "11111000000000000000000000000000",
                                                    "11111000000000000000000000000000",
                                                    "11111000000000000000000000000000",
                                                    "11111000000000000000000000000000",
                                                    "11111000000000000000000000000000",
                                                    "11111000000000000000000000000000",
                                                    "11111000000000000000000000000000",
                                                    "11111000000000000000000000000000",
                                                    "11111000000000000000000000000000",
                                                    "11111000000000000000000000000000",
                                                    "11111000000000000000000000000000",
                                                    "11111000000000000000000000000000",
                                                    "11111000000000000000000000000000",
        "11111000000000000000000000000000",

        "11111000000000000000000000000000" ),

( "00000000000000000000000000000000",

        "00000000000000000000000000000000",

        "00000000000000000000000000000000",
                                                    "00000000000000000000000000000000",
                                                    "00000000000000000000000000000000",
                                                    "00000000000000000000000000000000",
                                                    "00000000000000000000000000000000",
                                                    "00000000000000000000000000000000",
                                                    "00000000000000000000000000000000",
                                                    "00000000000000000000000000000000",

"00000000000000111100000000000000",
                                                    "00000000000000111111000000000000",
                                                    "00000000000000110011000000000000",
                                                    "00000000000000110011000000000000",
                                                    "00000000000000111111000000000000",
                                                    "00000000000000011110000000000000",
                                                    "00000000000000011000000000000000",
                                                    "00000000000000011110000000000000",
                                                    "00000000000000011110000000000000",
                                                    "00000000000000011000000000000000",
                                                    "00000000000000011111000000000000",
                                                    "00000000000000011111000000000000",
                                                    "00000000000000000000000000000000",
                                                    "00000000000000000000000000000000",
                                                    "00000000000000000000000000000000",
                                                    "00000000000000000000000000000000",
                                                    "00000000000000000000000000000000",
                                                    "00000000000000000000000000000000",
                                                    "00000000000000000000000000000000",
                                                    "00000000000000000000000000000000",
        "00000000000000000000000000000000",

        "00000000000000000000000000000000" ) );


  type char_tile_type is array(0 to 3) of tile_type;
  constant CHAR_TILES : char_tile_type :=

        ( ( "00000000000000000000000000000000",

        "00000000000000000000000000000000",

        "00000000000000000000000000000000",
                                                    "00000000000000000000000000000000",
```

```
                                                    "00000000000000000000000000000000",
                                                    "00000000000000000000000000000000",
                                                    "00000000000000000000000000000000",
                                                    "00000000000000000000000000000000",
                                                    "00000000000001111111000000000000",
                                                    "00000000000011111111000000000000",
                                                    "00000000000011111111000000000000",
                                                    "00000000000001111111000000000000",
                                                    "00000000000000011000000000000000",
                                                    "00000000000000011000000000000000",
                                                    "00000000111000110001110000000000",
                                                    "00000000000111011011100000000000",
                                                    "00000000000001111111000000000000",
                                                    "00000000000000011000000000000000",
                                                    "00000000000000011000000000000000",
                                                    "00000000000000011000000000000000",
                                                    "00000000000000011000000000000000",
                                                    "00000000000001111111000000000000",
                                                    "00000000000111000011100000000000",
                                                    "00000000111110000011110000000000",
                                                    "00000000000000000000000000000000",
                                                    "00000000000000000000000000000000",
                                                    "00000000000000000000000000000000",
                                                    "00000000000000000000000000000000",
                                                    "00000000000000000000000000000000",
        "00000000000000000000000000000000",

        "00000000000000000000000000000000" ),

( "00000000000000000000000000000000",

        "00000000000000000000000000000000",

        "00000000000000000000000000000000",
                                                    "00000000000000000000000000000000",
                                                    "00000000000000000000000000000000",
                                                    "00000000000000000000000000000000",
                                                    "00000000000000000000000000000000",
                                                    "00000000000001111111000000000000",
                                                    "00000000000011111111000000000000",
                                                    "00000000000011111111000000000000",
                                                    "00000000000001111111000000000000",
                                                    "00000000000000011000000000000000",
                                                    "00000000000000011000000000000000",
                                                    "00000000000000011000000000000000",
                                                    "00000000000000011000000000000000",
                                                    "00000001111111111111111100000000",
                                                    "00000000000000011000000000000000",
                                                    "00000000000000011000000000000000",
                                                    "00000000000000011000000000000000",
                                                    "00000000000000011000000000000000",
                                                    "00000000000001111111000000000000",
                                                    "00000000000111000011100000000000",
                                                    "00000000111110000011110000000000",
                                                    "00000000000000000000000000000000",
                                                    "00000000000000000000000000000000",
                                                    "00000000000000000000000000000000",
                                                    "00000000000000000000000000000000",
                                                    "00000000000000000000000000000000",
        "00000000000000000000000000000000",

        "00000000000000000000000000000000" ),

( "00000000000000000000000000000000",
```

```
        "00000000000000000000000000000000",

        "00000000000000000000000000000000",
                                                "00000000000000000000000000000000",
                                                "00000000000000000000000000000000",
                                                "00000000000000000000000000000000",
                                                "00000000000000000000000000000000",
                                                "00000000000000000000000000000000",
                                                "00000000000001111111000000000000",
                                                "00000000000011111111000000000000",
                                                "00000000000011111111000000000000",
                                                "00000000000001111111000000000000",
                                                "00000000000000011000000000000000",
                                                "00000000000000011000000000000000",
                                                "00000000011100011000111000000000",
                                                "00000000000111011011100000000000",
                                                "00000000000001111111000000000000",
                                                "00000000000000011000000000000000",
                                                "00000000000000011000000000000000",
                                                "00000000000000011000000000000000",
                                                "00000000000000011000000000000000",
                                                "00000000000001111111000000000000",
                                                "00000000000111000011100000000000",
                                                "00000000111110000011110000000000",
                                                "00000000000000000000000000000000",
                                                "00000000000000000000000000000000",
                                                "00000000000000000000000000000000",
                                                "00000000000000000000000000000000",
                                                "00000000000000000000000000000000",
        "00000000000000000000000000000000",

        "00000000000000000000000000000000" ),

( "00000000000000000000000000000000",

        "00000000000000000000000000000000",

        "00000000000000000000000000000000",
                                                "00000000000000000000000000000000",
                                                "00000000000000000000000000000000",
                                                "00000000000000000000000000000000",
                                                "00000000000000000000000000000000",
                                                "00000000000001111111000000000000",
                                                "00000000000011111111000000000000",
                                                "00000000000011111111000000000000",
                                                "00000000000001111111000000000000",
                                                "00000000000000011000000000000000",
                                                "00000000000000011000000000000000",
                                                "00000000000000011000000000000000",
                                                "00000000000000011000000000000000",
                                                "00000000111111111111111100000000",
                                                "00000000000000011000000000000000",
                                                "00000000000000011000000000000000",
                                                "00000000000000011000000000000000",
                                                "00000000000000011000000000000000",
                                                "00000000000001111111000000000000",
                                                "00000000000111000011100000000000",
                                                "00000000111110000011110000000000",
                                                "00000000000000000000000000000000",
                                                "00000000000000000000000000000000",
                                                "00000000000000000000000000000000",
                                                "00000000000000000000000000000000",
                                                "00000000000000000000000000000000",
                                                "00000000000000000000000000000000",
        "00000000000000000000000000000000",
```

```
      "00000000000000000000000000000000" ) );

  constant GOAL_TILE : tile_type :=

( "00000000000000000000000000000000",

      "00000000000000000000000000000000",

      "00000000000000000000000000000000",
                                                  "00000000000000000000000000000000",
                                                  "00000000000000000000000000000000",
                                                  "00000000000000000000000000000000",
                                                  "00000000000000000000000000000000",
                                                  "00000001111111111111111100000000",
                                                  "00000001111111111111111110000000",
                                                  "00000001100000011000000110000000",
                                                  "00000001100000011000000110000000",
                                                  "00000001100000011000000110000000",
                                                  "00000001100000011000000110000000",
                                                  "00000001100000011000000110000000",
                                                  "00000001100000011000000110000000",
                                                  "00000001111111111111111110000000",
                                                  "00000001111111111111111110000000",
                                                  "00000001100000011000000110000000",
                                                  "00000001100000011000000110000000",
                                                  "00000001100000011000000110000000",
                                                  "00000001100000011000000110000000",
                                                  "00000001100000011000000110000000",
                                                  "00000001100000011000000110000000",
                                                  "00000001111111111111111110000000",
                                                  "00000001111111111111111100000000",
                                                  "00000000000000000000000000000000",
                                                  "00000000000000000000000000000000",
                                                  "00000000000000000000000000000000",
                                                  "00000000000000000000000000000000",
                                                  "00000000000000000000000000000000",

      "00000000000000000000000000000000",

      "00000000000000000000000000000000" );

  -- Virtual Tiles

  -- Virtual Maze tiles
  -- Bit 7-6: index into MAZE_TILES array
  -- Bit 5-4: rotation (00 = none, 01 = 90 c, 10 = 180 c, 11 = 270 c)
  -- Bit 3:   allows player to move left? (1 if affirmative)          NO LONGER USED
  -- Bit 2:   allows player to move right? (1 if affirmative)         NO LONGER USED
  -- Bit 1:   allows player to move up? (1 if affirmative)            NO LONGER USED
  -- Bit 0:   allows player to move down? (1 if affirmative)          NO LONGER USED
  type virtual_maze_tile_type is array(0 to 14) of unsigned(7 downto 0);
  constant VIRTUAL_MAZE_TILES : virtual_maze_tile_type := ( "00000111",

                      "00011101",

                      "00101011",

                      "00111110",

                      "01000101",

                      "01011001",

                      "01101010",

                      "01110110",

                      "10000011",

                      "10011100",
```

```
                    "11000010",

                    "11010100",

                    "11100001",

                    "11111000",

                    "11111111" );


     constant LEFT_WALL                    : unsigned(3 downto 0) := "0000";
     constant TOP_WALL                     : unsigned(3 downto 0) := "0001";
     constant RIGHT_WALL            : unsigned(3 downto 0) := "0010";
     constant BOTTOM_WALL           : unsigned(3 downto 0) := "0011";
     constant TOP_LEFT_CORNER       : unsigned(3 downto 0) := "0100";
     constant TOP_RIGHT_CORNER      : unsigned(3 downto 0) := "0101";
     constant BOTTOM_RIGHT_CORNER : unsigned(3 downto 0) := "0110";
     constant BOTTOM_LEFT_CORNER  : unsigned(3 downto 0) := "0111";
     constant VERTICAL_HALL                : unsigned(3 downto 0) := "1000";
     constant HORIZONTAL_HALL       : unsigned(3 downto 0) := "1001";
     constant BOTTOM_DEADEND        : unsigned(3 downto 0) := "1010";
     constant LEFT_DEADEND          : unsigned(3 downto 0) := "1011";
     constant TOP_DEADEND           : unsigned(3 downto 0) := "1100";
     constant RIGHT_DEADEND                : unsigned(3 downto 0) := "1101";
     constant NO_MAZE_TILE          : unsigned(3 downto 0) := "1110";

-- Virtual object tiles
-- Bit 8:   index into OBJ_TILES array
-- Bit 7-6: rotation (00 = none, 01 = 90 c, 10 = 180 c, 11 = 270 c)
-- Bit 5-4: color (00 = red, 01 = green, 10 = blue, 11 = invalid)
-- Bit 3:   allows player to move left? (1 if affirmative)           NO LONGER USED
-- Bit 2:   allows player to move right? (1 if affirmative)          NO LONGER USED
-- Bit 1:   allows player to move up? (1 if affirmative)             NO LONGER USED
-- Bit 0:   allows player to move down? (1 if affirmative)           NO LONGER USED
type virtual_obj_tile_type is array(0 to 15) of unsigned(8 downto 0);
constant VIRTUAL_OBJ_TILES : virtual_obj_tile_type := ( "000000111",

               "001001101",

               "010001011",

               "011001110",

               "000010111",

               "001011101",

               "010011011",

               "011011110",

               "000100111",

               "001101101",

               "010101011",

               "011101110",

               "100001111",

               "100011111",

               "100101111",

               "111111111" );


     constant LEFT_WALL_RED                       : unsigned(3 downto 0) := "0000";
```

```
        constant TOP_WALL_RED                  : unsigned(3 downto 0) := "0001";
        constant RIGHT_WALL_RED                : unsigned(3 downto 0) := "0010";
        constant BOTTOM_WALL_RED               : unsigned(3 downto 0) := "0011";
        constant LEFT_WALL_GREEN               : unsigned(3 downto 0) := "0100";
        constant TOP_WALL_GREEN                    : unsigned(3 downto 0) := "0101";
        constant RIGHT_WALL_GREEN              : unsigned(3 downto 0) := "0110";
        constant BOTTOM_WALL_GREEN     : unsigned(3 downto 0) := "0111";
        constant LEFT_WALL_BLUE                : unsigned(3 downto 0) := "1000";
        constant TOP_WALL_BLUE                         : unsigned(3 downto 0) := "1001";
        constant RIGHT_WALL_BLUE               : unsigned(3 downto 0) := "1010";
        constant BOTTOM_WALL_BLUE              : unsigned(3 downto 0) := "1011";
        constant KEY_RED                               : unsigned(3 downto 0) := "1100";
        constant KEY_GREEN                             : unsigned(3 downto 0) := "1101";
        constant KEY_BLUE                              : unsigned(3 downto 0) := "1110";
        constant NO_OBJ_TILE                   : unsigned(3 downto 0) := "1111";


  -- Maze and object descriptors
  type desc_sub_type is array(0 to MAX_MAZE_SIZE_X-1) of unsigned(3 downto 0);
  type desc_type is array(0 to MAX_MAZE_SIZE_Y-1) of desc_sub_type;
  signal maze_desc : desc_type := ( (LEFT_DEADEND, LEFT_DEADEND, LEFT_DEADEND,
LEFT_DEADEND, LEFT_DEADEND, LEFT_DEADEND, LEFT_DEADEND, LEFT_DEADEND, LEFT_DEADEND,
LEFT_DEADEND, LEFT_DEADEND, LEFT_DEADEND, LEFT_DEADEND, LEFT_DEADEND, LEFT_DEADEND,
LEFT_DEADEND),
                                                                  (LEFT_DEADEND,
LEFT_DEADEND, LEFT_DEADEND, LEFT_DEADEND, LEFT_DEADEND, LEFT_DEADEND, LEFT_DEADEND,
LEFT_DEADEND, LEFT_DEADEND, LEFT_DEADEND, LEFT_DEADEND, LEFT_DEADEND, LEFT_DEADEND,
LEFT_DEADEND, LEFT_DEADEND, LEFT_DEADEND),
                                                                  (LEFT_DEADEND,
LEFT_DEADEND, LEFT_DEADEND, LEFT_DEADEND, LEFT_DEADEND, LEFT_DEADEND, LEFT_DEADEND,
LEFT_DEADEND, LEFT_DEADEND, LEFT_DEADEND, LEFT_DEADEND, LEFT_DEADEND, LEFT_DEADEND,
LEFT_DEADEND, LEFT_DEADEND, LEFT_DEADEND),
                                                               (LEFT_DEADEND,
LEFT_DEADEND, LEFT_DEADEND, LEFT_DEADEND, LEFT_DEADEND, LEFT_DEADEND, LEFT_DEADEND,
LEFT_DEADEND, LEFT_DEADEND, LEFT_DEADEND, LEFT_DEADEND, LEFT_DEADEND, LEFT_DEADEND,
LEFT_DEADEND, LEFT_DEADEND, LEFT_DEADEND),
                                                                  (LEFT_DEADEND,
LEFT_DEADEND, LEFT_DEADEND, LEFT_DEADEND, LEFT_DEADEND, LEFT_DEADEND, LEFT_DEADEND,
LEFT_DEADEND, LEFT_DEADEND, LEFT_DEADEND, LEFT_DEADEND, LEFT_DEADEND, LEFT_DEADEND,
LEFT_DEADEND, LEFT_DEADEND, LEFT_DEADEND),
                                                                  (LEFT_DEADEND,
LEFT_DEADEND, LEFT_DEADEND, LEFT_DEADEND, LEFT_DEADEND, LEFT_DEADEND, LEFT_DEADEND,
LEFT_DEADEND, LEFT_DEADEND, LEFT_DEADEND, LEFT_DEADEND, LEFT_DEADEND, LEFT_DEADEND,
LEFT_DEADEND, LEFT_DEADEND, LEFT_DEADEND),
                                                                  (LEFT_DEADEND,
LEFT_DEADEND, LEFT_DEADEND, LEFT_DEADEND, LEFT_DEADEND, LEFT_DEADEND, LEFT_DEADEND,
LEFT_DEADEND, LEFT_DEADEND, LEFT_DEADEND, LEFT_DEADEND, LEFT_DEADEND, LEFT_DEADEND,
LEFT_DEADEND, LEFT_DEADEND, LEFT_DEADEND),
                                                               (LEFT_DEADEND,
LEFT_DEADEND, LEFT_DEADEND, LEFT_DEADEND, LEFT_DEADEND, LEFT_DEADEND, LEFT_DEADEND,
LEFT_DEADEND, LEFT_DEADEND, LEFT_DEADEND, LEFT_DEADEND, LEFT_DEADEND, LEFT_DEADEND,
LEFT_DEADEND, LEFT_DEADEND, LEFT_DEADEND),
                                                               (LEFT_DEADEND,
LEFT_DEADEND, LEFT_DEADEND, LEFT_DEADEND, LEFT_DEADEND, LEFT_DEADEND, LEFT_DEADEND,
LEFT_DEADEND, LEFT_DEADEND, LEFT_DEADEND, LEFT_DEADEND, LEFT_DEADEND, LEFT_DEADEND,
LEFT_DEADEND, LEFT_DEADEND, LEFT_DEADEND),
                                                                  (LEFT_DEADEND,
LEFT_DEADEND, LEFT_DEADEND, LEFT_DEADEND, LEFT_DEADEND, LEFT_DEADEND, LEFT_DEADEND,
LEFT_DEADEND, LEFT_DEADEND, LEFT_DEADEND, LEFT_DEADEND, LEFT_DEADEND, LEFT_DEADEND,
LEFT_DEADEND, LEFT_DEADEND, LEFT_DEADEND),
                                                                  (LEFT_DEADEND,
LEFT_DEADEND, LEFT_DEADEND, LEFT_DEADEND, LEFT_DEADEND, LEFT_DEADEND, LEFT_DEADEND,
LEFT_DEADEND, LEFT_DEADEND, LEFT_DEADEND, LEFT_DEADEND, LEFT_DEADEND, LEFT_DEADEND,
LEFT_DEADEND, LEFT_DEADEND, LEFT_DEADEND),
                                                               (LEFT_DEADEND,
LEFT_DEADEND, LEFT_DEADEND, LEFT_DEADEND, LEFT_DEADEND, LEFT_DEADEND, LEFT_DEADEND,
LEFT_DEADEND, LEFT_DEADEND, LEFT_DEADEND, LEFT_DEADEND, LEFT_DEADEND, LEFT_DEADEND,
LEFT_DEADEND, LEFT_DEADEND, LEFT_DEADEND),
                                                               (LEFT_DEADEND,
LEFT_DEADEND, LEFT_DEADEND, LEFT_DEADEND, LEFT_DEADEND, LEFT_DEADEND, LEFT_DEADEND,
```

```
LEFT_DEADEND, LEFT_DEADEND, LEFT_DEADEND, LEFT_DEADEND, LEFT_DEADEND, LEFT_DEADEND,
LEFT_DEADEND, LEFT_DEADEND, LEFT_DEADEND),
                                                             (LEFT_DEADEND,
LEFT_DEADEND, LEFT_DEADEND, LEFT_DEADEND, LEFT_DEADEND, LEFT_DEADEND, LEFT_DEADEND,
LEFT_DEADEND, LEFT_DEADEND, LEFT_DEADEND, LEFT_DEADEND, LEFT_DEADEND, LEFT_DEADEND,
LEFT_DEADEND, LEFT_DEADEND, LEFT_DEADEND),
                                                             (LEFT_DEADEND,
LEFT_DEADEND, LEFT_DEADEND, LEFT_DEADEND, LEFT_DEADEND, LEFT_DEADEND, LEFT_DEADEND,
LEFT_DEADEND, LEFT_DEADEND, LEFT_DEADEND, LEFT_DEADEND, LEFT_DEADEND, LEFT_DEADEND,
LEFT_DEADEND, LEFT_DEADEND, LEFT_DEADEND),
                                                             (LEFT_DEADEND,
LEFT_DEADEND, LEFT_DEADEND, LEFT_DEADEND, LEFT_DEADEND, LEFT_DEADEND, LEFT_DEADEND,
LEFT_DEADEND, LEFT_DEADEND, LEFT_DEADEND, LEFT_DEADEND, LEFT_DEADEND, LEFT_DEADEND,
LEFT_DEADEND, LEFT_DEADEND, LEFT_DEADEND) );

  signal obj_desc : desc_type :=  ( (NO_OBJ_TILE, NO_OBJ_TILE, NO_OBJ_TILE, NO_OBJ_TILE,
NO_OBJ_TILE, NO_OBJ_TILE, NO_OBJ_TILE, NO_OBJ_TILE, NO_OBJ_TILE, NO_OBJ_TILE, NO_OBJ_TILE,
NO_OBJ_TILE, NO_OBJ_TILE, NO_OBJ_TILE, NO_OBJ_TILE, NO_OBJ_TILE),
                                                                 (NO_OBJ_TILE,
NO_OBJ_TILE, NO_OBJ_TILE, NO_OBJ_TILE, NO_OBJ_TILE, NO_OBJ_TILE, NO_OBJ_TILE, NO_OBJ_TILE,
NO_OBJ_TILE, NO_OBJ_TILE, NO_OBJ_TILE, NO_OBJ_TILE, NO_OBJ_TILE, NO_OBJ_TILE, NO_OBJ_TILE,
NO_OBJ_TILE),
                                                                 (NO_OBJ_TILE,
NO_OBJ_TILE, NO_OBJ_TILE, NO_OBJ_TILE, NO_OBJ_TILE, NO_OBJ_TILE, NO_OBJ_TILE, NO_OBJ_TILE,
NO_OBJ_TILE, NO_OBJ_TILE, NO_OBJ_TILE, NO_OBJ_TILE, NO_OBJ_TILE, NO_OBJ_TILE, NO_OBJ_TILE,
NO_OBJ_TILE),
                                                                 (NO_OBJ_TILE,
NO_OBJ_TILE, NO_OBJ_TILE, NO_OBJ_TILE, NO_OBJ_TILE, NO_OBJ_TILE, NO_OBJ_TILE, NO_OBJ_TILE,
NO_OBJ_TILE, NO_OBJ_TILE, NO_OBJ_TILE, NO_OBJ_TILE, NO_OBJ_TILE, NO_OBJ_TILE, NO_OBJ_TILE,
NO_OBJ_TILE),
                                                                 (NO_OBJ_TILE,
NO_OBJ_TILE, NO_OBJ_TILE, NO_OBJ_TILE, NO_OBJ_TILE, NO_OBJ_TILE, NO_OBJ_TILE, NO_OBJ_TILE,
NO_OBJ_TILE, NO_OBJ_TILE, NO_OBJ_TILE, NO_OBJ_TILE, NO_OBJ_TILE, NO_OBJ_TILE, NO_OBJ_TILE,
NO_OBJ_TILE),
                                                                 (NO_OBJ_TILE,
NO_OBJ_TILE, NO_OBJ_TILE, NO_OBJ_TILE, NO_OBJ_TILE, NO_OBJ_TILE, NO_OBJ_TILE, NO_OBJ_TILE,
NO_OBJ_TILE, NO_OBJ_TILE, NO_OBJ_TILE, NO_OBJ_TILE, NO_OBJ_TILE, NO_OBJ_TILE, NO_OBJ_TILE,
NO_OBJ_TILE),
                                                                 (NO_OBJ_TILE,
NO_OBJ_TILE, NO_OBJ_TILE, NO_OBJ_TILE, NO_OBJ_TILE, NO_OBJ_TILE, NO_OBJ_TILE, NO_OBJ_TILE,
NO_OBJ_TILE, NO_OBJ_TILE, NO_OBJ_TILE, NO_OBJ_TILE, NO_OBJ_TILE, NO_OBJ_TILE, NO_OBJ_TILE,
NO_OBJ_TILE),
                                                                 (NO_OBJ_TILE,
NO_OBJ_TILE, NO_OBJ_TILE, NO_OBJ_TILE, NO_OBJ_TILE, NO_OBJ_TILE, NO_OBJ_TILE, NO_OBJ_TILE,
NO_OBJ_TILE, NO_OBJ_TILE, NO_OBJ_TILE, NO_OBJ_TILE, NO_OBJ_TILE, NO_OBJ_TILE, NO_OBJ_TILE,
NO_OBJ_TILE),
                                                                 (NO_OBJ_TILE,
NO_OBJ_TILE, NO_OBJ_TILE, NO_OBJ_TILE, NO_OBJ_TILE, NO_OBJ_TILE, NO_OBJ_TILE, NO_OBJ_TILE,
NO_OBJ_TILE, NO_OBJ_TILE, NO_OBJ_TILE, NO_OBJ_TILE, NO_OBJ_TILE, NO_OBJ_TILE, NO_OBJ_TILE,
NO_OBJ_TILE),
                                                                 (NO_OBJ_TILE,
NO_OBJ_TILE, NO_OBJ_TILE, NO_OBJ_TILE, NO_OBJ_TILE, NO_OBJ_TILE, NO_OBJ_TILE, NO_OBJ_TILE,
NO_OBJ_TILE, NO_OBJ_TILE, NO_OBJ_TILE, NO_OBJ_TILE, NO_OBJ_TILE, NO_OBJ_TILE, NO_OBJ_TILE,
NO_OBJ_TILE),
                                                                 (NO_OBJ_TILE,
NO_OBJ_TILE, NO_OBJ_TILE, NO_OBJ_TILE, NO_OBJ_TILE, NO_OBJ_TILE, NO_OBJ_TILE, NO_OBJ_TILE,
NO_OBJ_TILE, NO_OBJ_TILE, NO_OBJ_TILE, NO_OBJ_TILE, NO_OBJ_TILE, NO_OBJ_TILE, NO_OBJ_TILE,
NO_OBJ_TILE),
                                                                 (NO_OBJ_TILE,
NO_OBJ_TILE, NO_OBJ_TILE, NO_OBJ_TILE, NO_OBJ_TILE, NO_OBJ_TILE, NO_OBJ_TILE, NO_OBJ_TILE,
NO_OBJ_TILE, NO_OBJ_TILE, NO_OBJ_TILE, NO_OBJ_TILE, NO_OBJ_TILE, NO_OBJ_TILE, NO_OBJ_TILE,
NO_OBJ_TILE),
                                                                 (NO_OBJ_TILE,
NO_OBJ_TILE, NO_OBJ_TILE, NO_OBJ_TILE, NO_OBJ_TILE, NO_OBJ_TILE, NO_OBJ_TILE, NO_OBJ_TILE,
NO_OBJ_TILE, NO_OBJ_TILE, NO_OBJ_TILE, NO_OBJ_TILE, NO_OBJ_TILE, NO_OBJ_TILE, NO_OBJ_TILE,
NO_OBJ_TILE),
                                                                 (NO_OBJ_TILE,
NO_OBJ_TILE, NO_OBJ_TILE, NO_OBJ_TILE, NO_OBJ_TILE, NO_OBJ_TILE, NO_OBJ_TILE, NO_OBJ_TILE,
NO_OBJ_TILE, NO_OBJ_TILE, NO_OBJ_TILE, NO_OBJ_TILE, NO_OBJ_TILE, NO_OBJ_TILE, NO_OBJ_TILE,
NO_OBJ_TILE),
```

```
                                                                (NO_OBJ_TILE,
NO_OBJ_TILE, NO_OBJ_TILE, NO_OBJ_TILE, NO_OBJ_TILE, NO_OBJ_TILE, NO_OBJ_TILE, NO_OBJ_TILE,
NO_OBJ_TILE, NO_OBJ_TILE, NO_OBJ_TILE, NO_OBJ_TILE, NO_OBJ_TILE, NO_OBJ_TILE, NO_OBJ_TILE,
NO_OBJ_TILE),
                                                                (NO_OBJ_TILE,
NO_OBJ_TILE, NO_OBJ_TILE, NO_OBJ_TILE, NO_OBJ_TILE, NO_OBJ_TILE, NO_OBJ_TILE, NO_OBJ_TILE,
NO_OBJ_TILE, NO_OBJ_TILE, NO_OBJ_TILE, NO_OBJ_TILE, NO_OBJ_TILE, NO_OBJ_TILE, NO_OBJ_TILE,
NO_OBJ_TILE) );

  -- Other game signals
  signal which_keys      : unsigned(15 downto 0) := "0000000000000111";   -- which keys
haven't been collected (first bit is red, second is blue, third is green)

  signal maze_size_x      : unsigned(3 downto 0) := to_unsigned(MAX_MAZE_SIZE_X,4);
              -- the size of the maze in number of tiles
  signal maze_size_y      : unsigned(3 downto 0) := to_unsigned(MAX_MAZE_SIZE_Y,4);
              -- the size of the maze in number of tiles
  signal maze_size_pix_x  : unsigned(15 downto 0) := to_unsigned(MAX_MAZE_SIZE_X *
TILE_SIZE,16); -- the size of the maze in pixels (not settable by software, but should be
set by the hardware if the maze_size is set)
  signal maze_size_pix_y  : unsigned(15 downto 0) := to_unsigned(MAX_MAZE_SIZE_Y *
TILE_SIZE,16); -- the size of the maze in pixels (not settable by software, but should be
set by the hardware if the maze_size is set)
  signal maze_x_offset, tmp_maze_x_offset  : unsigned(15 downto 0) := to_unsigned(HSYNC +
HBACK_PORCH,16)+EXTRA_LEFT;    -- left position of beginning of maze in pixels
( including SYNC and BACK_PORCH)
  signal maze_y_offset, tmp_maze_y_offset  : unsigned(15 downto 0) := to_unsigned(VSYNC +
VBACK_PORCH,16)+EXTRA_TOP;     -- top position of the beginning of the maze

  constant window_x_center     : unsigned(15 downto 0) :=
to_unsigned(HSYNC+HBACK_PORCH+320,16);
  constant window_y_center     : unsigned(15 downto 0) :=
to_unsigned(VSYNC+VBACK_PORCH+240,16);
  signal half_window_size_x  : unsigned(7 downto 0) := "00000010";          -- horiz size
of window in which you can see the maze in # of tiles
  signal half_window_size_y  : unsigned(7 downto 0) := "00000010";          -- verti size
of window in which you can see the maze in # of tiles
  signal is_on_window : std_logic := '0';
  signal is_in_window : std_logic := '0';

  signal end_xpos_maze   : unsigned(15 downto 0) := to_unsigned(MAX_MAZE_SIZE_X-1,16);
  signal end_ypos_maze   : unsigned(15 downto 0) := to_unsigned(MAX_MAZE_SIZE_Y-1,16);

  signal cur_xpos_pix, tmp_cur_xpos_pix  : unsigned(15 downto 0) := to_unsigned(HSYNC +
HBACK_PORCH,16)+EXTRA_LEFT;          -- character's current x position in pixels
  signal cur_ypos_pix, tmp_cur_ypos_pix  : unsigned(15 downto 0) := to_unsigned(VSYNC +
VBACK_PORCH,16)+EXTRA_TOP;                                          -- character's current
y position in pixels

  signal foo : unsigned(15 downto 0);
begin


-- Horizontal and vertical counters

  HCounter : process (clk)
  begin
    if rising_edge(clk) then
      if reset_n = '0' 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_n = '0' 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, HBLANK, and VBLANK

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

HBlankGen : process (clk)
begin
  if rising_edge(clk) then
    if reset_n = '0' 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;

VSyncGen : process (clk)
begin
  if rising_edge(clk) then
    if reset_n = '0' 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;

VBlankGen : process (clk)
begin
  if rising_edge(clk) then
    if reset_n = '0' 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;
```

```
-------------------------------------------------------------
-- counters to keep track of current non-rotated indices into
-- maze and object tiles
-------------------------------------------------------------
PreIndexCounter : process (clk)
  begin
       if rising_edge(clk) then
      tmp_count_x <= EXTRA_LEFT - maze_x_offset;
         tmp_count_y <= EXTRA_TOP - maze_y_offset;
    end if;
 end process PreIndexCounter;

  IndexCounter : process (clk)
  begin
    if rising_edge(clk) then
      if reset_n = '0' or EndOfField = '1' then
               if maze_x_offset < EXTRA_LEFT then
                       count_x <= tmp_count_x(4 downto 0) - 1;
               else
                       count_x <= (others => '0');
               end if;

               if maze_y_offset < EXTRA_TOP then
                       count_y <= tmp_count_y(4 downto 0);
               else
                       count_y <= (others => '0');
               end if;

         else
               if Hcount + EXTRA_LEFT > maze_x_offset and Vcount + EXTRA_TOP >=
maze_y_offset then
                       count_x <= count_x + 1;
               end if;
               if Vcount + EXTRA_TOP >= maze_y_offset and EndOfLine='1' then
                       count_y <= count_y + 1;
                       if maze_x_offset < EXTRA_LEFT then
                               count_x <= tmp_count_x(4 downto 0) - 1;
                       else
                               count_x <= (others => '0');
                       end if;
               end if;
         end if;
   end if;
  end process IndexCounter;

-------------------------------------------------------------
-- VGA Controller signals for maze tiles
-------------------------------------------------------------
  MazeHGen : process (clk)
  begin
    if rising_edge(clk) then
      if reset_n = '0' or Hcount + EXTRA_LEFT = maze_x_offset or (Hcount =
HSYNC+HBACK_PORCH and maze_x_offset < EXTRA_LEFT) then
        rect_maze_h <= '1';
               maze_tile_x <= 0;
      elsif Hcount + EXTRA_LEFT= maze_x_offset + maze_size_pix_x - 1 or (Hcount =
HSYNC+HBACK_PORCH+HACTIVE) then
        rect_maze_h <= '0';
      end if;
       maze_tile_x <= to_integer((Hcount-maze_x_offset+EXTRA_LEFT) srl 5);
    end if;
  end process MazeHGen;

  MazeVGen : process (clk)
  begin
    if rising_edge(clk) then
      if reset_n = '0' then
        rect_maze_v <= '0';
           maze_tile_y <= 0;
      elsif EndOfLine = '1' then
```

```
        if Vcount + EXTRA_TOP = maze_y_offset or (Vcount = VSYNC+VBACK_PORCH and
maze_y_offset < EXTRA_TOP) then
          rect_maze_v <= '1';
        elsif Vcount + EXTRA_TOP = maze_y_offset + maze_size_pix_y - 1 or (Vcount =
VSYNC+VBACK_PORCH+VACTIVE) then
          rect_maze_v <= '0';
               end if;
      end if;
    maze_tile_y <= to_integer((Vcount-maze_y_offset+EXTRA_TOP) srl 5);
    end if;
  end process MazeVGen;

  MazeTileInfo : process (clk)
  begin
       if rising_edge(clk) then
      if reset_n = '0' then
               which_maze_tile <= 0;
               maze_x <= 0;
               maze_y <= 0;
         else
               -- take first two bits of virtual tile description to get real tile #
               which_maze_tile <=
to_integer(VIRTUAL_MAZE_TILES(to_integer(maze_desc(maze_tile_y)(maze_tile_x)))(7 downto
6));

               -- take next two to get rotation and set indices to appropriate values
               if VIRTUAL_MAZE_TILES(to_integer(maze_desc(maze_tile_y)(maze_tile_x)))(5
downto 4) = "00" then
                       maze_x <= to_integer(count_x);
                       maze_y <= to_integer(count_y);
               elsif
VIRTUAL_MAZE_TILES(to_integer(maze_desc(maze_tile_y)(maze_tile_x)))(5 downto 4) = "01"
then
                       maze_x <= to_integer(count_y);
                       maze_y <= to_integer(TILE_SIZE - 1 - count_x);
               elsif
VIRTUAL_MAZE_TILES(to_integer(maze_desc(maze_tile_y)(maze_tile_x)))(5 downto 4) = "10"
then
                       maze_x <= to_integer(TILE_SIZE - 1 - count_x);
                       maze_y <= to_integer(TILE_SIZE - 1 - count_y);
               elsif
VIRTUAL_MAZE_TILES(to_integer(maze_desc(maze_tile_y)(maze_tile_x)))(5 downto 4) = "11"
then
                       maze_x <= to_integer(TILE_SIZE - 1 - count_y);
                       maze_y <= to_integer(count_x);
               end if;

               -- check if this is the empty tile
               if maze_desc(maze_tile_y)(maze_tile_x) = NO_MAZE_TILE then
                       valid_maze_tile <= '0';
               else
                       valid_maze_tile <= '1';
               end if;

       end if;
  end if;
end process MazeTileInfo;

-----------------------------------------------------------------
-- check if this is the goal tile
-----------------------------------------------------------------
  GoalVHGen : process (clk)
  begin
    if rising_edge(clk) then
      if reset_n = '0' then
        rect_goal_v <= '0';
            rect_goal_h <= '0';
      else
               if maze_tile_x = end_xpos_maze and maze_tile_y = end_ypos_maze then
                       rect_goal_v <= '1';
                       rect_goal_h <= '1';
```

```
               else
                       rect_goal_v <= '0';
                       rect_goal_h <= '0';
               end if;
    end if;
 end if;
end process GoalVHGen;

-----------------------------------------------------------------
-- VGA controller signals for object tiles
-----------------------------------------------------------------
  ObjHGen : process (clk)
  begin
    if rising_edge(clk) then
      if reset_n = '0' or Hcount + EXTRA_LEFT = maze_x_offset or (Hcount =
HSYNC+HBACK_PORCH and maze_x_offset < EXTRA_LEFT) then
        rect_obj_h <= '1';
               obj_tile_x <= 0;
      elsif Hcount + EXTRA_LEFT = maze_x_offset + maze_size_pix_x - 1 or (Hcount =
HSYNC+HBACK_PORCH+HACTIVE) then
        rect_obj_h <= '0';
      end if;
       obj_tile_x <= to_integer((Hcount-maze_x_offset+EXTRA_LEFT) srl 5);
    end if;
  end process ObjHGen;

  ObjVGen : process (clk)
  begin
    if rising_edge(clk) then
      if reset_n = '0' then
        rect_obj_v <= '0';
            obj_tile_y <= 0;
      elsif EndOfLine = '1' then
        if Vcount + EXTRA_TOP = maze_y_offset or (Vcount = VSYNC+VBACK_PORCH and
maze_y_offset < EXTRA_TOP)  then
          rect_obj_v <= '1';
        elsif Vcount + EXTRA_TOP = maze_y_offset + maze_size_pix_x - 1 or (Vcount =
VSYNC+VBACK_PORCH+VACTIVE) then
          rect_obj_v <= '0';
               end if;
      end if;
    obj_tile_y <= to_integer((Vcount-maze_y_offset+EXTRA_TOP) srl 5);
    end if;
  end process ObjVGen;

  ObjTileInfo : process (clk)
  begin
       if rising_edge(clk) then
      if reset_n = '0' then
               which_obj_tile <= 0;
               obj_x <= 0;
               obj_y <= 0;
          else
               -- take first bit of virtual tile description to get real tile #
               which_obj_tile <=
to_integer(VIRTUAL_OBJ_TILES(to_integer(obj_desc(obj_tile_y)(obj_tile_x)))(8 downto 8));

               -- take next two to get rotation and set indices to appropriate values
               if VIRTUAL_OBJ_TILES(to_integer(obj_desc(obj_tile_y)(obj_tile_x)))(7
downto 6) = "00" then
                       obj_x <= to_integer(count_x);
                       obj_y <= to_integer(count_y);
               elsif VIRTUAL_OBJ_TILES(to_integer(obj_desc(obj_tile_y)(obj_tile_x)))(7
downto 6) = "01" then
                       obj_x <= to_integer(count_y);
                       obj_y <= to_integer(TILE_SIZE - 1 - count_x);
               elsif VIRTUAL_OBJ_TILES(to_integer(obj_desc(obj_tile_y)(obj_tile_x)))(7
downto 6) = "10" then
                       obj_x <= to_integer(TILE_SIZE - 1 - count_x);
                       obj_y <= to_integer(TILE_SIZE - 1 - count_y);
```

```
               elsif VIRTUAL_OBJ_TILES(to_integer(obj_desc(obj_tile_y)(obj_tile_x)))(7
downto 6) = "11" then
                       obj_x <= to_integer(TILE_SIZE - 1 - count_y);
                       obj_y <= to_integer(count_x);
               end if;

               -- take the next 2 bits to get color info
               if VIRTUAL_OBJ_TILES(to_integer(obj_desc(obj_tile_y)(obj_tile_x)))(5
downto 4) = "00" then
                       tmp_vga_r <= "1111111111";
                       tmp_vga_g <= "0000000000";
                       tmp_vga_b <= "0000000000";
               elsif VIRTUAL_OBJ_TILES(to_integer(obj_desc(obj_tile_y)(obj_tile_x)))(5
downto 4) = "01" then
                       tmp_vga_r <= "0000000000";
                       tmp_vga_g <= "1111111111";
                       tmp_vga_b <= "0000000000";
               elsif VIRTUAL_OBJ_TILES(to_integer(obj_desc(obj_tile_y)(obj_tile_x)))(5
downto 4) = "10" then
                       tmp_vga_r <= "0000000000";
                       tmp_vga_g <= "0000000000";
                       tmp_vga_b <= "1111111111";
               else
                       tmp_vga_r <= "0000000000";
                       tmp_vga_g <= "0000000000";
                       tmp_vga_b <= "0000000000";
               end if;

               can_show_cur_color <=
which_keys(to_integer(VIRTUAL_OBJ_TILES(to_integer(obj_desc(obj_tile_y)(obj_tile_x)))(5
downto 4)));

               -- check if this is the empty tile
               if obj_desc(obj_tile_y)(obj_tile_x) = NO_OBJ_TILE then
                       valid_obj_tile <= '0';
               else
                       valid_obj_tile <= '1';
               end if;

       end if;
  end if;
end process ObjTileInfo;

-----------------------------------------------------------------
-- VGA Controller signals for character tiles
-----------------------------------------------------------------
  CharHGen : process (clk)
  begin
    if rising_edge(clk) then
      if reset_n = '0' or Hcount + EXTRA_LEFT = cur_xpos_pix then
        rect_char_h <= '1';
      elsif Hcount + EXTRA_LEFT  = cur_xpos_pix + TILE_SIZE or (Hcount =
HSYNC+HBACK_PORCH+HACTIVE) then
        rect_char_h <= '0';
      end if;
       char_x <= to_integer(cur_xpos_pix - EXTRA_LEFT - HCount);     --- watch out! this
may go below 0 and we are dealingwith unsigned numbers
    end if;
  end process CharHGen;

  CharVGen : process (clk)
  begin
    if rising_edge(clk) then
      if reset_n = '0' then
        rect_char_v <= '0';
      elsif EndOfLine = '1' then
        if Vcount + EXTRA_TOP = cur_ypos_pix  then
          rect_char_v <= '1';
        elsif Vcount + EXTRA_TOP = cur_ypos_pix + TILE_SIZE or (Vcount =
VSYNC+VBACK_PORCH+VACTIVE) then
          rect_char_v <= '0';
```

```
              end if;
      end if;
    char_y <= to_integer(Vcount - cur_ypos_pix + EXTRA_TOP);
    end if;
  end process CharVGen;

  ----------------------------------------
  -- deal with window
  ----------------------------------------
  WindowGen : process (clk)
  begin
    if rising_edge(clk) then
      if reset_n = '0' then
        is_on_window <= '0';
              is_in_window <= '0';
      else
              if ( (Hcount = window_x_center - TILE_SIZE * half_window_size_x or Hcount
= window_x_center + TILE_SIZE * half_window_size_x) and
                  (Vcount >= window_y_center - TILE_SIZE * half_window_size_y and Vcount
<= window_y_center + TILE_SIZE * half_window_size_y) ) or
                  ( (Vcount = window_y_center - TILE_SIZE * half_window_size_y or Vcount
= window_y_center + TILE_SIZE * half_window_size_y) and
                  (Hcount >= window_x_center - TILE_SIZE * half_window_size_x and Hcount
<= window_x_center + TILE_SIZE * half_window_size_x) ) then
                              is_on_window <= '1';
              else
                              is_on_window <= '0';
              end if;

              if Hcount >= window_x_center - TILE_SIZE * half_window_size_x and Hcount
<= window_x_center + TILE_SIZE * half_window_size_x and
                  Vcount >= window_y_center - TILE_SIZE * half_window_size_y and Vcount
<= window_y_center + TILE_SIZE * half_window_size_y then
                              is_in_window <= '1';
              else
                              is_in_window <= '0';
              end if;

     end if;

    end if;
  end process WindowGen;

  draw_char <= rect_char_h and rect_char_v and
CHAR_TILES(to_integer(which_char_tile))(char_y)(char_x) and is_in_window;
  draw_obj  <= rect_obj_h and rect_obj_v and OBJ_TILES(which_obj_tile)(obj_y)(obj_x) and
can_show_cur_color and valid_obj_tile and is_in_window;
  draw_goal <= rect_goal_h and rect_goal_v and
GOAL_TILE(to_integer(count_y))(to_integer(count_x)) and is_in_window;
  draw_maze <= rect_maze_h and rect_maze_v and
MAZE_TILES(which_maze_tile)(maze_y)(maze_x) and valid_maze_tile and is_in_window;

  -- Registered video signals going to the video DAC
  VideoOut: process (clk, reset_n)
  begin
    if reset_n = '0' then
      VGA_R <= "0000000000";
      VGA_G <= "0000000000";
      VGA_B <= "0000000000";
    elsif rising_edge(clk) then
         if is_on_window = '1' then
           VGA_R <= "1111111111";
        VGA_G <= "1111111111";
        VGA_B <= "1111111111";
         elsif draw_char = '1' then
           VGA_R <= "1111111111";
        VGA_G <= "1111111111";
        VGA_B <= "0000000000";
         elsif draw_obj = '1' then
        VGA_R <= tmp_vga_r;
        VGA_G <= tmp_vga_g;
```

```
        VGA_B <= tmp_vga_b;
          elsif draw_goal = '1' then
        VGA_R <= "0000000000";
        VGA_G <= "1111111111";
        VGA_B <= "1111111111";
          elsif draw_maze = '1' then
                VGA_R <= "1111111111";
        VGA_G <= "1111111111";
        VGA_B <= "1111111111";
      elsif vga_hblank = '0' and vga_vblank ='0' then
        VGA_R <= "0000000000";
        VGA_G <= "0000000000";
        VGA_B <= "0000000000";
      else
        VGA_R <= "0000000000";
        VGA_G <= "0000000000";
        VGA_B <= "0000000000";
      end if;
    end if;
  end process VideoOut;

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


-----------------------------------------------------------------
-- all communications with software from here
-----------------------------------------------------------------
GetData: process (clk)
  begin
       if rising_edge(clk) then
                      if chipselect = '1' then

                          if write = '1' then
                                    if address = "0000000000000000" then
                                           tmp_maze_x_offset <= writedata;
                                    elsif address = "0000000000000010" then
                                           tmp_maze_y_offset <= writedata;
                                    elsif address = "0000000000000100" then
                                           tmp_cur_xpos_pix <= writedata;
                                    elsif address = "0000000000000110" then
                                           tmp_cur_ypos_pix <= writedata;
                                    elsif address = "0000000000001000" then
                                           end_xpos_maze <= writedata;
                                    elsif address = "0000000000001010" then
                                           end_ypos_maze <= writedata;
                                    elsif address = "0000000000001100" then
                                           half_window_size_x <= writedata(15 downto
8);
                                           half_window_size_y <= writedata(7 downto 0);
                                    elsif address = "0000000000001110" then
                                           which_keys <= writedata;

                                    elsif address = "0000000000010000" then
                                           maze_desc(0)(0) <= writedata(15 downto 12);
                                           maze_desc(0)(1) <= writedata(11 downto 8);
                                           maze_desc(0)(2) <= writedata(7 downto 4);
                                           maze_desc(0)(3) <= writedata(3 downto 0);
                                    elsif address = "0000000000010010" then
                                           maze_desc(0)(4) <= writedata(15 downto 12);
                                           maze_desc(0)(5) <= writedata(11 downto 8);
                                           maze_desc(0)(6) <= writedata(7 downto 4);
                                           maze_desc(0)(7) <= writedata(3 downto 0);
                                    elsif address = "0000000000010100" then
                                           maze_desc(0)(8) <= writedata(15 downto 12);
                                           maze_desc(0)(9) <= writedata(11 downto 8);
                                           maze_desc(0)(10) <= writedata(7 downto 4);
                                           maze_desc(0)(11) <= writedata(3 downto 0);
```

```
elsif address = "0000000000010110" then
       maze_desc(0)(12) <= writedata(15 downto 12);
       maze_desc(0)(13) <= writedata(11 downto 8);
       maze_desc(0)(14) <= writedata(7 downto 4);
       maze_desc(0)(15) <= writedata(3 downto 0);

-- second row of maze
elsif address = "0000000000011000" then
       maze_desc(1)(0) <= writedata(15 downto 12);
       maze_desc(1)(1) <= writedata(11 downto 8);
       maze_desc(1)(2) <= writedata(7 downto 4);
       maze_desc(1)(3) <= writedata(3 downto 0);
elsif address = "0000000000011010" then
       maze_desc(1)(4) <= writedata(15 downto 12);
       maze_desc(1)(5) <= writedata(11 downto 8);
       maze_desc(1)(6) <= writedata(7 downto 4);
       maze_desc(1)(7) <= writedata(3 downto 0);
elsif address = "0000000000011100" then
       maze_desc(1)(8) <= writedata(15 downto 12);
       maze_desc(1)(9) <= writedata(11 downto 8);
       maze_desc(1)(10) <= writedata(7 downto 4);
       maze_desc(1)(11) <= writedata(3 downto 0);
elsif address = "0000000000011110" then
       maze_desc(1)(12) <= writedata(15 downto 12);
       maze_desc(1)(13) <= writedata(11 downto 8);
       maze_desc(1)(14) <= writedata(7 downto 4);
       maze_desc(1)(15) <= writedata(3 downto 0);

-- third row of maze
elsif address = "0000000000100000" then
       maze_desc(2)(0) <= writedata(15 downto 12);
       maze_desc(2)(1) <= writedata(11 downto 8);
       maze_desc(2)(2) <= writedata(7 downto 4);
       maze_desc(2)(3) <= writedata(3 downto 0);
elsif address = "0000000000100010" then
       maze_desc(2)(4) <= writedata(15 downto 12);
       maze_desc(2)(5) <= writedata(11 downto 8);
       maze_desc(2)(6) <= writedata(7 downto 4);
       maze_desc(2)(7) <= writedata(3 downto 0);
elsif address = "0000000000100100" then
       maze_desc(2)(8) <= writedata(15 downto 12);
       maze_desc(2)(9) <= writedata(11 downto 8);
       maze_desc(2)(10) <= writedata(7 downto 4);
       maze_desc(2)(11) <= writedata(3 downto 0);
elsif address = "0000000000100110" then
       maze_desc(2)(12) <= writedata(15 downto 12);
       maze_desc(2)(13) <= writedata(11 downto 8);
       maze_desc(2)(14) <= writedata(7 downto 4);
       maze_desc(2)(15) <= writedata(3 downto 0);

-- fourth row of maze
elsif address = "0000000000101000" then
       maze_desc(3)(0) <= writedata(15 downto 12);
       maze_desc(3)(1) <= writedata(11 downto 8);
       maze_desc(3)(2) <= writedata(7 downto 4);
       maze_desc(3)(3) <= writedata(3 downto 0);
elsif address = "0000000000101010" then
       maze_desc(3)(4) <= writedata(15 downto 12);
       maze_desc(3)(5) <= writedata(11 downto 8);
       maze_desc(3)(6) <= writedata(7 downto 4);
       maze_desc(3)(7) <= writedata(3 downto 0);
elsif address = "0000000000101100" then
       maze_desc(3)(8) <= writedata(15 downto 12);
       maze_desc(3)(9) <= writedata(11 downto 8);
       maze_desc(3)(10) <= writedata(7 downto 4);
       maze_desc(3)(11) <= writedata(3 downto 0);
elsif address = "0000000000101110" then
       maze_desc(3)(12) <= writedata(15 downto 12);
       maze_desc(3)(13) <= writedata(11 downto 8);
       maze_desc(3)(14) <= writedata(7 downto 4);
       maze_desc(3)(15) <= writedata(3 downto 0);
```

```
-- fifth row of maze
elsif address = "0000000000110000" then
       maze_desc(4)(0) <= writedata(15 downto 12);
       maze_desc(4)(1) <= writedata(11 downto 8);
       maze_desc(4)(2) <= writedata(7 downto 4);
       maze_desc(4)(3) <= writedata(3 downto 0);
elsif address = "0000000000110010" then
       maze_desc(4)(4) <= writedata(15 downto 12);
       maze_desc(4)(5) <= writedata(11 downto 8);
       maze_desc(4)(6) <= writedata(7 downto 4);
       maze_desc(4)(7) <= writedata(3 downto 0);
elsif address = "0000000000110100" then
       maze_desc(4)(8) <= writedata(15 downto 12);
       maze_desc(4)(9) <= writedata(11 downto 8);
       maze_desc(4)(10) <= writedata(7 downto 4);
       maze_desc(4)(11) <= writedata(3 downto 0);
elsif address = "0000000000110110" then
       maze_desc(4)(12) <= writedata(15 downto 12);
       maze_desc(4)(13) <= writedata(11 downto 8);
       maze_desc(4)(14) <= writedata(7 downto 4);
       maze_desc(4)(15) <= writedata(3 downto 0);

-- sixth row of maze
elsif address = "0000000000111000" then
       maze_desc(5)(0) <= writedata(15 downto 12);
       maze_desc(5)(1) <= writedata(11 downto 8);
       maze_desc(5)(2) <= writedata(7 downto 4);
       maze_desc(5)(3) <= writedata(3 downto 0);
elsif address = "0000000000111010" then
       maze_desc(5)(4) <= writedata(15 downto 12);
       maze_desc(5)(5) <= writedata(11 downto 8);
       maze_desc(5)(6) <= writedata(7 downto 4);
       maze_desc(5)(7) <= writedata(3 downto 0);
elsif address = "0000000000111100" then
       maze_desc(5)(8) <= writedata(15 downto 12);
       maze_desc(5)(9) <= writedata(11 downto 8);
       maze_desc(5)(10) <= writedata(7 downto 4);
       maze_desc(5)(11) <= writedata(3 downto 0);
elsif address = "0000000000111110" then
       maze_desc(5)(12) <= writedata(15 downto 12);
       maze_desc(5)(13) <= writedata(11 downto 8);
       maze_desc(5)(14) <= writedata(7 downto 4);
       maze_desc(5)(15) <= writedata(3 downto 0);

-- seventh row of maze
elsif address = "0000000001000000" then
       maze_desc(6)(0) <= writedata(15 downto 12);
       maze_desc(6)(1) <= writedata(11 downto 8);
       maze_desc(6)(2) <= writedata(7 downto 4);
       maze_desc(6)(3) <= writedata(3 downto 0);
elsif address = "0000000001000010" then
       maze_desc(6)(4) <= writedata(15 downto 12);
       maze_desc(6)(5) <= writedata(11 downto 8);
       maze_desc(6)(6) <= writedata(7 downto 4);
       maze_desc(6)(7) <= writedata(3 downto 0);
elsif address = "0000000001000100" then
       maze_desc(6)(8) <= writedata(15 downto 12);
       maze_desc(6)(9) <= writedata(11 downto 8);
       maze_desc(6)(10) <= writedata(7 downto 4);
       maze_desc(6)(11) <= writedata(3 downto 0);
elsif address = "0000000001000110" then
       maze_desc(6)(12) <= writedata(15 downto 12);
       maze_desc(6)(13) <= writedata(11 downto 8);
       maze_desc(6)(14) <= writedata(7 downto 4);
       maze_desc(6)(15) <= writedata(3 downto 0);

-- eigth row of maze
elsif address = "0000000001001000" then
       maze_desc(7)(0) <= writedata(15 downto 12);
       maze_desc(7)(1) <= writedata(11 downto 8);
```

```
        maze_desc(7)(2) <= writedata(7 downto 4);
        maze_desc(7)(3) <= writedata(3 downto 0);
elsif address = "0000000001001010" then
        maze_desc(7)(4) <= writedata(15 downto 12);
        maze_desc(7)(5) <= writedata(11 downto 8);
        maze_desc(7)(6) <= writedata(7 downto 4);
        maze_desc(7)(7) <= writedata(3 downto 0);
elsif address = "0000000001001100" then
        maze_desc(7)(8) <= writedata(15 downto 12);
        maze_desc(7)(9) <= writedata(11 downto 8);
        maze_desc(7)(10) <= writedata(7 downto 4);
        maze_desc(7)(11) <= writedata(3 downto 0);
elsif address = "0000000001001110" then
        maze_desc(7)(12) <= writedata(15 downto 12);
        maze_desc(7)(13) <= writedata(11 downto 8);
        maze_desc(7)(14) <= writedata(7 downto 4);
        maze_desc(7)(15) <= writedata(3 downto 0);

-- ninth row of maze
elsif address = "0000000001010000" then
        maze_desc(8)(0) <= writedata(15 downto 12);
        maze_desc(8)(1) <= writedata(11 downto 8);
        maze_desc(8)(2) <= writedata(7 downto 4);
        maze_desc(8)(3) <= writedata(3 downto 0);
elsif address = "0000000001010010" then
        maze_desc(8)(4) <= writedata(15 downto 12);
        maze_desc(8)(5) <= writedata(11 downto 8);
        maze_desc(8)(6) <= writedata(7 downto 4);
        maze_desc(8)(7) <= writedata(3 downto 0);
elsif address = "0000000001010100" then
        maze_desc(8)(8) <= writedata(15 downto 12);
        maze_desc(8)(9) <= writedata(11 downto 8);
        maze_desc(8)(10) <= writedata(7 downto 4);
        maze_desc(8)(11) <= writedata(3 downto 0);
elsif address = "0000000001010110" then
        maze_desc(8)(12) <= writedata(15 downto 12);
        maze_desc(8)(13) <= writedata(11 downto 8);
        maze_desc(8)(14) <= writedata(7 downto 4);
        maze_desc(8)(15) <= writedata(3 downto 0);

-- tenth row of maze
elsif address = "0000000001011000" then
        maze_desc(9)(0) <= writedata(15 downto 12);
        maze_desc(9)(1) <= writedata(11 downto 8);
        maze_desc(9)(2) <= writedata(7 downto 4);
        maze_desc(9)(3) <= writedata(3 downto 0);
elsif address = "0000000001011010" then
        maze_desc(9)(4) <= writedata(15 downto 12);
        maze_desc(9)(5) <= writedata(11 downto 8);
        maze_desc(9)(6) <= writedata(7 downto 4);
        maze_desc(9)(7) <= writedata(3 downto 0);
elsif address = "0000000001011100" then
        maze_desc(9)(8) <= writedata(15 downto 12);
        maze_desc(9)(9) <= writedata(11 downto 8);
        maze_desc(9)(10) <= writedata(7 downto 4);
        maze_desc(9)(11) <= writedata(3 downto 0);
elsif address = "0000000001011110" then
        maze_desc(9)(12) <= writedata(15 downto 12);
        maze_desc(9)(13) <= writedata(11 downto 8);
        maze_desc(9)(14) <= writedata(7 downto 4);
        maze_desc(9)(15) <= writedata(3 downto 0);

-- eleventh row of maze
elsif address = "0000000001100000" then
        maze_desc(10)(0) <= writedata(15 downto 12);
        maze_desc(10)(1) <= writedata(11 downto 8);
        maze_desc(10)(2) <= writedata(7 downto 4);
        maze_desc(10)(3) <= writedata(3 downto 0);
elsif address = "0000000001100010" then
        maze_desc(10)(4) <= writedata(15 downto 12);
        maze_desc(10)(5) <= writedata(11 downto 8);
```

```
                                        maze_desc(10)(6) <= writedata(7 downto 4);
                                        maze_desc(10)(7) <= writedata(3 downto 0);
                                    elsif address = "0000000001100100" then
                                        maze_desc(10)(8) <= writedata(15 downto 12);
                                        maze_desc(10)(9) <= writedata(11 downto 8);
                                        maze_desc(10)(10) <= writedata(7 downto 4);
                                        maze_desc(10)(11) <= writedata(3 downto 0);
                                    elsif address = "0000000001100110" then
                                        maze_desc(10)(12) <= writedata(15 downto
12);
                                        maze_desc(10)(13) <= writedata(11 downto 8);
                                        maze_desc(10)(14) <= writedata(7 downto 4);
                                        maze_desc(10)(15) <= writedata(3 downto 0);

                                    -- twelfth row of maze
                                    elsif address = "0000000001101000" then
                                        maze_desc(11)(0) <= writedata(15 downto 12);
                                        maze_desc(11)(1) <= writedata(11 downto 8);
                                        maze_desc(11)(2) <= writedata(7 downto 4);
                                        maze_desc(11)(3) <= writedata(3 downto 0);
                                    elsif address = "0000000001101010" then
                                        maze_desc(11)(4) <= writedata(15 downto 12);
                                        maze_desc(11)(5) <= writedata(11 downto 8);
                                        maze_desc(11)(6) <= writedata(7 downto 4);
                                        maze_desc(11)(7) <= writedata(3 downto 0);
                                    elsif address = "0000000001101100" then
                                        maze_desc(11)(8) <= writedata(15 downto 12);
                                        maze_desc(11)(9) <= writedata(11 downto 8);
                                        maze_desc(11)(10) <= writedata(7 downto 4);
                                        maze_desc(11)(11) <= writedata(3 downto 0);
                                    elsif address = "0000000001101110" then
                                        maze_desc(11)(12) <= writedata(15 downto
12);
                                        maze_desc(11)(13) <= writedata(11 downto 8);
                                        maze_desc(11)(14) <= writedata(7 downto 4);
                                        maze_desc(11)(15) <= writedata(3 downto 0);


            --                          thirteenth row of maze
                                    elsif address = "0000000001110000" then
                                        maze_desc(12)(0) <= writedata(15 downto 12);
                                        maze_desc(12)(1) <= writedata(11 downto 8);
                                        maze_desc(12)(2) <= writedata(7 downto 4);
                                        maze_desc(12)(3) <= writedata(3 downto 0);
                                    elsif address = "0000000001110010" then
                                        maze_desc(12)(4) <= writedata(15 downto 12);
                                        maze_desc(12)(5) <= writedata(11 downto 8);
                                        maze_desc(12)(6) <= writedata(7 downto 4);
                                        maze_desc(12)(7) <= writedata(3 downto 0);
                                    elsif address = "0000000001110100" then
                                        maze_desc(12)(8) <= writedata(15 downto 12);
                                        maze_desc(12)(9) <= writedata(11 downto 8);
                                        maze_desc(12)(10) <= writedata(7 downto 4);
                                        maze_desc(12)(11) <= writedata(3 downto 0);
                                    elsif address = "0000000001110110" then
                                        maze_desc(12)(12) <= writedata(15 downto
12);
                                        maze_desc(12)(13) <= writedata(11 downto 8);
                                        maze_desc(12)(14) <= writedata(7 downto 4);
                                        maze_desc(12)(15) <= writedata(3 downto 0);


            --                          fourteenth row of maze
                                    elsif address = "0000000001111000" then
                                        maze_desc(13)(0) <= writedata(15 downto 12);
                                        maze_desc(13)(1) <= writedata(11 downto 8);
                                        maze_desc(13)(2) <= writedata(7 downto 4);
                                        maze_desc(13)(3) <= writedata(3 downto 0);
                                    elsif address = "0000000001111010" then
                                        maze_desc(13)(4) <= writedata(15 downto 12);
                                        maze_desc(13)(5) <= writedata(11 downto 8);
```

```
                                        maze_desc(13)(6) <= writedata(7 downto 4);
                                        maze_desc(13)(7) <= writedata(3 downto 0);
                                    elsif address = "0000000001111100" then
                                        maze_desc(13)(8) <= writedata(15 downto 12);
                                        maze_desc(13)(9) <= writedata(11 downto 8);
                                        maze_desc(13)(10) <= writedata(7 downto 4);
                                        maze_desc(13)(11) <= writedata(3 downto 0);
                                    elsif address = "0000000001111110" then
                                        maze_desc(13)(12) <= writedata(15 downto
12);
                                        maze_desc(13)(13) <= writedata(11 downto 8);
                                        maze_desc(13)(14) <= writedata(7 downto 4);
                                        maze_desc(13)(15) <= writedata(3 downto 0);


        --                               ffiteenth row of maze
                                    elsif address = "0000000010000000" then
                                        maze_desc(14)(0) <= writedata(15 downto 12);
                                        maze_desc(14)(1) <= writedata(11 downto 8);
                                        maze_desc(14)(2) <= writedata(7 downto 4);
                                        maze_desc(14)(3) <= writedata(3 downto 0);
                                    elsif address = "0000000010000010" then
                                        maze_desc(14)(4) <= writedata(15 downto 12);
                                        maze_desc(14)(5) <= writedata(11 downto 8);
                                        maze_desc(14)(6) <= writedata(7 downto 4);
                                        maze_desc(14)(7) <= writedata(3 downto 0);
                                    elsif address = "0000000010000100" then
                                        maze_desc(14)(8) <= writedata(15 downto 12);
                                        maze_desc(14)(9) <= writedata(11 downto 8);
                                        maze_desc(14)(10) <= writedata(7 downto 4);
                                        maze_desc(14)(11) <= writedata(3 downto 0);
                                    elsif address = "0000000010000110" then
                                        maze_desc(14)(12) <= writedata(15 downto
12);
                                        maze_desc(14)(13) <= writedata(11 downto 8);
                                        maze_desc(14)(14) <= writedata(7 downto 4);
                                        maze_desc(14)(15) <= writedata(3 downto 0);


        --                              sixteenth row of maze
                                    elsif address = "0000000010001000" then
                                        maze_desc(15)(0) <= writedata(15 downto 12);
                                        maze_desc(15)(1) <= writedata(11 downto 8);
                                        maze_desc(15)(2) <= writedata(7 downto 4);
                                        maze_desc(15)(3) <= writedata(3 downto 0);
                                    elsif address = "0000000010001010" then
                                        maze_desc(15)(4) <= writedata(15 downto 12);
                                        maze_desc(15)(5) <= writedata(11 downto 8);
                                        maze_desc(15)(6) <= writedata(7 downto 4);
                                        maze_desc(15)(7) <= writedata(3 downto 0);
                                    elsif address = "0000000010001100" then
                                        maze_desc(15)(8) <= writedata(15 downto 12);
                                        maze_desc(15)(9) <= writedata(11 downto 8);
                                        maze_desc(15)(10) <= writedata(7 downto 4);
                                        maze_desc(15)(11) <= writedata(3 downto 0);
                                    elsif address = "0000000010001110" then
                                        maze_desc(15)(12) <= writedata(15 downto
12);
                                        maze_desc(15)(13) <= writedata(11 downto 8);
                                        maze_desc(15)(14) <= writedata(7 downto 4);
                                        maze_desc(15)(15) <= writedata(3 downto 0);



                                    -- top row of obj
                                    elsif address = "0000000010010010" then
                                        obj_desc(0)(0) <= writedata(15 downto 12);
                                        obj_desc(0)(1) <= writedata(11 downto 8);
                                        obj_desc(0)(2) <= writedata(7 downto 4);
                                        obj_desc(0)(3) <= writedata(3 downto 0);
                                    elsif address = "0000000010010100" then
```

```
        obj_desc(0)(4) <= writedata(15 downto 12);
        obj_desc(0)(5) <= writedata(11 downto 8);
        obj_desc(0)(6) <= writedata(7 downto 4);
        obj_desc(0)(7) <= writedata(3 downto 0);
    elsif address = "0000000010010110" then
        obj_desc(0)(8) <= writedata(15 downto 12);
        obj_desc(0)(9) <= writedata(11 downto 8);
        obj_desc(0)(10) <= writedata(7 downto 4);
        obj_desc(0)(11) <= writedata(3 downto 0);
    elsif address = "0000000010011000" then
        obj_desc(0)(12) <= writedata(15 downto 12);
        obj_desc(0)(13) <= writedata(11 downto 8);
        obj_desc(0)(14) <= writedata(7 downto 4);
        obj_desc(0)(15) <= writedata(3 downto 0);

    -- second row of obj
    elsif address = "0000000010011010" then
        obj_desc(1)(0) <= writedata(15 downto 12);
        obj_desc(1)(1) <= writedata(11 downto 8);
        obj_desc(1)(2) <= writedata(7 downto 4);
        obj_desc(1)(3) <= writedata(3 downto 0);
    elsif address = "0000000010011100" then
        obj_desc(1)(4) <= writedata(15 downto 12);
        obj_desc(1)(5) <= writedata(11 downto 8);
        obj_desc(1)(6) <= writedata(7 downto 4);
        obj_desc(1)(7) <= writedata(3 downto 0);
    elsif address = "0000000010011110" then
        obj_desc(1)(8) <= writedata(15 downto 12);
        obj_desc(1)(9) <= writedata(11 downto 8);
        obj_desc(1)(10) <= writedata(7 downto 4);
        obj_desc(1)(11) <= writedata(3 downto 0);
    elsif address = "0000000010100000" then
        obj_desc(1)(12) <= writedata(15 downto 12);
        obj_desc(1)(13) <= writedata(11 downto 8);
        obj_desc(1)(14) <= writedata(7 downto 4);
        obj_desc(1)(15) <= writedata(3 downto 0);

    -- third row of obj
    elsif address = "0000000010100010" then
        obj_desc(2)(0) <= writedata(15 downto 12);
        obj_desc(2)(1) <= writedata(11 downto 8);
        obj_desc(2)(2) <= writedata(7 downto 4);
        obj_desc(2)(3) <= writedata(3 downto 0);
    elsif address = "0000000010100100" then
        obj_desc(2)(4) <= writedata(15 downto 12);
        obj_desc(2)(5) <= writedata(11 downto 8);
        obj_desc(2)(6) <= writedata(7 downto 4);
        obj_desc(2)(7) <= writedata(3 downto 0);
    elsif address = "0000000010100110" then
        obj_desc(2)(8) <= writedata(15 downto 12);
        obj_desc(2)(9) <= writedata(11 downto 8);
        obj_desc(2)(10) <= writedata(7 downto 4);
        obj_desc(2)(11) <= writedata(3 downto 0);
    elsif address = "0000000010101000" then
        obj_desc(2)(12) <= writedata(15 downto 12);
        obj_desc(2)(13) <= writedata(11 downto 8);
        obj_desc(2)(14) <= writedata(7 downto 4);
        obj_desc(2)(15) <= writedata(3 downto 0);

    -- fourth row of obj
    elsif address = "0000000010101010" then
        obj_desc(3)(0) <= writedata(15 downto 12);
        obj_desc(3)(1) <= writedata(11 downto 8);
        obj_desc(3)(2) <= writedata(7 downto 4);
        obj_desc(3)(3) <= writedata(3 downto 0);
    elsif address = "0000000010101100" then
        obj_desc(3)(4) <= writedata(15 downto 12);
        obj_desc(3)(5) <= writedata(11 downto 8);
        obj_desc(3)(6) <= writedata(7 downto 4);
        obj_desc(3)(7) <= writedata(3 downto 0);
    elsif address = "0000000010101110" then
```

```
        obj_desc(3)(8) <= writedata(15 downto 12);
        obj_desc(3)(9) <= writedata(11 downto 8);
        obj_desc(3)(10) <= writedata(7 downto 4);
        obj_desc(3)(11) <= writedata(3 downto 0);
    elsif address = "0000000010110000" then
        obj_desc(3)(12) <= writedata(15 downto 12);
        obj_desc(3)(13) <= writedata(11 downto 8);
        obj_desc(3)(14) <= writedata(7 downto 4);
        obj_desc(3)(15) <= writedata(3 downto 0);

    -- fifth row of obj
    elsif address = "0000000010110010" then
        obj_desc(4)(0) <= writedata(15 downto 12);
        obj_desc(4)(1) <= writedata(11 downto 8);
        obj_desc(4)(2) <= writedata(7 downto 4);
        obj_desc(4)(3) <= writedata(3 downto 0);
    elsif address = "0000000010110100" then
        obj_desc(4)(4) <= writedata(15 downto 12);
        obj_desc(4)(5) <= writedata(11 downto 8);
        obj_desc(4)(6) <= writedata(7 downto 4);
        obj_desc(4)(7) <= writedata(3 downto 0);
    elsif address = "0000000010110110" then
        obj_desc(4)(8) <= writedata(15 downto 12);
        obj_desc(4)(9) <= writedata(11 downto 8);
        obj_desc(4)(10) <= writedata(7 downto 4);
        obj_desc(4)(11) <= writedata(3 downto 0);
    elsif address = "0000000010111000" then
        obj_desc(4)(12) <= writedata(15 downto 12);
        obj_desc(4)(13) <= writedata(11 downto 8);
        obj_desc(4)(14) <= writedata(7 downto 4);
        obj_desc(4)(15) <= writedata(3 downto 0);

    -- sixth row of obj
    elsif address = "0000000010111010" then
        obj_desc(5)(0) <= writedata(15 downto 12);
        obj_desc(5)(1) <= writedata(11 downto 8);
        obj_desc(5)(2) <= writedata(7 downto 4);
        obj_desc(5)(3) <= writedata(3 downto 0);
    elsif address = "0000000010111100" then
        obj_desc(5)(4) <= writedata(15 downto 12);
        obj_desc(5)(5) <= writedata(11 downto 8);
        obj_desc(5)(6) <= writedata(7 downto 4);
        obj_desc(5)(7) <= writedata(3 downto 0);
    elsif address = "0000000010111110" then
        obj_desc(5)(8) <= writedata(15 downto 12);
        obj_desc(5)(9) <= writedata(11 downto 8);
        obj_desc(5)(10) <= writedata(7 downto 4);
        obj_desc(5)(11) <= writedata(3 downto 0);
    elsif address = "0000000011000000" then
        obj_desc(5)(12) <= writedata(15 downto 12);
        obj_desc(5)(13) <= writedata(11 downto 8);
        obj_desc(5)(14) <= writedata(7 downto 4);
        obj_desc(5)(15) <= writedata(3 downto 0);

    -- seventh row of obj
    elsif address = "0000000011000010" then
        obj_desc(6)(0) <= writedata(15 downto 12);
        obj_desc(6)(1) <= writedata(11 downto 8);
        obj_desc(6)(2) <= writedata(7 downto 4);
        obj_desc(6)(3) <= writedata(3 downto 0);
    elsif address = "0000000011000100" then
        obj_desc(6)(4) <= writedata(15 downto 12);
        obj_desc(6)(5) <= writedata(11 downto 8);
        obj_desc(6)(6) <= writedata(7 downto 4);
        obj_desc(6)(7) <= writedata(3 downto 0);
    elsif address = "0000000011000110" then
        obj_desc(6)(8) <= writedata(15 downto 12);
        obj_desc(6)(9) <= writedata(11 downto 8);
        obj_desc(6)(10) <= writedata(7 downto 4);
        obj_desc(6)(11) <= writedata(3 downto 0);
    elsif address = "0000000011001000" then
```

```
        obj_desc(6)(12) <= writedata(15 downto 12);
        obj_desc(6)(13) <= writedata(11 downto 8);
        obj_desc(6)(14) <= writedata(7 downto 4);
        obj_desc(6)(15) <= writedata(3 downto 0);

    -- eigth row of obj
    elsif address = "0000000011001010" then
        obj_desc(7)(0) <= writedata(15 downto 12);
        obj_desc(7)(1) <= writedata(11 downto 8);
        obj_desc(7)(2) <= writedata(7 downto 4);
        obj_desc(7)(3) <= writedata(3 downto 0);
    elsif address = "0000000011001100" then
        obj_desc(7)(4) <= writedata(15 downto 12);
        obj_desc(7)(5) <= writedata(11 downto 8);
        obj_desc(7)(6) <= writedata(7 downto 4);
        obj_desc(7)(7) <= writedata(3 downto 0);
    elsif address = "0000000011001110" then
        obj_desc(7)(8) <= writedata(15 downto 12);
        obj_desc(7)(9) <= writedata(11 downto 8);
        obj_desc(7)(10) <= writedata(7 downto 4);
        obj_desc(7)(11) <= writedata(3 downto 0);
    elsif address = "0000000011010000" then
        obj_desc(7)(12) <= writedata(15 downto 12);
        obj_desc(7)(13) <= writedata(11 downto 8);
        obj_desc(7)(14) <= writedata(7 downto 4);
        obj_desc(7)(15) <= writedata(3 downto 0);

    -- ninth row of obj
    elsif address = "0000000011010010" then
        obj_desc(8)(0) <= writedata(15 downto 12);
        obj_desc(8)(1) <= writedata(11 downto 8);
        obj_desc(8)(2) <= writedata(7 downto 4);
        obj_desc(8)(3) <= writedata(3 downto 0);
    elsif address = "0000000011010100" then
        obj_desc(8)(4) <= writedata(15 downto 12);
        obj_desc(8)(5) <= writedata(11 downto 8);
        obj_desc(8)(6) <= writedata(7 downto 4);
        obj_desc(8)(7) <= writedata(3 downto 0);
    elsif address = "0000000011010110" then
        obj_desc(8)(8) <= writedata(15 downto 12);
        obj_desc(8)(9) <= writedata(11 downto 8);
        obj_desc(8)(10) <= writedata(7 downto 4);
        obj_desc(8)(11) <= writedata(3 downto 0);
    elsif address = "0000000011011000" then
        obj_desc(8)(12) <= writedata(15 downto 12);
        obj_desc(8)(13) <= writedata(11 downto 8);
        obj_desc(8)(14) <= writedata(7 downto 4);
        obj_desc(8)(15) <= writedata(3 downto 0);

    -- tenth row of obj
    elsif address = "0000000011011010" then
        obj_desc(9)(0) <= writedata(15 downto 12);
        obj_desc(9)(1) <= writedata(11 downto 8);
        obj_desc(9)(2) <= writedata(7 downto 4);
        obj_desc(9)(3) <= writedata(3 downto 0);
    elsif address = "0000000011011100" then
        obj_desc(9)(4) <= writedata(15 downto 12);
        obj_desc(9)(5) <= writedata(11 downto 8);
        obj_desc(9)(6) <= writedata(7 downto 4);
        obj_desc(9)(7) <= writedata(3 downto 0);
    elsif address = "0000000011011110" then
        obj_desc(9)(8) <= writedata(15 downto 12);
        obj_desc(9)(9) <= writedata(11 downto 8);
        obj_desc(9)(10) <= writedata(7 downto 4);
        obj_desc(9)(11) <= writedata(3 downto 0);
    elsif address = "0000000011100000" then
        obj_desc(9)(12) <= writedata(15 downto 12);
        obj_desc(9)(13) <= writedata(11 downto 8);
        obj_desc(9)(14) <= writedata(7 downto 4);
        obj_desc(9)(15) <= writedata(3 downto 0);
```

```
                                -- eleventh row of obj
                                elsif address = "0000000011100010" then
                                       obj_desc(10)(0) <= writedata(15 downto 12);
                                       obj_desc(10)(1) <= writedata(11 downto 8);
                                       obj_desc(10)(2) <= writedata(7 downto 4);
                                       obj_desc(10)(3) <= writedata(3 downto 0);
                                elsif address = "0000000011100100" then
                                       obj_desc(10)(4) <= writedata(15 downto 12);
                                       obj_desc(10)(5) <= writedata(11 downto 8);
                                       obj_desc(10)(6) <= writedata(7 downto 4);
                                       obj_desc(10)(7) <= writedata(3 downto 0);
                                elsif address = "0000000011100110" then
                                       obj_desc(10)(8) <= writedata(15 downto 12);
                                       obj_desc(10)(9) <= writedata(11 downto 8);
                                       obj_desc(10)(10) <= writedata(7 downto 4);
                                       obj_desc(10)(11) <= writedata(3 downto 0);
                                elsif address = "0000000011101000" then
                                       obj_desc(10)(12) <= writedata(15 downto 12);
                                       obj_desc(10)(13) <= writedata(11 downto 8);
                                       obj_desc(10)(14) <= writedata(7 downto 4);
                                       obj_desc(10)(15) <= writedata(3 downto 0);

                                -- twelfth row of obj
                                elsif address = "0000000011101010" then
                                       obj_desc(11)(0) <= writedata(15 downto 12);
                                       obj_desc(11)(1) <= writedata(11 downto 8);
                                       obj_desc(11)(2) <= writedata(7 downto 4);
                                       obj_desc(11)(3) <= writedata(3 downto 0);
                                elsif address = "0000000011101100" then
                                       obj_desc(11)(4) <= writedata(15 downto 12);
                                       obj_desc(11)(5) <= writedata(11 downto 8);
                                       obj_desc(11)(6) <= writedata(7 downto 4);
                                       obj_desc(11)(7) <= writedata(3 downto 0);
                                elsif address = "0000000011101110" then
                                       obj_desc(11)(8) <= writedata(15 downto 12);
                                       obj_desc(11)(9) <= writedata(11 downto 8);
                                       obj_desc(11)(10) <= writedata(7 downto 4);
                                       obj_desc(11)(11) <= writedata(3 downto 0);
                                elsif address = "0000000011110000" then
                                       obj_desc(11)(12) <= writedata(15 downto 12);
                                       obj_desc(11)(13) <= writedata(11 downto 8);
                                       obj_desc(11)(14) <= writedata(7 downto 4);
                                       obj_desc(11)(15) <= writedata(3 downto 0);


                        --              thirteenth row of obj
                                elsif address = "0000000011110010" then
                                       obj_desc(12)(0) <= writedata(15 downto 12);
                                       obj_desc(12)(1) <= writedata(11 downto 8);
                                       obj_desc(12)(2) <= writedata(7 downto 4);
                                       obj_desc(12)(3) <= writedata(3 downto 0);
                                elsif address = "0000000011110100" then
                                       obj_desc(12)(4) <= writedata(15 downto 12);
                                       obj_desc(12)(5) <= writedata(11 downto 8);
                                       obj_desc(12)(6) <= writedata(7 downto 4);
                                       obj_desc(12)(7) <= writedata(3 downto 0);
                                elsif address = "0000000011110110" then
                                       obj_desc(12)(8) <= writedata(15 downto 12);
                                       obj_desc(12)(9) <= writedata(11 downto 8);
                                       obj_desc(12)(10) <= writedata(7 downto 4);
                                       obj_desc(12)(11) <= writedata(3 downto 0);
                                elsif address = "0000000011111000" then
                                       obj_desc(12)(12) <= writedata(15 downto 12);
                                       obj_desc(12)(13) <= writedata(11 downto 8);
                                       obj_desc(12)(14) <= writedata(7 downto 4);
                                       obj_desc(12)(15) <= writedata(3 downto 0);


                        --              fourteenth row of obj
                                elsif address = "0000000011111010" then
                                       obj_desc(13)(0) <= writedata(15 downto 12);
```

```
                                        obj_desc(13)(1) <= writedata(11 downto 8);
                                        obj_desc(13)(2) <= writedata(7 downto 4);
                                        obj_desc(13)(3) <= writedata(3 downto 0);
                                elsif address = "0000000011111100" then
                                        obj_desc(13)(4) <= writedata(15 downto 12);
                                        obj_desc(13)(5) <= writedata(11 downto 8);
                                        obj_desc(13)(6) <= writedata(7 downto 4);
                                        obj_desc(13)(7) <= writedata(3 downto 0);
                                elsif address = "0000000011111110" then
                                        obj_desc(13)(8) <= writedata(15 downto 12);
                                        obj_desc(13)(9) <= writedata(11 downto 8);
                                        obj_desc(13)(10) <= writedata(7 downto 4);
                                        obj_desc(13)(11) <= writedata(3 downto 0);
                                elsif address = "0000000100000000" then
                                        obj_desc(13)(12) <= writedata(15 downto 12);
                                        obj_desc(13)(13) <= writedata(11 downto 8);
                                        obj_desc(13)(14) <= writedata(7 downto 4);
                                        obj_desc(13)(15) <= writedata(3 downto 0);


--                                       fifteenth row of obj
                                elsif address = "0000000100000010" then
                                        obj_desc(14)(0) <= writedata(15 downto 12);
                                        obj_desc(14)(1) <= writedata(11 downto 8);
                                        obj_desc(14)(2) <= writedata(7 downto 4);
                                        obj_desc(14)(3) <= writedata(3 downto 0);
                                elsif address = "0000000100000100" then
                                        obj_desc(14)(4) <= writedata(15 downto 12);
                                        obj_desc(14)(5) <= writedata(11 downto 8);
                                        obj_desc(14)(6) <= writedata(7 downto 4);
                                        obj_desc(14)(7) <= writedata(3 downto 0);
                                elsif address = "0000000100000110" then
                                        obj_desc(14)(8) <= writedata(15 downto 12);
                                        obj_desc(14)(9) <= writedata(11 downto 8);
                                        obj_desc(14)(10) <= writedata(7 downto 4);
                                        obj_desc(14)(11) <= writedata(3 downto 0);
                                elsif address = "0000000100001000" then
                                        obj_desc(14)(12) <= writedata(15 downto 12);
                                        obj_desc(14)(13) <= writedata(11 downto 8);
                                        obj_desc(14)(14) <= writedata(7 downto 4);
                                        obj_desc(14)(15) <= writedata(3 downto 0);


--                                      sixteenth row of obj
                                elsif address = "0000000100001010" then
                                        obj_desc(15)(0) <= writedata(15 downto 12);
                                        obj_desc(15)(1) <= writedata(11 downto 8);
                                        obj_desc(15)(2) <= writedata(7 downto 4);
                                        obj_desc(15)(3) <= writedata(3 downto 0);
                                elsif address = "0000000100001100" then
                                        obj_desc(15)(4) <= writedata(15 downto 12);
                                        obj_desc(15)(5) <= writedata(11 downto 8);
                                        obj_desc(15)(6) <= writedata(7 downto 4);
                                        obj_desc(15)(7) <= writedata(3 downto 0);
                                elsif address = "0000000100001110" then
                                        obj_desc(15)(8) <= writedata(15 downto 12);
                                        obj_desc(15)(9) <= writedata(11 downto 8);
                                        obj_desc(15)(10) <= writedata(7 downto 4);
                                        obj_desc(15)(11) <= writedata(3 downto 0);
                                elsif address = "0000000100010000" then
                                        obj_desc(15)(12) <= writedata(15 downto 12);
                                        obj_desc(15)(13) <= writedata(11 downto 8);
                                        obj_desc(15)(14) <= writedata(7 downto 4);
                                        obj_desc(15)(15) <= writedata(3 downto 0);

                                elsif address = "0000000100010010" then
                                        tmp_char_tile <= writedata;
                                else
                                        foo <= writedata;
                                end if;
                        elsif read = '1' then
```

```
if address = "0000000000000000" then
       readdata <= maze_x_offset;
elsif address = "0000000000000010" then
       readdata <= maze_y_offset;
elsif address = "0000000000000100" then
       readdata <= cur_xpos_pix;
elsif address = "0000000000000110" then
       readdata <= cur_ypos_pix;
elsif address = "0000000000001000" then
       readdata <= end_xpos_maze;
elsif address = "0000000000001010" then
       readdata <= end_ypos_maze;
elsif address = "0000000000001100" then
       readdata(15 downto 8) <= half_window_size_x;
       readdata(7 downto 0) <= half_window_size_y;
elsif address = "0000000000001110" then
       readdata <= which_keys;
elsif address = "0000000100010010" then
       readdata <= which_char_tile;

elsif address = "0000000000010000" then
       readdata(15 downto 12) <= maze_desc(0)(0);
       readdata(11 downto 8) <= maze_desc(0)(1);
       readdata(7 downto 4) <= maze_desc(0)(2);
       readdata(3 downto 0) <= maze_desc(0)(3);
elsif address = "0000000000010010" then
       readdata(15 downto 12) <= maze_desc(0)(4);
       readdata(11 downto 8) <= maze_desc(0)(5);
       readdata(7 downto 4) <= maze_desc(0)(6);
       readdata(3 downto 0) <= maze_desc(0)(7);
elsif address = "0000000000010100" then
       readdata(15 downto 12) <= maze_desc(0)(8);
       readdata(11 downto 8) <= maze_desc(0)(9);
       readdata(7 downto 4) <= maze_desc(0)(10);
       readdata(3 downto 0) <= maze_desc(0)(11);
elsif address = "0000000000010110" then
       readdata(15 downto 12) <= maze_desc(0)(12);
       readdata(11 downto 8) <= maze_desc(0)(13);
       readdata(7 downto 4) <= maze_desc(0)(14);
       readdata(3 downto 0) <= maze_desc(0)(15);

-- second row of maze
elsif address = "0000000000011000" then
       readdata(15 downto 12) <= maze_desc(1)(0);
       readdata(11 downto 8) <= maze_desc(1)(1);
       readdata(7 downto 4) <= maze_desc(1)(2);
       readdata(3 downto 0) <= maze_desc(1)(3);
elsif address = "0000000000011010" then
       readdata(15 downto 12) <= maze_desc(1)(4);
       readdata(11 downto 8) <= maze_desc(1)(5);
       readdata(7 downto 4) <= maze_desc(1)(6);
       readdata(3 downto 0) <= maze_desc(1)(7);
elsif address = "0000000000011100" then
       readdata(15 downto 12) <= maze_desc(1)(8);
       readdata(11 downto 8) <= maze_desc(1)(9);
       readdata(7 downto 4) <= maze_desc(1)(10);
       readdata(3 downto 0) <= maze_desc(1)(11);
elsif address = "0000000000011110" then
       readdata(15 downto 12) <= maze_desc(1)(12);
       readdata(11 downto 8) <= maze_desc(1)(13);
       readdata(7 downto 4) <= maze_desc(1)(14);
       readdata(3 downto 0) <= maze_desc(1)(15);

-- third row of maze
elsif address = "0000000000100000" then
       readdata(15 downto 12) <= maze_desc(2)(0);
       readdata(11 downto 8) <= maze_desc(2)(1);
       readdata(7 downto 4) <= maze_desc(2)(2);
       readdata(3 downto 0) <= maze_desc(2)(3);
elsif address = "0000000000100010" then
       readdata(15 downto 12) <= maze_desc(2)(4);
```

```
        readdata(11 downto 8) <= maze_desc(2)(5);
        readdata(7 downto 4) <= maze_desc(2)(6);
        readdata(3 downto 0) <= maze_desc(2)(7);
elsif address = "0000000000100100" then
        readdata(15 downto 12) <= maze_desc(2)(8);
        readdata(11 downto 8) <= maze_desc(2)(9);
        readdata(7 downto 4) <= maze_desc(2)(10);
        readdata(3 downto 0) <= maze_desc(2)(11);
elsif address = "0000000000100110" then
        readdata(15 downto 12) <= maze_desc(2)(12);
        readdata(11 downto 8) <= maze_desc(2)(13);
        readdata(7 downto 4) <= maze_desc(2)(14);
        readdata(3 downto 0) <= maze_desc(2)(15);

-- fourth row of maze
elsif address = "0000000000101000" then
        readdata(15 downto 12) <= maze_desc(3)(0);
        readdata(11 downto 8) <= maze_desc(3)(1);
        readdata(7 downto 4) <= maze_desc(3)(2);
        readdata(3 downto 0) <= maze_desc(3)(3);
elsif address = "0000000000101010" then
        readdata(15 downto 12) <= maze_desc(3)(4);
        readdata(11 downto 8) <= maze_desc(3)(5);
        readdata(7 downto 4) <= maze_desc(3)(6);
        readdata(3 downto 0) <= maze_desc(3)(7);
elsif address = "0000000000101100" then
        readdata(15 downto 12) <= maze_desc(3)(8);
        readdata(11 downto 8) <= maze_desc(3)(9);
        readdata(7 downto 4) <= maze_desc(3)(10);
        readdata(3 downto 0) <= maze_desc(3)(11);
elsif address = "0000000000101110" then
        readdata(15 downto 12) <= maze_desc(3)(12);
        readdata(11 downto 8) <= maze_desc(3)(13);
        readdata(7 downto 4) <= maze_desc(3)(14);
        readdata(3 downto 0) <= maze_desc(3)(15);

-- fifth row of maze
elsif address = "0000000000110000" then
        readdata(15 downto 12) <= maze_desc(4)(0);
        readdata(11 downto 8) <= maze_desc(4)(1);
        readdata(7 downto 4) <= maze_desc(4)(2);
        readdata(3 downto 0) <= maze_desc(4)(3);
elsif address = "0000000000110010" then
        readdata(15 downto 12) <= maze_desc(4)(4);
        readdata(11 downto 8) <= maze_desc(4)(5);
        readdata(7 downto 4) <= maze_desc(4)(6);
        readdata(3 downto 0) <= maze_desc(4)(7);
elsif address = "0000000000110100" then
        readdata(15 downto 12) <= maze_desc(4)(8);
        readdata(11 downto 8) <= maze_desc(4)(9);
        readdata(7 downto 4) <= maze_desc(4)(10);
        readdata(3 downto 0) <= maze_desc(4)(11);
elsif address = "0000000000110110" then
        readdata(15 downto 12) <= maze_desc(4)(12);
        readdata(11 downto 8) <= maze_desc(4)(13);
        readdata(7 downto 4) <= maze_desc(4)(14);
        readdata(3 downto 0) <= maze_desc(4)(15);

-- sixth row of maze
elsif address = "0000000000111000" then
        readdata(15 downto 12) <= maze_desc(5)(0);
        readdata(11 downto 8) <= maze_desc(5)(1);
        readdata(7 downto 4) <= maze_desc(5)(2);
        readdata(3 downto 0) <= maze_desc(5)(3);
elsif address = "0000000000111010" then
        readdata(15 downto 12) <= maze_desc(5)(4);
        readdata(11 downto 8) <= maze_desc(5)(5);
        readdata(7 downto 4) <= maze_desc(5)(6);
        readdata(3 downto 0) <= maze_desc(5)(7);
elsif address = "0000000000111100" then
        readdata(15 downto 12) <= maze_desc(5)(8);
```

```
        readdata(11 downto 8) <= maze_desc(5)(9);
        readdata(7 downto 4) <= maze_desc(5)(10);
        readdata(3 downto 0) <= maze_desc(5)(11);
elsif address = "0000000000111110" then
        readdata(15 downto 12) <= maze_desc(5)(12);
        readdata(11 downto 8) <= maze_desc(5)(13);
        readdata(7 downto 4) <= maze_desc(5)(14);
        readdata(3 downto 0) <= maze_desc(5)(15);

-- seventh row of maze
elsif address = "0000000001000000" then
        readdata(15 downto 12) <= maze_desc(6)(0);
        readdata(11 downto 8) <= maze_desc(6)(1);
        readdata(7 downto 4) <= maze_desc(6)(2);
        readdata(3 downto 0) <= maze_desc(6)(3);
elsif address = "0000000001000010" then
        readdata(15 downto 12) <= maze_desc(6)(4);
        readdata(11 downto 8) <= maze_desc(6)(5);
        readdata(7 downto 4) <= maze_desc(6)(6);
        readdata(3 downto 0) <= maze_desc(6)(7);
elsif address = "0000000001000100" then
        readdata(15 downto 12) <= maze_desc(6)(8);
        readdata(11 downto 8) <= maze_desc(6)(9);
        readdata(7 downto 4) <= maze_desc(6)(10);
        readdata(3 downto 0) <= maze_desc(6)(11);
elsif address = "0000000001000110" then
        readdata(15 downto 12) <= maze_desc(6)(12);
        readdata(11 downto 8) <= maze_desc(6)(13);
        readdata(7 downto 4) <= maze_desc(6)(14);
        readdata(3 downto 0) <= maze_desc(6)(15);

-- eigth row of maze
elsif address = "0000000001001000" then
        readdata(15 downto 12) <= maze_desc(7)(0);
        readdata(11 downto 8) <= maze_desc(7)(1);
        readdata(7 downto 4) <= maze_desc(7)(2);
        readdata(3 downto 0) <= maze_desc(7)(3);
elsif address = "0000000001001010" then
        readdata(15 downto 12) <= maze_desc(7)(4);
        readdata(11 downto 8) <= maze_desc(7)(5);
        readdata(7 downto 4) <= maze_desc(7)(6);
        readdata(3 downto 0) <= maze_desc(7)(7);
elsif address = "0000000001001100" then
        readdata(15 downto 12) <= maze_desc(7)(8);
        readdata(11 downto 8) <= maze_desc(7)(9);
        readdata(7 downto 4) <= maze_desc(7)(10);
        readdata(3 downto 0) <= maze_desc(7)(11);
elsif address = "0000000001001110" then
        readdata(15 downto 12) <= maze_desc(7)(12);
        readdata(11 downto 8) <= maze_desc(7)(13);
        readdata(7 downto 4) <= maze_desc(7)(14);
        readdata(3 downto 0) <= maze_desc(7)(15);

-- ninth row of maze
elsif address = "0000000001010000" then
        readdata(15 downto 12) <= maze_desc(8)(0);
        readdata(11 downto 8) <= maze_desc(8)(1);
        readdata(7 downto 4) <= maze_desc(8)(2);
        readdata(3 downto 0) <= maze_desc(8)(3);
elsif address = "0000000001010010" then
        readdata(15 downto 12) <= maze_desc(8)(4);
        readdata(11 downto 8) <= maze_desc(8)(5);
        readdata(7 downto 4) <= maze_desc(8)(6);
        readdata(3 downto 0) <= maze_desc(8)(7);
elsif address = "0000000001010100" then
        readdata(15 downto 12) <= maze_desc(8)(8);
        readdata(11 downto 8) <= maze_desc(8)(9);
        readdata(7 downto 4) <= maze_desc(8)(10);
        readdata(3 downto 0) <= maze_desc(8)(11);
elsif address = "0000000001010110" then
        readdata(15 downto 12) <= maze_desc(8)(12);
```

```
        readdata(11 downto 8) <= maze_desc(8)(13);
        readdata(7 downto 4) <= maze_desc(8)(14);
        readdata(3 downto 0) <= maze_desc(8)(15);

    -- tenth row of maze
    elsif address = "0000000001011000" then
        readdata(15 downto 12) <= maze_desc(9)(0);
        readdata(11 downto 8) <= maze_desc(9)(1);
        readdata(7 downto 4) <= maze_desc(9)(2);
        readdata(3 downto 0) <= maze_desc(9)(3);
    elsif address = "0000000001011010" then
        readdata(15 downto 12) <= maze_desc(9)(4);
        readdata(11 downto 8) <= maze_desc(9)(5);
        readdata(7 downto 4) <= maze_desc(9)(6);
        readdata(3 downto 0) <= maze_desc(9)(7);
    elsif address = "0000000001011100" then
        readdata(15 downto 12) <= maze_desc(9)(8);
        readdata(11 downto 8) <= maze_desc(9)(9);
        readdata(7 downto 4) <= maze_desc(9)(10);
        readdata(3 downto 0) <= maze_desc(9)(11);
    elsif address = "0000000001011110" then
        readdata(15 downto 12) <= maze_desc(9)(12);
        readdata(11 downto 8) <= maze_desc(9)(13);
        readdata(7 downto 4) <= maze_desc(9)(14);
        readdata(3 downto 0) <= maze_desc(9)(15);

    -- eleventh row of maze
    elsif address = "0000000001100000" then
        readdata(15 downto 12) <= maze_desc(10)(0);
        readdata(11 downto 8) <= maze_desc(10)(1);
        readdata(7 downto 4) <= maze_desc(10)(2);
        readdata(3 downto 0) <= maze_desc(10)(3);
    elsif address = "0000000001100010" then
        readdata(15 downto 12) <= maze_desc(10)(4);
        readdata(11 downto 8) <= maze_desc(10)(5);
        readdata(7 downto 4) <= maze_desc(10)(6);
        readdata(3 downto 0) <= maze_desc(10)(7);
    elsif address = "0000000001100100" then
        readdata(15 downto 12) <= maze_desc(10)(8);
        readdata(11 downto 8) <= maze_desc(10)(9);
        readdata(7 downto 4) <= maze_desc(10)(10);
        readdata(3 downto 0) <= maze_desc(10)(11);
    elsif address = "0000000001100110" then
        readdata(15 downto 12) <= maze_desc(10)(12);
        readdata(11 downto 8) <= maze_desc(10)(13);
        readdata(7 downto 4) <= maze_desc(10)(14);
        readdata(3 downto 0) <= maze_desc(10)(15);

    -- twelfth row of maze
    elsif address = "0000000001101000" then
        readdata(15 downto 12) <= maze_desc(11)(0);
        readdata(11 downto 8) <= maze_desc(11)(1);
        readdata(7 downto 4) <= maze_desc(11)(2);
        readdata(3 downto 0) <= maze_desc(11)(3);
    elsif address = "0000000001101010" then
        readdata(15 downto 12) <= maze_desc(11)(4);
        readdata(11 downto 8) <= maze_desc(11)(5);
        readdata(7 downto 4) <= maze_desc(11)(6);
        readdata(3 downto 0) <= maze_desc(11)(7);
    elsif address = "0000000001101100" then
        readdata(15 downto 12) <= maze_desc(11)(8);
        readdata(11 downto 8) <= maze_desc(11)(9);
        readdata(7 downto 4) <= maze_desc(11)(10);
        readdata(3 downto 0) <= maze_desc(11)(11);
    elsif address = "0000000001101110" then
        readdata(15 downto 12) <= maze_desc(11)(12);
        readdata(11 downto 8) <= maze_desc(11)(13);
        readdata(7 downto 4) <= maze_desc(11)(14);
        readdata(3 downto 0) <= maze_desc(11)(15);
```

```
--                          thirteenth row of maze
                          elsif address = "0000000001110000" then
                                 readdata(15 downto 12) <= maze_desc(12)(0);
                                 readdata(11 downto 8) <= maze_desc(12)(1);
                                 readdata(7 downto 4) <= maze_desc(12)(2);
                                 readdata(3 downto 0) <= maze_desc(12)(3);
                          elsif address = "0000000001110010" then
                                 readdata(15 downto 12) <= maze_desc(12)(4);
                                 readdata(11 downto 8) <= maze_desc(12)(5);
                                 readdata(7 downto 4) <= maze_desc(12)(6);
                                 readdata(3 downto 0) <= maze_desc(12)(7);
                          elsif address = "0000000001110100" then
                                 readdata(15 downto 12) <= maze_desc(12)(8);
                                 readdata(11 downto 8) <= maze_desc(12)(9);
                                 readdata(7 downto 4) <= maze_desc(12)(10);
                                 readdata(3 downto 0) <= maze_desc(12)(11);
                          elsif address = "0000000001110110" then
                                 readdata(15 downto 12) <= maze_desc(12)(12);
                                 readdata(11 downto 8) <= maze_desc(12)(13);
                                 readdata(7 downto 4) <= maze_desc(12)(14);
                                 readdata(3 downto 0) <= maze_desc(12)(15);


--                          fourteenth row of maze
                          elsif address = "0000000001111000" then
                                 readdata(15 downto 12) <= maze_desc(13)(0);
                                 readdata(11 downto 8) <= maze_desc(13)(1);
                                 readdata(7 downto 4) <= maze_desc(13)(2);
                                 readdata(3 downto 0) <= maze_desc(13)(3);
                          elsif address = "0000000001111010" then
                                 readdata(15 downto 12) <= maze_desc(13)(4);
                                 readdata(11 downto 8) <= maze_desc(13)(5);
                                 readdata(7 downto 4) <= maze_desc(13)(6);
                                 readdata(3 downto 0) <= maze_desc(13)(7);
                          elsif address = "0000000001111100" then
                                 readdata(15 downto 12) <= maze_desc(13)(8);
                                 readdata(11 downto 8) <= maze_desc(13)(9);
                                 readdata(7 downto 4) <= maze_desc(13)(10);
                                 readdata(3 downto 0) <= maze_desc(13)(11);
                          elsif address = "0000000001111110" then
                                 readdata(15 downto 12) <= maze_desc(13)(12);
                                 readdata(11 downto 8) <= maze_desc(13)(13);
                                 readdata(7 downto 4) <= maze_desc(13)(14);
                                 readdata(3 downto 0) <= maze_desc(13)(15);


--                          ffiteenth row of maze
                          elsif address = "0000000010000000" then
                                 readdata(15 downto 12) <= maze_desc(14)(0);
                                 readdata(11 downto 8) <= maze_desc(14)(1);
                                 readdata(7 downto 4) <= maze_desc(14)(2);
                                 readdata(3 downto 0) <= maze_desc(14)(3);
                          elsif address = "0000000010000010" then
                                 readdata(15 downto 12) <= maze_desc(14)(4);
                                 readdata(11 downto 8) <= maze_desc(14)(5);
                                 readdata(7 downto 4) <= maze_desc(14)(6);
                                 readdata(3 downto 0) <= maze_desc(14)(7);
                          elsif address = "0000000010000100" then
                                 readdata(15 downto 12) <= maze_desc(14)(8);
                                 readdata(11 downto 8) <= maze_desc(14)(9);
                                 readdata(7 downto 4) <= maze_desc(14)(10);
                                 readdata(3 downto 0) <= maze_desc(14)(11);
                          elsif address = "0000000010000110" then
                                 readdata(15 downto 12) <= maze_desc(14)(12);
                                 readdata(11 downto 8) <= maze_desc(14)(13);
                                 readdata(7 downto 4) <= maze_desc(14)(14);
                                 readdata(3 downto 0) <= maze_desc(14)(15);


--                          sixteenth row of maze
                          elsif address = "0000000010001000" then
```

```
        readdata(15 downto 12) <= maze_desc(15)(0);
        readdata(11 downto 8) <= maze_desc(15)(1);
        readdata(7 downto 4) <= maze_desc(15)(2);
        readdata(3 downto 0) <= maze_desc(15)(3);
    elsif address = "0000000010001010" then
        readdata(15 downto 12) <= maze_desc(15)(4);
        readdata(11 downto 8) <= maze_desc(15)(5);
        readdata(7 downto 4) <= maze_desc(15)(6);
        readdata(3 downto 0) <= maze_desc(15)(7);
    elsif address = "0000000010001100" then
        readdata(15 downto 12) <= maze_desc(15)(8);
        readdata(11 downto 8) <= maze_desc(15)(9);
        readdata(7 downto 4) <= maze_desc(15)(10);
        readdata(3 downto 0) <= maze_desc(15)(11);
    elsif address = "0000000010001110" then
        readdata(15 downto 12) <= maze_desc(15)(12);
        readdata(11 downto 8) <= maze_desc(15)(13);
        readdata(7 downto 4) <= maze_desc(15)(14);
        readdata(3 downto 0) <= maze_desc(15)(15);



    -- top row of obj
    elsif address = "0000000010010010" then
        readdata(15 downto 12) <= obj_desc(0)(0);
        readdata(11 downto 8) <= obj_desc(0)(1);
        readdata(7 downto 4) <= obj_desc(0)(2);
        readdata(3 downto 0) <= obj_desc(0)(3);
    elsif address = "0000000010010100" then
        readdata(15 downto 12) <= obj_desc(0)(4);
        readdata(11 downto 8) <= obj_desc(0)(5);
        readdata(7 downto 4) <= obj_desc(0)(6);
        readdata(3 downto 0) <= obj_desc(0)(7);
    elsif address = "0000000010010110" then
        readdata(15 downto 12) <= obj_desc(0)(8);
        readdata(11 downto 8) <= obj_desc(0)(9);
        readdata(7 downto 4) <= obj_desc(0)(10);
        readdata(3 downto 0) <= obj_desc(0)(11);
    elsif address = "0000000010011000" then
        readdata(15 downto 12) <= obj_desc(0)(12);
        readdata(11 downto 8) <= obj_desc(0)(13);
        readdata(7 downto 4) <= obj_desc(0)(14);
        readdata(3 downto 0) <= obj_desc(0)(15);

    -- second row of obj
    elsif address = "0000000010011010" then
        readdata(15 downto 12) <= obj_desc(1)(0);
        readdata(11 downto 8) <= obj_desc(1)(1);
        readdata(7 downto 4) <= obj_desc(1)(2);
        readdata(3 downto 0) <= obj_desc(1)(3);
    elsif address = "0000000010011100" then
        readdata(15 downto 12) <= obj_desc(1)(4);
        readdata(11 downto 8) <= obj_desc(1)(5);
        readdata(7 downto 4) <= obj_desc(1)(6);
        readdata(3 downto 0) <= obj_desc(1)(7);
    elsif address = "0000000010011110" then
        readdata(15 downto 12) <= obj_desc(1)(8);
        readdata(11 downto 8) <= obj_desc(1)(9);
        readdata(7 downto 4) <= obj_desc(1)(10);
        readdata(3 downto 0) <= obj_desc(1)(11);
    elsif address = "0000000010100000" then
        readdata(15 downto 12) <= obj_desc(1)(12);
        readdata(11 downto 8) <= obj_desc(1)(13);
        readdata(7 downto 4) <= obj_desc(1)(14);
        readdata(3 downto 0) <= obj_desc(1)(15);

    -- third row of obj
    elsif address = "0000000010100010" then
        readdata(15 downto 12) <= obj_desc(2)(0);
        readdata(11 downto 8) <= obj_desc(2)(1);
        readdata(7 downto 4) <= obj_desc(2)(2);
```

```
        readdata(3 downto 0) <= obj_desc(2)(3);
elsif address = "0000000010100100" then
        readdata(15 downto 12) <= obj_desc(2)(4);
        readdata(11 downto 8) <= obj_desc(2)(5);
        readdata(7 downto 4) <= obj_desc(2)(6);
        readdata(3 downto 0) <= obj_desc(2)(7);
elsif address = "0000000010100110" then
        readdata(15 downto 12) <= obj_desc(2)(8);
        readdata(11 downto 8) <= obj_desc(2)(9);
        readdata(7 downto 4) <= obj_desc(2)(10);
        readdata(3 downto 0) <= obj_desc(2)(11);
elsif address = "0000000010101000" then
        readdata(15 downto 12) <= obj_desc(2)(12);
        readdata(11 downto 8) <= obj_desc(2)(13);
        readdata(7 downto 4) <= obj_desc(2)(14);
        readdata(3 downto 0) <= obj_desc(2)(15);

-- fourth row of obj
elsif address = "0000000010101010" then
        readdata(15 downto 12) <= obj_desc(3)(0);
        readdata(11 downto 8) <= obj_desc(3)(1);
        readdata(7 downto 4) <= obj_desc(3)(2);
        readdata(3 downto 0) <= obj_desc(3)(3);
elsif address = "0000000010101100" then
        readdata(15 downto 12) <= obj_desc(3)(4);
        readdata(11 downto 8) <= obj_desc(3)(5);
        readdata(7 downto 4) <= obj_desc(3)(6);
        readdata(3 downto 0) <= obj_desc(3)(7);
elsif address = "0000000010101110" then
        readdata(15 downto 12) <= obj_desc(3)(8);
        readdata(11 downto 8) <= obj_desc(3)(9);
        readdata(7 downto 4) <= obj_desc(3)(10);
        readdata(3 downto 0) <= obj_desc(3)(11);
elsif address = "0000000010110000" then
        readdata(15 downto 12) <= obj_desc(3)(12);
        readdata(11 downto 8) <= obj_desc(3)(13);
        readdata(7 downto 4) <= obj_desc(3)(14);
        readdata(3 downto 0) <= obj_desc(3)(15);

-- fifth row of obj
elsif address = "0000000010110010" then
        readdata(15 downto 12) <= obj_desc(4)(0);
        readdata(11 downto 8) <= obj_desc(4)(1);
        readdata(7 downto 4) <= obj_desc(4)(2);
        readdata(3 downto 0) <= obj_desc(4)(3);
elsif address = "0000000010110100" then
        readdata(15 downto 12) <= obj_desc(4)(4);
        readdata(11 downto 8) <= obj_desc(4)(5);
        readdata(7 downto 4) <= obj_desc(4)(6);
        readdata(3 downto 0) <= obj_desc(4)(7);
elsif address = "0000000010110110" then
        readdata(15 downto 12) <= obj_desc(4)(8);
        readdata(11 downto 8) <= obj_desc(4)(9);
        readdata(7 downto 4) <= obj_desc(4)(10);
        readdata(3 downto 0) <= obj_desc(4)(11);
elsif address = "0000000010111000" then
        readdata(15 downto 12) <= obj_desc(4)(12);
        readdata(11 downto 8) <= obj_desc(4)(13);
        readdata(7 downto 4) <= obj_desc(4)(14);
        readdata(3 downto 0) <= obj_desc(4)(15);

-- sixth row of obj
elsif address = "0000000010111010" then
        readdata(15 downto 12) <= obj_desc(5)(0);
        readdata(11 downto 8) <= obj_desc(5)(1);
        readdata(7 downto 4) <= obj_desc(5)(2);
        readdata(3 downto 0) <= obj_desc(5)(3);
elsif address = "0000000010111100" then
        readdata(15 downto 12) <= obj_desc(5)(4);
        readdata(11 downto 8) <= obj_desc(5)(5);
        readdata(7 downto 4) <= obj_desc(5)(6);
```

```
        readdata(3 downto 0) <= obj_desc(5)(7);
elsif address = "0000000010111110" then
        readdata(15 downto 12) <= obj_desc(5)(8);
        readdata(11 downto 8) <= obj_desc(5)(9);
        readdata(7 downto 4) <= obj_desc(5)(10);
        readdata(3 downto 0) <= obj_desc(5)(11);
elsif address = "0000000011000000" then
        readdata(15 downto 12) <= obj_desc(5)(12);
        readdata(11 downto 8) <= obj_desc(5)(13);
        readdata(7 downto 4) <= obj_desc(5)(14);
        readdata(3 downto 0) <= obj_desc(5)(15);

-- seventh row of obj
elsif address = "0000000011000010" then
        readdata(15 downto 12) <= obj_desc(6)(0);
        readdata(11 downto 8) <= obj_desc(6)(1);
        readdata(7 downto 4) <= obj_desc(6)(2);
        readdata(3 downto 0) <= obj_desc(6)(3);
elsif address = "0000000011000100" then
        readdata(15 downto 12) <= obj_desc(6)(4);
        readdata(11 downto 8) <= obj_desc(6)(5);
        readdata(7 downto 4) <= obj_desc(6)(6);
        readdata(3 downto 0) <= obj_desc(6)(7);
elsif address = "0000000011000110" then
        readdata(15 downto 12) <= obj_desc(6)(8);
        readdata(11 downto 8) <= obj_desc(6)(9);
        readdata(7 downto 4) <= obj_desc(6)(10);
        readdata(3 downto 0) <= obj_desc(6)(11);
elsif address = "0000000011001000" then
        readdata(15 downto 12) <= obj_desc(6)(12);
        readdata(11 downto 8) <= obj_desc(6)(13);
        readdata(7 downto 4) <= obj_desc(6)(14);
        readdata(3 downto 0) <= obj_desc(6)(15);

-- eigth row of obj
elsif address = "0000000011001010" then
        readdata(15 downto 12) <= obj_desc(7)(0);
        readdata(11 downto 8) <= obj_desc(7)(1);
        readdata(7 downto 4) <= obj_desc(7)(2);
        readdata(3 downto 0) <= obj_desc(7)(3);
elsif address = "0000000011001100" then
        readdata(15 downto 12) <= obj_desc(7)(4);
        readdata(11 downto 8) <= obj_desc(7)(5);
        readdata(7 downto 4) <= obj_desc(7)(6);
        readdata(3 downto 0) <= obj_desc(7)(7);
elsif address = "0000000011001110" then
        readdata(15 downto 12) <= obj_desc(7)(8);
        readdata(11 downto 8) <= obj_desc(7)(9);
        readdata(7 downto 4) <= obj_desc(7)(10);
        readdata(3 downto 0) <= obj_desc(7)(11);
elsif address = "0000000011010000" then
        readdata(15 downto 12) <= obj_desc(7)(12);
        readdata(11 downto 8) <= obj_desc(7)(13);
        readdata(7 downto 4) <= obj_desc(7)(14);
        readdata(3 downto 0) <= obj_desc(7)(15);

-- ninth row of obj
elsif address = "0000000011010010" then
        readdata(15 downto 12) <= obj_desc(8)(0);
        readdata(11 downto 8) <= obj_desc(8)(1);
        readdata(7 downto 4) <= obj_desc(8)(2);
        readdata(3 downto 0) <= obj_desc(8)(3);
elsif address = "0000000011010100" then
        readdata(15 downto 12) <= obj_desc(8)(4);
        readdata(11 downto 8) <= obj_desc(8)(5);
        readdata(7 downto 4) <= obj_desc(8)(6);
        readdata(3 downto 0) <= obj_desc(8)(7);
elsif address = "0000000011010110" then
        readdata(15 downto 12) <= obj_desc(8)(8);
        readdata(11 downto 8) <= obj_desc(8)(9);
        readdata(7 downto 4) <= obj_desc(8)(10);
```

```
        readdata(3 downto 0) <= obj_desc(8)(11);
elsif address = "0000000011011000" then
        readdata(15 downto 12) <= obj_desc(8)(12);
        readdata(11 downto 8) <= obj_desc(8)(13);
        readdata(7 downto 4) <= obj_desc(8)(14);
        readdata(3 downto 0) <= obj_desc(8)(15);

-- tenth row of obj
elsif address = "0000000011011010" then
        readdata(15 downto 12) <= obj_desc(9)(0);
        readdata(11 downto 8) <= obj_desc(9)(1);
        readdata(7 downto 4) <= obj_desc(9)(2);
        readdata(3 downto 0) <= obj_desc(9)(3);
elsif address = "0000000011011100" then
        readdata (15 downto 12)<= obj_desc(9)(4);
        readdata(11 downto 8) <= obj_desc(9)(5);
        readdata(7 downto 4) <= obj_desc(9)(6);
        readdata(3 downto 0) <= obj_desc(9)(7);
elsif address = "0000000011011110" then
        readdata(15 downto 12) <= obj_desc(9)(8);
        readdata(11 downto 8) <= obj_desc(9)(9);
        readdata(7 downto 4) <= obj_desc(9)(10);
        readdata(3 downto 0) <= obj_desc(9)(11);
elsif address = "0000000011100000" then
        readdata(15 downto 12) <= obj_desc(9)(12);
        readdata(11 downto 8) <= obj_desc(9)(13);
        readdata(7 downto 4) <= obj_desc(9)(14);
        readdata(3 downto 0) <= obj_desc(9)(15);

-- eleventh row of obj
elsif address = "0000000011100010" then
        readdata(15 downto 12) <= obj_desc(10)(0);
        readdata(11 downto 8) <= obj_desc(10)(1);
        readdata(7 downto 4) <= obj_desc(10)(2);
        readdata(3 downto 0) <= obj_desc(10)(3);
elsif address = "0000000011100100" then
        readdata(15 downto 12) <= obj_desc(10)(4);
        readdata(11 downto 8) <= obj_desc(10)(5);
        readdata(7 downto 4) <= obj_desc(10)(6);
        readdata(3 downto 0) <= obj_desc(10)(7);
elsif address = "0000000011100110" then
        readdata(15 downto 12) <= obj_desc(10)(8);
        readdata(11 downto 8) <= obj_desc(10)(9);
        readdata(7 downto 4) <= obj_desc(10)(10);
        readdata(3 downto 0) <= obj_desc(10)(11);
elsif address = "0000000011101000" then
        readdata(15 downto 12) <= obj_desc(10)(12);
        readdata(11 downto 8) <= obj_desc(10)(13);
        readdata(7 downto 4) <= obj_desc(10)(14);
        readdata(3 downto 0) <= obj_desc(10)(15);

-- twelfth row of obj
elsif address = "0000000011101010" then
        readdata(15 downto 12) <= obj_desc(11)(0);
        readdata(11 downto 8) <= obj_desc(11)(1);
        readdata(7 downto 4) <= obj_desc(11)(2);
        readdata(3 downto 0) <= obj_desc(11)(3);
elsif address = "0000000011101100" then
        readdata(15 downto 12) <= obj_desc(11)(4);
        readdata(11 downto 8) <= obj_desc(11)(5);
        readdata(7 downto 4) <= obj_desc(11)(6);
        readdata(3 downto 0) <= obj_desc(11)(7);
elsif address = "0000000011101110" then
        readdata(15 downto 12) <= obj_desc(11)(8);
        readdata(11 downto 8) <= obj_desc(11)(9);
        readdata(7 downto 4) <= obj_desc(11)(10);
        readdata(3 downto 0) <= obj_desc(11)(11);
elsif address = "0000000011110000" then
        readdata(15 downto 12) <= obj_desc(11)(12);
        readdata(11 downto 8) <= obj_desc(11)(13);
        readdata(7 downto 4) <= obj_desc(11)(14);
```

```
                readdata(3 downto 0) <= obj_desc(11)(15);


--                thirteenth row of obj
              elsif address = "0000000011110010" then
                readdata(15 downto 12) <= obj_desc(12)(0);
                readdata(11 downto 8) <= obj_desc(12)(1);
                readdata(7 downto 4) <= obj_desc(12)(2);
                readdata(3 downto 0) <= obj_desc(12)(3);
              elsif address = "0000000011110100" then
                readdata(15 downto 12) <= obj_desc(12)(4);
                readdata(11 downto 8) <= obj_desc(12)(5);
                readdata(7 downto 4) <= obj_desc(12)(6);
                readdata(3 downto 0) <= obj_desc(12)(7);
              elsif address = "0000000011110110" then
                readdata(15 downto 12) <= obj_desc(12)(8);
                readdata(11 downto 8) <= obj_desc(12)(9);
                readdata(7 downto 4) <= obj_desc(12)(10);
                readdata(3 downto 0) <= obj_desc(12)(11);
              elsif address = "0000000011111000" then
                readdata(15 downto 12) <= obj_desc(12)(12);
                readdata(11 downto 8) <= obj_desc(12)(13);
                readdata(7 downto 4) <= obj_desc(12)(14);
                readdata(3 downto 0) <= obj_desc(12)(15);


--                fourteenth row of obj
              elsif address = "0000000011111010" then
                readdata(15 downto 12) <= obj_desc(13)(0);
                readdata(11 downto 8) <= obj_desc(13)(1);
                readdata(7 downto 4) <= obj_desc(13)(2);
                readdata(3 downto 0) <= obj_desc(13)(3);
              elsif address = "0000000011111100" then
                readdata(15 downto 12) <= obj_desc(13)(4);
                readdata(11 downto 8) <= obj_desc(13)(5);
                readdata(7 downto 4) <= obj_desc(13)(6);
                readdata(3 downto 0) <= obj_desc(13)(7);
              elsif address = "0000000011111110" then
                readdata(15 downto 12) <= obj_desc(13)(8);
                readdata(11 downto 8) <= obj_desc(13)(9);
                readdata(7 downto 4) <= obj_desc(13)(10);
                readdata(3 downto 0) <= obj_desc(13)(11);
              elsif address = "0000000100000000" then
                readdata(15 downto 12) <= obj_desc(13)(12);
                readdata(11 downto 8) <= obj_desc(13)(13);
                readdata(7 downto 4) <= obj_desc(13)(14);
                readdata(3 downto 0) <= obj_desc(13)(15);


--                fifteenth row of obj
              elsif address = "0000000100000010" then
                readdata(15 downto 12) <= obj_desc(14)(0);
                readdata(11 downto 8) <= obj_desc(14)(1);
                readdata(7 downto 4) <= obj_desc(14)(2);
                readdata(3 downto 0) <= obj_desc(14)(3);
              elsif address = "0000000100000100" then
                readdata(15 downto 12) <= obj_desc(14)(4);
                readdata(11 downto 8) <= obj_desc(14)(5);
                readdata(7 downto 4) <= obj_desc(14)(6);
                readdata(3 downto 0) <= obj_desc(14)(7);
              elsif address = "0000000100000110" then
                readdata(15 downto 12) <= obj_desc(14)(8);
                readdata(11 downto 8) <= obj_desc(14)(9);
                readdata(7 downto 4) <= obj_desc(14)(10);
                readdata(3 downto 0) <= obj_desc(14)(11);
              elsif address = "0000000100001000" then
                readdata(15 downto 12) <= obj_desc(14)(12);
                readdata(11 downto 8) <= obj_desc(14)(13);
                readdata(7 downto 4) <= obj_desc(14)(14);
                readdata(3 downto 0) <= obj_desc(14)(15);
```

```
        --                              sixteenth row of obj
                                        elsif address = "0000000100001010" then
                                                readdata(15 downto 12) <= obj_desc(15)(0);
                                                readdata(11 downto 8) <= obj_desc(15)(1);
                                                readdata(7 downto 4) <= obj_desc(15)(2);
                                                readdata(3 downto 0) <= obj_desc(15)(3);
                                        elsif address = "0000000100001100" then
                                                readdata(15 downto 12) <= obj_desc(15)(4);
                                                readdata(11 downto 8) <= obj_desc(15)(5);
                                                readdata(7 downto 4) <= obj_desc(15)(6);
                                                readdata(3 downto 0) <= obj_desc(15)(7);
                                        elsif address = "0000000100001110" then
                                                readdata(15 downto 12) <= obj_desc(15)(8);
                                                readdata(11 downto 8) <= obj_desc(15)(9);
                                                readdata(7 downto 4) <= obj_desc(15)(10);
                                                readdata(3 downto 0) <= obj_desc(15)(11);
                                        elsif address = "0000000100010000" then
                                                readdata(15 downto 12) <= obj_desc(15)(12);
                                                readdata(11 downto 8) <= obj_desc(15)(13);
                                                readdata(7 downto 4) <= obj_desc(15)(14);
                                                readdata(3 downto 0) <= obj_desc(15)(15);


                                        else
                                                 readdata <= foo;

                                        end if;
                                end if;

                        end if;
        end if;
 end process GetData;

DoIncPos : process (clk)
begin
        if rising_edge(clk) then
                if EndOfField = '1' then
                        maze_x_offset <= tmp_maze_x_offset;
                        maze_y_offset <= tmp_maze_y_offset;
                        cur_xpos_pix <= tmp_cur_xpos_pix;
                        cur_ypos_pix <= tmp_cur_ypos_pix;
                        which_char_tile <= tmp_char_tile;
                end if;
        end if;
end process DoIncPos;

end rtl;
```

Sources

http://wiibrew.org/index.php?title=Wiimote
http://libwiimote.sourceforge.net/
http://people.csail.mit.edu/albert/bluez-intro/